UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR INFORMATIONSSYSTEME

Özgür L. Özçep

Ontology-Based Data Access

Lecture 7: Motivation, Description Logics

30 November, 2016

Foundations of Ontologies and Databases for Information Systems

CS5130 (Winter 16/17)

Recap of Lecture 6

Data Exchange

- Specific semantic integration scenario for two data sources with possibly different schemata
- Mapping $\mathcal{M} = (\sigma, \tau, M_{\sigma\tau}, M_\tau)$
  - σ: source schema
  - τ: target schema
  - $M_{\sigma\tau}$: source target dependencies (mostly: st-tgds)
  - M_τ: target dependencies
- Ultimate aim: For given σ instance find appropriate τ instance (solution) to do query answering on it
- Chase construction gave universal model: model with weakest assumptions
- Universal model may contain redundancies: considered cores; but as universal models are sufficient and cores may be costly, sticked to unversal models
- Looked at certain answering and the use of rewriting to yield certain answers

End of Recap

References

- ESSLLI 2010 Course by Calvanese and Zakharyaschev

  http://www.inf.unibz.it/~calvanese/teaching/2010-08-ESSLLI-DL-QA/
- 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/
- Course notes by Franz Baader on Description Logics
- 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

Ontology-Based Data Access as Integration

- Data Exchange can be considered as semantic integration purely on DB level

- OBDA can be considered as integration using an ontology
- Bridges DB world (closes world assumption) and ontology world (open world assumption)

Closed World Assumption

- DB theory: closed-world assumption (CWA)
  - All and only those facts mentioned in DB hold.
- Simple form of uncertain knowledge expressed by NULLs
  - For one incomplete DB there are many completions
  - Nonetheless: Type information on attribute constraints the possible attribute instances

- In DE incompleteness generated by different schemata
- Flight scenario: Source DB had no flight number, whilst target DB has
  $\Longrightarrow$ introduction of NULLs for flight number attribute

- Logical theories (ontologies) adhere to open world assumption (OWA)
  - If something is not told, then we do not know
  - Logical theories (ontologies) may have many models

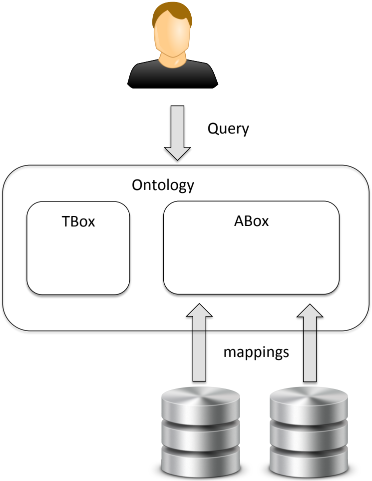
OBDA: Motivation and Overview

Ontology-Based Data Access

- Use ontologies as interface

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

  ...
- using mappings

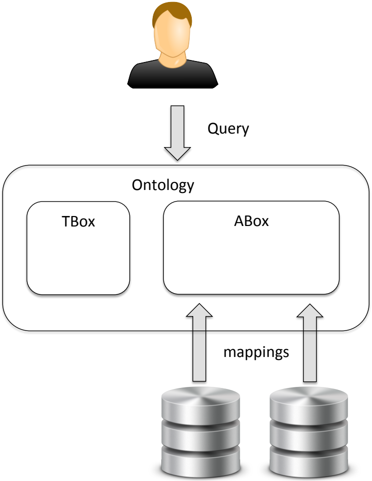



Ontology (Australia) Paradise


Backend DB Dungeon

Ontology-Based Data Access

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



Ontologies

- Ontologies are structures of the form $\mathcal{O} = (\sigma, \mathcal{T}, \mathcal{A})$
  - Signature σ: Non-logical vocabulary
    $\sigma = \mathit{Const}_\sigma \cup \mathit{Conc}_\sigma \cup \mathit{Role}_\sigma$
  - TBox $\mathcal{T}$: set of σ-axioms in some logic to capture terminological knowledge
    This lecture: ontologies represented in Description Logics (DLs)
  - ABox $\mathcal{A}$: set of σ-axioms in (same logic) to capture assertional/contingential knowledge
- Note: Sometimes only TBox termed ontology
- Semantics defined on the basis of σ-interpretations $\mathcal{I}$
  - $\mathcal{I} \models \mathit{Ax}$ iff $\mathcal{I}$ makes all axioms in Ax true
  - $\mathit{Mod}(\mathit{Ax}) = \{\mathcal{I} \models \mathit{Ax}\}$

Ontologies

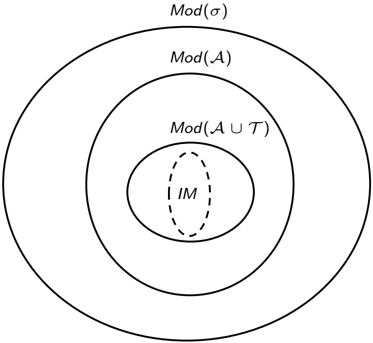
- Ontologies are structures of the form $\mathcal{O} = (\sigma, \mathcal{T}, \mathcal{A})$
  - Signature σ: Non-logical vocabulary
    $\sigma = \mathit{Const}_\sigma \cup \mathit{Conc}_\sigma \cup \mathit{Role}_\sigma$
  - TBox $\mathcal{T}$: set of σ-axioms in some logic to capture terminological knowledge
    This lecture: ontologies represented in Description Logics (DLs)
  - ABox $\mathcal{A}$: set of σ-axioms in (same logic) to capture assertional/contingential knowledge

