UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR INFORMATIONSSYSTEME

Özgür L. Özçep

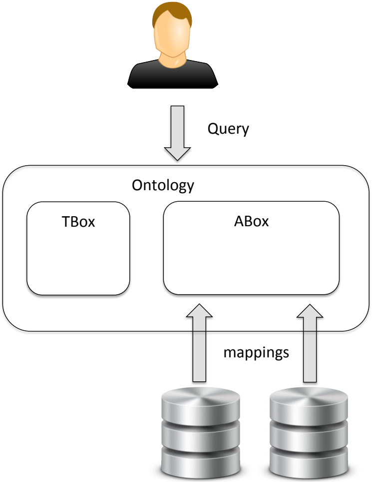
Ontology-Based Data Access II

Lecture 9: DL-Lite, Rewriting, Unfolding
11 June 2020

Informationssysteme CS4130
(Summer 2020)

Ontology-Based Data Access

- Use ontologies as interface
  ...
- to access (here: query)
- data stored in some format
  ...
- using mappings



- Description logics as ontology representation language
- Semantics

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

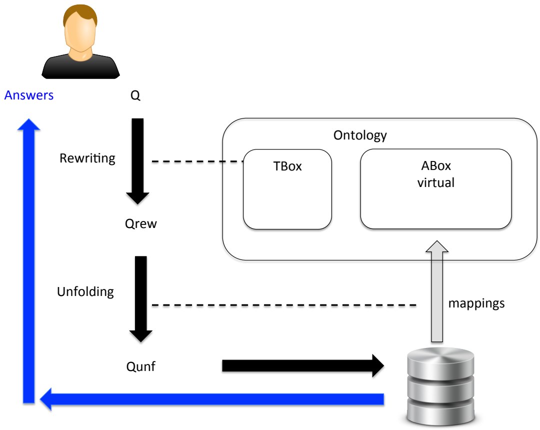
  https://www.inf.unibz.it/~calvanese/teaching/14-15-odbs/lecture-notes/
- Parts of Reasoning Web Summer School 2014 course by Ö. on Ontology-Based Data Access on Temporal and Streaming Data

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

  Temporal_and_Streaming_Data.pdf

OBDA in the Classical Sense

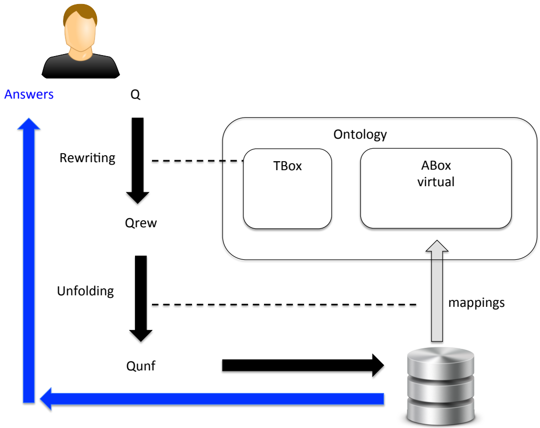
- Keep the data where they are because of large volume
- Abox not loaded into main memory, kept virtual



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 (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 use of functional role
  - Both due to rewritability

Properties

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

Example

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

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 (Eliminating Qualified Existentials on Rhs)

- 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

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

- DL-Lite$_{\mathcal{A}}$ extends DL-Lite$_{\mathcal{F}}$ and DL-Lite$_{\mathcal{R}}$ by allowing for
  - attribute expressions
    (binary 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: $(id\ basicConcept\ path_1, \ldots, path_n)$)

Identity assertions

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

- $\mathit{fillers}_\pi(i)$ = objects reachable from i via π

Example

$\mathit{hasChild} \circ \mathit{Woman}?$ = path connecting objects i with his/her daughters (its fillers)

- Identity assertions: $(\mathit{id}\ B\ \pi_1, \ldots, \pi_n))$
  Semantics: Different instances $i \neq i'$ of B are distinguished by at least one of their fillers: There is π_j such that

$$\mathit{fillers}_{\pi_j}(i) \neq \mathit{fillers}_{\pi_j}(i')$$

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

Definition

A positive inclusion (PI) has of the following forms:

$$A_1 \sqsubseteq A_2, \exists Q \sqsubseteq A_2, A_1 \sqsubseteq \exists Q_2, \exists Q_1 \sqsubseteq \exists Q_2, Q_1 \sqsubseteq Q_2$$