- Note: Sometimes only TBox termed ontology

- Semantics defined on the basis of σ-interpretations $\mathcal{I}$
  - $\mathcal{I} \models \mathit{Ax}$ iff $\mathcal{I}$ makes all axioms in Ax true
  - $\mathit{Mod}(\mathit{Ax}) = \{\mathcal{I} \models \mathit{Ax}\}$

General Idea

- $\mathcal{A}$: Represents facts in domain of interest
- Open world assumption: $Mod(\mathcal{A})$ is not a singleton
- $\mathcal{T}$: Constrains $Mod(\mathcal{A})$ with intended σ readings
- Usually one has only approximations of intended models IM
- Realize inference services on the basis of the constrained interpretations



WARNING: A Misconception

- With ontologies one does not declare data structures

- ABox data in most cases show pattern of data structures
- One does not have to re-model patterns/constraints in the ABox data
  - Knowing "All A are B" in the ABox is different from stipulating $A \sqsubseteq B$ (the former is known as **integrity constraint**)
  - Add $A \sqsubseteq B$, if you need to handle this relation for objects not mentioned in the ABox

- Motto: Keep the TBox simple

Ontology-Based Data Access

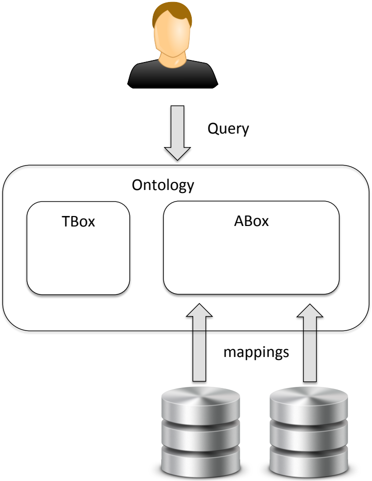
- Use ontologies as interface

  ...

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

  ...

- using mappings



Reasoning Services

- Different standard and nonstandard reasoning services exists
- May be reducible to each other
- Examples: consistency check, subsumption check, taxonomy calculations, most specific subsumer, most specific concept, matching, ...

- In classical OBDA focus on
  - Consistency checking: $Mod(\mathcal{A} \cup \mathcal{T}) \neq \emptyset$.
  - Query answering
- Next to ABox and TBox language query language QL over σ is a relevant factor for OBDA
- Certain query answering

$$cert(\psi(\vec{x}), \mathcal{T} \cup \mathcal{A}) \quad = \quad \{\vec{a} \in (Const_\sigma)^n \mid \mathcal{T} \cup \mathcal{A} \models \psi[\vec{x}/\vec{a}]\}$$

Reasoning Services

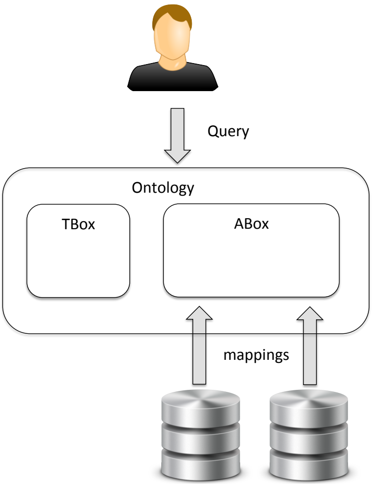
- Different standard and nonstandard reasoning services exists
- May be reducible to each other
- Examples: consistency check, subsumption check, taxonomy calculations, most specific subsumer, most specific concept, matching, ...

- In classical OBDA focus on
  - Consistency checking: $Mod(\mathcal{A} \cup \mathcal{T}) \neq \emptyset$.
  - Query answering
- Next to ABox and TBox language query language QL over σ is a relevant factor for OBDA
- Certain query answering

$$cert(\psi(\vec{x}), \mathcal{T} \cup \mathcal{A}) \quad = \quad \{\vec{a} \in (Const_\sigma)^n \mid \mathcal{T} \cup \mathcal{A} \models \psi[\vec{x}/\vec{a}]\}$$

Ontology-Based Data Access

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

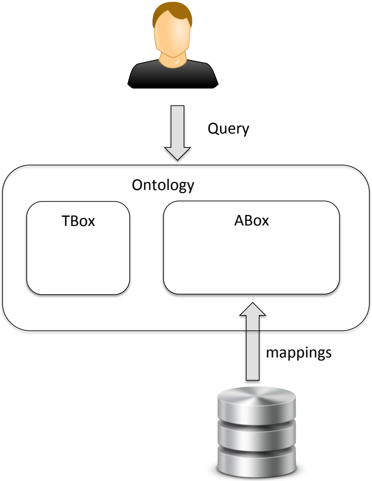


Backend Data Sources

- Classically: relational SQL DBs with static data
- Possible extensions: non-SQL DBs
  - datawarehouse repositories for statistical applications
  - pure logfiles
  - RDF repositories
- Non-static data
  - historical data (stored in temporal DB)
  - dynamic data coming in streams
- Originally intended for multiple DBs but ...

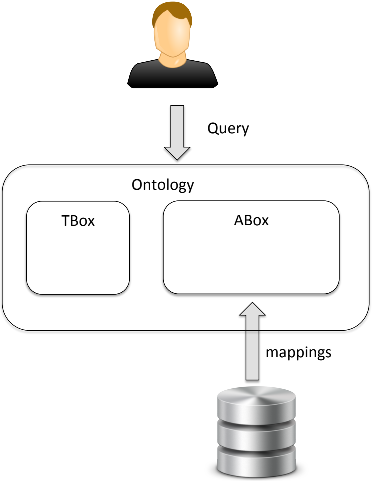
Federation

- ... we would have to deal with federation
- not trivial in classical OBDA ...
- because one has to integrate data from different DBs
- Ignore federation aspect: we have one DB but possibly many tables



Ontology-Based Data Access

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



Mappings

- Mappings have an important crucial role in OBDA
- Lift data to the ontology level
  - Data level: (nearly) close world
  - Ontology Level: open world

Schema of Mappings

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

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

- Mappings M over backend sources generates ABox $\mathcal{A}(M, DB)$.

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: $\quad Sens(x) \wedge name(x, y) \longleftarrow$

```
SELECT f(SID) as x,  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: $\mathit{Sens}(x) \wedge \mathit{name}(x, y) \longleftarrow$

```
SELECT f(SID) as x,  Sname as y FROM SENSOR
```

- the row data in `SENSOR` table

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

- generates facts

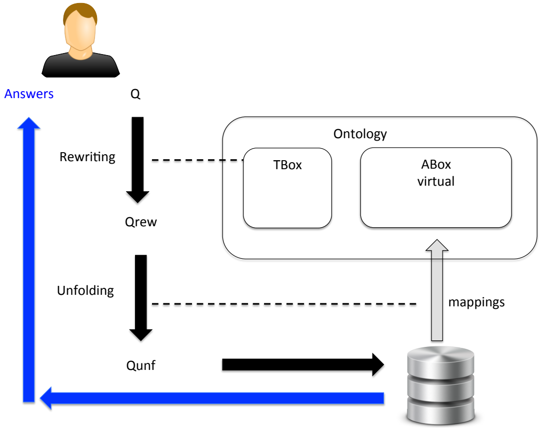
$\mathit{Sens}(f(123)), \mathit{name}(f(123), \mathit{TempSens}) \in \mathcal{A}(m, DB)$

(Strange) Maps of a Different Kind

- Jacobs strange maps:
  http://bigthink.com/articles?blog=strange-maps

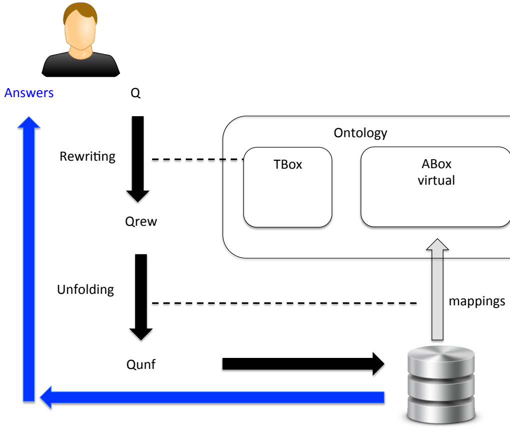
OBDA in the Classical Sense

- Keep the data where they are because of large volume
- ABox is virtual (no materialization)



OBDA in the Classical Sense

- First-order logic (FOL) perfect rewriting + unfolding for realizing reasoning services



OBDA in the Classical Sense

- $\mathcal{T}$ language: Some logic of the DL-Lite family
- $\mathcal{A}$ language: assertions of the form $A(a), R(a, b)$
- QL : Unions of conjuctive queries (UCQs)
- Language of $Qrew$: safe FOL
- Allows for perfect rewriting (of consistency checking and) UCQ answering

$$cert(Q, (\sigma, \mathcal{T}, \mathcal{A})) = cert(Qrew, \mathcal{A}) = ans(Qrew, DB(\mathcal{A}))$$

- and unfolding

$$cert(Q, (\sigma, \mathcal{T}, \mathcal{A}(M, DB))) = ans(Qunf, DB)$$

- Note that query language over DB is relevant for possibility of unfolding

OBDA in the Classical Sense

- $\mathcal{T}$ language: Some logic of the DL-Lite family
- $\mathcal{A}$ language: assertions of the form $A(a), R(a,b)$
- QL : Unions of conjuctive queries (UCQs)
- Language of $Qrew$: safe FOL
- Allows for perfect rewriting (of consistency checking and) UCQ answering

$$cert(Q, (\sigma, \mathcal{T}, \mathcal{A})) = cert(Qrew, \mathcal{A}) = ans(Qrew, DB(\mathcal{A}))$$

- and unfolding

$$cert(Q, (\sigma, \mathcal{T}, \mathcal{A}(M, DB))) = ans(Qunf, DB)$$

- Note that query language over DB is relevant for possibility of unfolding

OBDA in the Classical Sense

- $\mathcal{T}$ language: Some logic of the DL-Lite family
- $\mathcal{A}$ language: assertions of the form $A(a), R(a, b)$
- QL : Unions of conjuctive queries (UCQs)
- Language of *Qrew*: safe FOL
- Allows for perfect rewriting (of consistency checking and) UCQ answering

$$cert(Q, (\sigma, \mathcal{T}, \mathcal{A})) = cert(Qrew, \mathcal{A}) = ans(Qrew, DB(\mathcal{A}))$$

- and unfolding

$$cert(Q, (\sigma, \mathcal{T}, \mathcal{A}(M, DB))) = ans(Qunf, DB)$$

- Note that query language over DB is relevant for possibility of unfolding

Extended OBDA

- Use more expressive TBox language
  - ABDEO (Accessing very big data using expressive ontologies)
  - Rewritability for UCQs not guaranteed
  - Materialize ABox and use ABox modularization to answer queries
- Use different (more expressive) QL
  - E.g. SPARQL instead of UCQ; but no full existentials in combination with DL-Lite
  - OWL2QL + SPARQL used in Optique platform
- Use different reasoning/rewriting paradigm
  - e.g. combined rewriting: First enhance ABox with TBox information and then rewrite
  - Streaming

Ontologies and Description Logics

Description Logics

Definition

Description logics (DLs) are logics for use in knowledge representation with special attention on a good balance of **expressibility** and **feasibility** of reasoning services

- Can be mapped to fragments of FOL
- Use
  - as ontology representation language for conceptual modeling
  - in particular in the semantic web
  - Formal counterpart of standard web ontology language (OWL)
  - and in particular for ontology-based data access (OBDA)
- Have been investigated for ca. 30 years now
  - Many theoretical insights on various different purpose DLs
  - General-purpose reasoners (RacerPro, Fact++, ...) and specific reasoners (Quest,...)
  - Various editing tools (most notably Protege)

Family of DLs

- Variable-free logics centered around concepts
- concepts = one-ary predicates in FOL = classes in OWL

Example (Concepts)

- $Students$ ("students")