A negative inclusion (NI) has of the following forms:

$$A_1 \sqsubseteq \neg A_2, \exists Q_1 \sqsubseteq \neg A_2, A_1 \sqsubseteq \neg \exists Q_2, \exists Q_1 \sqsubseteq \neg \exists Q_2, Q_1 \sqsubseteq \neg Q_2$$

Example (Query answering by rewriting)

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

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

- $Q_{rew}(x) \leftarrow teaches(x, y), Course(y)$
- $Q_{rew}(x) \leftarrow teaches(x, y), teaches(\_, y)$
- $Q_{rew}(x) \leftarrow teaches(x, y)$ (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 a UCQ and is called the perfect rewriting of Q
- $ans(Q_{rew}, DB(\mathcal{A})) = \{schroedinger, einstein\} = cert(Q, (\mathcal{T}, \mathcal{A}))$

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 q ∈ PR' do
        forall g ∈ q do
            forall PI I ∈ TP do
                if I is applicable to g then
                    PR := PR ∪ {ApplyPI(q, g, I)}
                end
            end
        end
        forall 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 $P(x_1, x_2)$ 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: g_1, g_2 atoms in body of CQ q
  - Output: Returns a CQ q' obtained by applying to q the most general unifier between g_1 and g_2
  - Required for generating possibly unbound variables
- Anonymization
  - Substitute variables that are not bound with _.
  - Variable is bound iff it is a distinguished variable (=answer 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 see example before)
- Chase used here as tool for proving completeness
  - Answering Q_{rew} on the minimal Herbrand model of the abox is the same as answering Q on the chase.
  - Shown by induction on chase depth

Satisfiability Check for Ontologies

- In case an ontology is unsatisfiable, answer set becomes trivial: An unsatisfiable ontology entails all assertions

  $\Longrightarrow$ To determine correct answers need satisfiability check

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

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

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

Rewritability

Theorem

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

$\mathcal{O}$ is satisfiable iff $Q_N \vee Q_F$ is false.

- Note: $Q_N \vee Q_F$ is a UCQ$^{\neq}$ and hence an FOL query
- 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 loss of FOL-rewritability
- This can be proved using complexity theory and FOL (un-)definability arguments

Why Disallowing Qualified Existentials on Lhs

- Reachability in directed graphs is NLOGSPACE-complete
- X is FOL expressible iff $X \in AC^0$
  (and we know: $AC^0 \subsetneq NLOGSPACE$)
- Reachability reducible to QA with DL-lite and qualified existentials in lhs

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 $\mathfrak{G}, t$
- $\mathcal{A}_{\mathfrak{G},t}$ constructible in LOGSPACE from $\mathfrak{G}, s, t$.

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

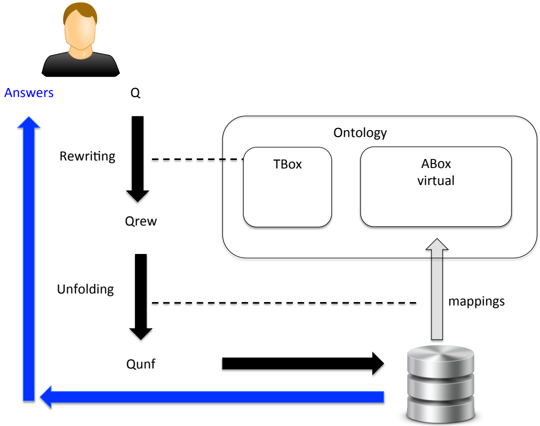
$$\forall x.manager(x) \rightarrow manages(x, x)$$

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

- Recent research on DLs: Re-introduce n-ary relations for $n > 2$

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

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

- $\psi(\vec{f}(\vec{x}))$: 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, TempSens, TC255, ‘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
- W3C standard (since 2012), http://www.w3.org/TR/r2rml/
- Read only (not allowed to write the RDFs view generated by the mappings)
- 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)

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

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

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` =: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

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

- Query Q_{Aux} with Aux-views

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

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
  - One approach Use different rewriting to ensure conciseness
  - Use additional knowledge on the data: integrity constraints, (H)-completeness
- Have a look at OBDA framework ontop (`https://github.com/ontop/ontop`)
  - Open source
  - available as Protege plugin
  - implementing many optimizations