- $Students \sqcap Male$ (" Male students")
- $\exists attends.MathCourse$ ("Those attending a math course")
- $\forall hasFriends.Freaks$ ("Those having only freaks as friends")
- $Person \sqcap \forall attends.(Course \sqcap \neg easy)$
  ("Persons attending only non-easy courses")

An (Semi-)Expressive Logic: $\mathcal{ALC}$

- **Vocabulary:** constants N_i, atomic concepts N_C, roles N_R
- **Concept(description)s: syntax**

$$C \quad ::= \quad A \quad (\text{for } A \in N_C) \mid C \sqcap C \mid C \sqcup C \mid \neg C \mid \forall r.C \mid \exists r.C \ (\text{for } r \in N_R) \mid \bot \mid \top$$

- **Concept(description)s: semantics**

- Interpretation $\mathcal{I} = (\underbrace{\Delta^{\mathcal{I}}}_{domain}, \overbrace{\cdot^{\mathcal{I}}}^{denotation\ function})$
- $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ for all $A \in N_C$
- $c^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ for all $c \in N_i$
- $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ for all $r \in N_r$

- $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$
- $(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$
- $\neg C = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
- $(\forall r.C)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid$ for all $e \in \Delta^{\mathcal{I}}$: If $(d, e) \in r^{\mathcal{I}}$ then $e \in C^{\mathcal{I}}\}$
- $(\exists r.C)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid$ there is $e \in \Delta^{\mathcal{I}}$ s.t. $(d, e) \in r^{\mathcal{I}}$ and $e \in C^{\mathcal{I}}\}$

TBox and ABox

- **Terminological Box (TBox)** $\mathcal{T}$
  - Finite set of general concept inclusions (GCIs)
  - GCI: axioms of form $C \sqsubseteq D$ (for arbitrary concept descriptions) $C \equiv D$ abbreviates $\{C \sqsubseteq D, D \sqsubseteq C\}$
  - Semantics: $\mathcal{I} \models C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$.

- **Assertional Box (ABox)** $\mathcal{A}$
  - Finite set of assertions
  - Assertion: $C(a)$ (concept assertion), $r(a, b)$ (role assertion)
  - Semantics: $\mathcal{I} \models a^{\mathcal{I}} \in C^{\mathcal{I}}$, $\mathcal{I} \models r(a, b)$ iff $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$.

- **Ontology**: $(\sigma, \mathcal{T}, \mathcal{A})$

We follow the bad CS practice of calling KBs in DLs ontologies. We apologize to all philosophers for this use ;)

Example (University)

$$
\begin{aligned}
\mathcal{T} &= \{\ \mathit{GradStudent} \sqsubseteq \mathit{Student}, \\
&\quad\ \mathit{GradStudent} \sqsubseteq \exists \mathit{takesCourse}.\mathit{GradCourse}\ \} \\
\mathcal{A} &= \{\ \mathit{GradStudent}(\mathit{john})\ \}
\end{aligned}
$$

Consider the following interpretations

Example (University)

$$
\begin{aligned}
\mathcal{T} &= \{\ GradStudent \sqsubseteq Student, \\
&\quad GradStudent \sqsubseteq \exists takesCourse.GradCourse\ \} \\
\mathcal{A} &= \{\ GradStudent(john)\ \}
\end{aligned}
$$

Consider the following interpretations

- $\mathcal{I}_1$:
  - $john^{\mathcal{I}_1} = j$
  - $GradStudent^{\mathcal{I}_1} = \{j\}$
  - $Student^{\mathcal{I}_1} = \{j\}$
  - $GradCourse^{\mathcal{I}_1} = \{s\}$
  - $takesCourse^{\mathcal{I}_1} = \{(j, s)\}$
- $\mathcal{I}_1 \models \mathcal{T} \cup \mathcal{A}$

- $\mathcal{I}_2$:
  - $john^{\mathcal{I}_2} = j$
  - $GradStudent^{\mathcal{I}_2} = \{j\}$
  - $Student^{\mathcal{I}_2} = \{j\}$
  - $GradCourse^{\mathcal{I}_2} = \{j\}$
  - $takesCourse^{\mathcal{I}_2} = \{(j, j)\}$
- $\mathcal{I}_2 \models \mathcal{T} \cup \mathcal{A}$

Example (University)

$$
\begin{aligned}
\mathcal{T} &= \{\ \mathit{GradStudent} \sqsubseteq \mathit{Student}, \\
&\quad\ \mathit{GradStudent} \sqsubseteq \exists \mathit{takesCourse}.\mathit{GradCourse}\ \} \\
\mathcal{A} &= \{\ \mathit{GradStudent}(\mathit{john})\ \}
\end{aligned}
$$

Consider the following interpretations

- $\mathcal{I}_3$:
  - $\mathit{john}^{\mathcal{I}_1} = j$
  - $\mathit{GradStudent}^{\mathcal{I}_1} = \{j\}$
  - $\mathit{Student}^{\mathcal{I}_1} = \{j\}$
  - $\mathit{GradCourse}^{\mathcal{I}_1} = \emptyset$
  - $\mathit{takesCourse}^{\mathcal{I}_1} = \emptyset$
- $\mathcal{I}_3 \not\models \mathcal{T} \cup \mathcal{A}$

Stricter notion of TBox

- Above definition of TBox very general
  - "Meanings" of concept names determined only implicitly in the whole ontology
  - No guarantee for unique extensions

- Early notion of TBox more related to idea of **explicitly defining** concept names
- $C \equiv D$ used as abbreviation for $C \sqsubseteq D$ and $D \sqsubseteq C$
- Concept definition: $A \equiv D$ (where A atomic)

Definition

A **TBox in a strict sense** is a finite set of concept definitions not defining a concept multiple times or in a cyclic manner. **Defined concepts** occur on the lhs, **primitive concept** on the rhs of definitions.

Implicit vs. Explicit Definability

- Sometimes a general TBox may fix the denotation of a concept name w.r.t. denotations of the others $\Longrightarrow$ implicit definability
- Maybe then it can also be defined explicitly?

Definition

Given an FOL theory Ψ over signature σ and a predicate symbol R.

- R is **implicitly defined** in Ψ iff
  for any two models $\mathfrak{A} \models \Psi$ and $\mathfrak{B} \models \Psi$ agreeing on $\sigma \setminus \{R\}$ one has $R^{\mathfrak{A}} = R^{\mathfrak{B}}$.
- R is **explicitly defined** in Ψ by a formula $\phi(\vec{x})$ not containing R iff $\Psi \models \forall\vec{x}R(\vec{x}) \leftrightarrow \phi(\vec{x})$

Beth Definability Theorem

- For FOL both notions of definition coincide

Theorem

An FOL theory defines a predicate implicitly iff it defines it explicitly

- Though DLs are embedable into FOL, this coincidence does not transfer necessarily to DL
- At least it does for $\mathcal{ALC}$ theories

  Lit: B. ten Cate, E. Franconi, and I. Seylan. Beth definability in expressive description logics. J. Artif. Int. Res., 48(1): 347–414, Oct. 2013.

Reasoning services

- Semantical notions as in FOL but additional notions due to focus on concepts
- Let $\mathcal{O} = (\sigma, \mathcal{T}, \mathcal{A})$

Definition (Basic Semantical Notions)

- **Model:** $\mathcal{I} \models \mathcal{O}$ iff $\mathcal{I} \models \mathcal{T} \cup \mathcal{A}$
- **Satisfiability:** $\mathcal{O}$ is satisfiable iff $\mathcal{T} \cup \mathcal{A}$ is satisfiable
- **Coherence:** $\mathcal{O}$ is coherent iff $\mathcal{T} \cup \mathcal{A}$ has a model $\mathcal{I}$ s.t. for all concept names $A^{\mathcal{I}} \neq \emptyset$
- **Concept satisfiability:** C is satisfiable w.r.t. $\mathcal{O}$ iff there is $\mathcal{I} \models \mathcal{O}$ s.t. $C^{\mathcal{I}} \neq \emptyset$
- **Subsumption:** C is subsumed by D w.r.t. $\mathcal{O}$ iff $\mathcal{O} \vDash C \sqsubseteq D$ iff $\mathcal{T} \cup \mathcal{A} \vDash C \sqsubseteq D$
- **Instance check:** a is an instance of C w.r.t. $\mathcal{O}$ iff $\mathcal{O} \vDash C(a)$

Reduction Examples

- Many of the semantical notions are reducible to each other
- We give only one example in the following exercise

Wake-Up Exercise

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).

Extended Reasoning Services

Definition

- **Instance retrieval:** Find all constants x s.t. $\mathcal{O} \vDash C(x)$
- **Query answering:** Certain answers
  $cert(\phi(x), \mathcal{O}) = \{\vec{a} \in Const_\sigma \mid \mathcal{O} \vDash \phi[\vec{x}/\vec{a}]\}$
- **Classification**: Compute the subsumption hierarchy of all concept names
- **Realization:** Compute the most specific concept name to which a given constant belongs
- Pinpointing, matching, ...

Example (Certain Answers for Conjunctive Queries)

$$\begin{array}{rcl} \mathcal{T} & = & \{\, \top \sqsubseteq \mathit{Male} \sqcup \mathit{Female}, \mathit{Male} \sqcap \mathit{Female} \sqsubseteq \bot \,\} \\ \mathcal{A} & = & \{\, \mathit{friend}(\mathit{john}, \mathit{susan}), \mathit{friend}(\mathit{john}, \mathit{andrea}), \mathit{female}(\mathit{susan}), \\ & & \mathit{loves}(\mathit{susan}, \mathit{andrea}), \quad \mathit{loves}(\mathit{andrea}, \mathit{bill}), \mathit{Male}(\mathit{bill}) \,\} \end{array}$$

$$Q(x) \quad = \quad \exists y, z(\mathit{friend}(x, y) \wedge \mathit{Female}(y) \wedge \mathit{loves}(y, z) \wedge \mathit{Male}(z))$$

- $\mathit{cert}(Q(x), \mathcal{O}) =?$

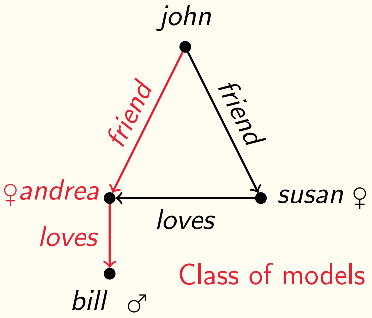
- We have to consider all possible models of the ontology
- But there actually two classes:
  Andrea is male vs. Andrea is not male.

Example (Certain Answers for Conjunctive Queries)

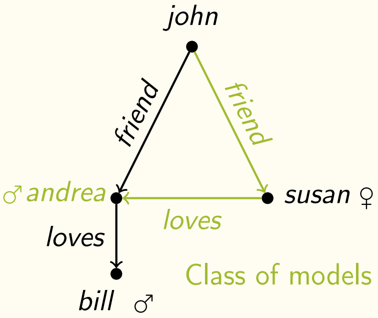
$$\mathcal{T} = \{ \top \sqsubseteq \mathit{Male} \sqcup \mathit{Female}, \mathit{Male} \sqcap \mathit{Female} \sqsubseteq \bot \}$$

$$\mathcal{A} = \{ \mathit{friend}(\mathit{john}, \mathit{susan}), \mathit{friend}(\mathit{john}, \mathit{andrea}), \mathit{female}(\mathit{susan}), \mathit{loves}(\mathit{susan}, \mathit{andrea}), \quad \mathit{loves}(\mathit{andrea}, \mathit{bill}), \mathit{Male}(\mathit{bill}) \}$$

$$Q(x) = \exists y, z(\mathit{friend}(x, y) \wedge \mathit{Female}(y) \wedge \mathit{loves}(y, z) \wedge \mathit{Male}(z))$$



Class of models 1



Class of models 2

$\mathit{cert}(Q(x), \mathcal{O}) = \{\mathit{john}\}$

Embedding into FOL

- Most DLs (such as $\mathcal{ALC}$) can be embedded into FOL
- Notion of embedding is well-defined as FOL structures are used for semantics of DLs.

- Correspondence idea
  Concept names = unary predicates, roles = binary predicates, GCI = $\forall$ rules
- Define for any concept description and variable x its corresponding x-open formula $\tau_x(C)$
  - $\tau_x(A) = A(x)$
  - $\tau_x(C \sqcap D) = \tau_x(C) \wedge \tau_x(D)$
  - $\tau_x(C \sqcup D) = \tau_x(C) \vee \tau_x(D)$
  - $\tau_x(\neg C) = \neg\tau_x(C)$
  - $\tau_x(\forall r.C) = \forall y(r(x,y) \rightarrow \tau_y(C))$
  - $\tau_x(\exists r.C) = \exists y(r(x,y) \wedge \tau_y(C))$
- ABox axioms not changed
- TBox axioms: $C \sqsubseteq D$ becomes $\forall x(\tau_x(C) \rightarrow \tau_x(D))$

Embedding into FOL

- For translation two variables are sufficient ("2 finger movement")
- Hence: DLs embeddable into known **2-variable fragment** of FOL
- Also the fragment is a **guarded fragment**: one quantifies over variables fixed within atom.

Wake-Up Exercise

Calculate $\tau_x(\forall r.(A \sqcap \exists r.B))$ using only two variables.

Embedding into FOL

- For translation two variables are sufficient ("2 finger movement")
- Hence: DLs embeddable into known **2-variable fragment** of FOL
- Also the fragment is a **guarded fragment**: one quantifies over variables fixed within atom.

Wake-Up Exercise

Calculate $\tau_x(\forall r.(A \sqcap \exists r.B))$ using only two variables.

Solution:

$$\forall y[r(x,y) \rightarrow (A(y) \wedge \exists x[r(y,x) \wedge B(x)])]$$

NB: There are free and bound occurrences of x

DL Family

- Different DLs for different purposes
  - What is more important: Expressivity or feasibility?
  - Which kinds of reasoning services does one have to provide?
- Differences regarding
  - the allowed set of concept constructors
  - the allowed set of role constructors
  - the allowed types of TBox axioms
  - the allowed types of ABox axioms
  - the allowance of concrete domains and attributes (such as *hasAge* with range the domain of integers)

Family of DLs and their Namings

- $\mathcal{AL}$: attributive language
- $\mathcal{C}$: (full) complement/negation
- $\mathcal{I}$: inverse roles $((r^{-1})^{\mathcal{I}} = \{(d, e) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (e, d) \in r^{\mathcal{I}}\})$
- $\mathcal{H}$: role inclusions $(hasFather \sqsubseteq hasParent\ )$

- $\mathcal{S}$: $\mathcal{ALC}$ + transitive roles $(trans\ isReachable)$
- $\mathcal{N}$: unqualified number restrictions
  $((\geq n\ r)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \#(\{e \mid (d, e) \in r^{\mathcal{I}}\}) \geq n))$
- $\mathcal{O}$: nominals $\{b\}^{\mathcal{I}} = \{b^{\mathcal{I}}\}$
- $\mathcal{Q}$: qualified number restrictions
  $((\geq n\ r.C)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \#(\{e \mid (d, e) \in r^{\mathcal{I}}\} \text{ and } e \in C^{\mathcal{I}}) \geq n))$
- $\mathcal{F}$: functionality constraints $\mathcal{I} \models (func\ R)$ iff $R^{\mathcal{I}}$ is a function
- $\mathcal{R}$: role chains and $\exists R.Self$ $(hasFather \circ hasMother \sqsubseteq hasgrandMa)$
  $(narcist \equiv \exists likes.Self)$

- OWL 2 is $\mathcal{SROIQ}$

Lightweight DLs

- Lightweight DLs favor feasibility over expressibility by, roughly, dis-allowing disjunction
- In principle three lightweight logics that have corresponding OWL 2 profiles

- $\mathcal{EL}$ (OWL 2 EL)
  - No inverses, no negation, no $\forall$
  - polynomial time algorithms for all the standard reasoning tasks with large ontologies
- DL-Lite (OWL 2 QL)
  - TBox: No qualified existentials on lhs
  - Feasible CQ answering using rewriting and unfolding leveraging RDBS technology
- RL (OWL 2 RL)
  - TBox restriction: "Only concept names on the rhs"
  - Polynomial time algorithms leveraging rule-extended database technologies operating directly on RDF triples

Comparison

	RL	EL	QL
inverse roles	+	-	+
rhs qual. exist	-	+	+
lhs qual. exist.	+	+	-

Complexity

- A nearly complete picture of reasoning services for DLs
- Research in DL community as of now resembles complexity farming
- DL complexity navigator:
  `http://www.cs.man.ac.uk/~ezolin/dl` (Last update 2013)

Tableaux Calculus for $\mathcal{ALC}$

- Efficient calculi are at the core of DL reasoners
- Tableaux calculi have been implemented successfully
- Refutation calculus based on disjunctive normal form
- We demonstrate it here at an example for $\mathcal{ALC}$ TBoxes
- For a full description and proofs see handbook article by Baader

Lit: F. Baader and W. Nutt. Basic description logics. In F. Baader et al., editors, The Description Logic Handbook, pages 43–95. Cambridge University Press, 2003.

Tableaux Example

- $\mathcal{ALC}$ tableau gives tests for satisfiability of ABox
- by checking whether obvious contradictions (clashes with complementary literals) are contained
- An ABox that is complete (no rules applicable anymore) and open (no clashes) describes a model
  - applies tableau rules to extend ABox

Rules

- Starts with an ABox $\mathcal{A}_0$ which is in negation normal form (NNF, $\neg$ in front of concept names)

- Apply rules to construct new ABoxes; indeterminism due to $\sqcup$ rule

Rule	Condition	$\rightsquigarrow$	Effect
$\rightsquigarrow_\sqcap$	$(C \sqcap D)(x) \in \mathcal{A}$	$\rightsquigarrow$	$\mathcal{A} \cup \{C(x), D(x)\}$
$\rightsquigarrow_\sqcup$	$(C \sqcup D)(x) \in \mathcal{A}$	$\rightsquigarrow$	$\mathcal{A} \cup \{C(x)\}$ **or** $\mathcal{A} \cup \{D(x)\}$
$\rightsquigarrow_\exists$	$(\exists r.C)(x) \in \mathcal{A}$	$\rightsquigarrow$	$\mathcal{A} \cup \{r(x,y), C(y)\}$ for fresh y
$\rightsquigarrow_\forall$	$(\forall r.C)(x), r(x,y) \in \mathcal{A}$	$\rightsquigarrow$	$\mathcal{A} \cup \{C(y)\}$

- Rules only applicable if they lead to an addition of assertion
- One obtains a tree with ABoxes (due to indeterminism)
- Within each ABox a tree-like structure is established (tree-model property)

Example

- Given: $\mathcal{T} = \{GoodStudent \equiv Smart \sqcap Studious\}$
- Subsumption test:
  $\mathcal{T} \models \exists knows.Smart \sqcap \exists knows.Studious \sqsubseteq \exists knows.GoodStudent$
- Reduction to ABox satisfiability:
  $\{\exists knows.Smart \sqcap \exists knows.Studious \sqcap \neg(\exists knows.GoodStudent)(a)\}$ satisfiable?
- Expansions of definition
  $\{\exists knows.Smart \sqcap \exists knows.Studious \sqcap \neg(\exists knows.(Smart \sqcap Studious))(a)\}$ satisfiable?
- Transform to NNF
  $\{\exists knows.Smart \sqcap \exists knows.Studious \sqcap \forall knows.(\neg Smart \sqcup \neg Studious)(a)\}$ satisfiable?

- GCIs can be transformed to definitions (i.e. axioms of the form $A \equiv C$ using additional symbols

Example

- Given: $\mathcal{T} = \{GoodStudent \equiv Smart \sqcap Studious\}$
- Subsumption test:
  $\mathcal{T} \models \exists knows.Smart \sqcap \exists knows.Studious \sqsubseteq \exists knows.GoodStudent$
- Reduction to ABox satisfiability:
  $\{\exists knows.Smart \sqcap \exists knows.Studious \sqcap \neg(\exists knows.GoodStudent)(a)\}$ satisfiable?
- Expansions of definition
  $\{\exists knows.Smart \sqcap \exists knows.Studious \sqcap \neg(\exists knows.(Smart \sqcap Studious))(a)\}$ satisfiable?
- Transform to NNF
  $\{\exists knows.Smart \sqcap \exists knows.Studious \sqcap \forall knows.(\neg Smart \sqcup \neg Studious)(a)\}$ satisfiable?

- GCIs can be transformed to definitions (i.e. axioms of the form $A \equiv C$ using additional symbols

Example (A Tableau Derivation)

- $\{\exists knows.Smart \sqcap \exists knows.Studious \sqcap \forall knows.(\neg Smart \sqcup \neg Studious)(a)\}$
- Abbreviation: $\{\exists r.A \sqcap \exists r.B \sqcap \forall r.(\neg A \sqcup \neg B)(a)\}$

$$\mathcal{A}_0 = \exists r.A \sqcap \exists r.B \sqcap \forall r.(\neg A \sqcup \neg B)(a)$$

$\rightsquigarrow_{\sqcap}$ (2 times)

$$\mathcal{A}_1 = \mathcal{A}_0 \cup \{(\exists r.A)(a),\ (\exists r.B)(a),\ (\forall r.(\neg A \sqcup \neg B))(a)\}$$

$\rightsquigarrow_{\exists}$ (2 times)

$$\mathcal{A}_2 = \mathcal{A}_1 \cup \{r(a,b), A(b), r(a,c), B(c)\}$$

$\rightsquigarrow_{\forall}$ (2 times)

$$\mathcal{A}_3 = \mathcal{A}_2 \cup \{(\neg A \sqcup \neg B)(b), (\neg A \sqcup \neg B)(c)\}$$

$\rightsquigarrow_{\sqcup}$ $\rightsquigarrow_{\sqcup}$

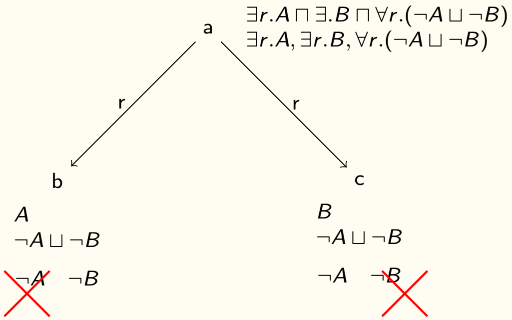
$$\mathcal{A}_{4.1} = \mathcal{A}_3 \cup \{(\neg A)(b)\} \qquad \mathcal{A}_{4.2} = \mathcal{A}_3 \cup \{(\neg B)(b)\}$$

$\rightsquigarrow_{\sqcup}$ $\rightsquigarrow_{\sqcup}$ $\rightsquigarrow_{\sqcup}$ $\rightsquigarrow_{\sqcup}$

$\mathcal{A}_{5.11} =$ $\mathcal{A}_{4.1} \cup \{\neg A(c)\}$	$\mathcal{A}_{5.12} =$ $\mathcal{A}_{4.1} \cup \{\neg B(c)\}$	$\mathcal{A}_{5.21} =$ $\mathcal{A}_{4.2} \cup \{\neg A(c)\}$	$\mathcal{A}_{5.22} =$ $\mathcal{A}_{4.2} \cup \{\neg B(c)\}$
clash	clash		clash

Example (The partial tree model in the ABoxes)

- $\{\exists knows.Smart \sqcap \exists knows.Studious \sqcap \forall knows.(\neg Smart \sqcup \neg Studious)(a)\}$
- Abbreviation: $\{\exists r.A \sqcap \exists r.B \sqcap \forall r.(\neg A \sqcup \neg B)(a)\}$



- Canonical tree model(s) can be directly read off:
  $\mathcal{I} = (\{a, b, c\}, \cdot^{\mathcal{I}})$ with
  $r^{\mathcal{I}} = \{(a, b), (a, c)\}$ $\quad A^{\mathcal{I}} = \{b\}$ $\quad B^{\mathcal{I}} = \{c\}$

Tableaux Calculus

- The tableau calculus for $\mathcal{ALC}$ is complete, correct, and terminates.
- Hence, the following properties hold

Theorem

- *$\mathcal{ALC}$ ABox satisfiability (concept satisfiability, subsumption...) is decidable*
- *$\mathcal{ALC}$ has the finite model property, i.e.*
  if an $\mathcal{ALC}$ ontology has a model, then it has a finite model.
- *$\mathcal{ALC}$ has the tree model property*

Solutions to Exercise 5 (10 Points)

Exercise 5.1 (4 Points)

Prove the folklore proposition that conjunctive queries are preserved under homomorphisms, i.e., show that if there is a homomorphism h from a DB instance $\mathfrak{T}$ to a DB instance $\mathfrak{T}'$, then for any CQ $\phi(\vec{x})$:

$$\{h(\vec{d}) \mid \vec{d} \in ans(\phi(\vec{x}), \mathfrak{T})\} \subseteq ans(\phi(\vec{x}), \mathfrak{T}')$$

Solution

- Let $\phi(\vec{x}) = \exists \vec{y} \bigwedge R_i(\vec{x}, \vec{y})$
- Let $\vec{d} \in ans(\phi(\vec{x}), \mathfrak{T})$.
- Hence there are $\vec{e}$ s.t. all $R_i(\vec{d}, \vec{e})$ are contained in $\mathfrak{T}$. Due to the homomorphism condition we have that all $R_i(h(\vec{d}), h(\vec{e}))$ are in $\mathfrak{T}'$. Hence $h(\vec{d}) \in ans(\phi(\vec{x}), \mathfrak{T}')$.

Exercise 5.2 (6 Points)

1. Prove that every finite graph has a core (2 points)
2. Prove that two cores of the same graph are isomorphic. (4 points)

Solution

1. Stepwise eliminate facts until no sub-instance can be embedded homomorphically into it. Will reach core after finite steps as graph is finite.
2. Take two cores of a graph $\mathfrak{G}1$ and $\mathfrak{G}2$. There is $h1 : \mathfrak{G} \xrightarrow{hom} \mathfrak{G}1$ and $h2 : \mathfrak{G} \xrightarrow{hom} \mathfrak{G}2$. The restriction $h1'$ of $h1$ to $\mathfrak{G}2$ must be a surjective homomorphism (otherwise the image of the restriction would be a proper subinstance into which $h1' \circ h2$ would give a homomorphic embedding of $\mathfrak{G}$. Similarly for $h2$. Hence $\mathfrak{G}1$ and $\mathfrak{G}2$ have surjective homomorphisms into each other and so they are isomorphic.

Next lecture on December 7th 2016 is cancelled!