TECHNISCHE UNIVERSITÄT DRESDEN

Computational Logic Group

Hannes Strass (based on slides by Bernardo Cuenca Grau, Ian Horrocks, Przemysław Wałęga)
Faculty of Computer Science, Institute of Artificial Intelligence, Computational Logic Group

Description Logics – Reasoning with Data

Lecture 6, 21st Nov 2022 // Foundations of Knowledge Representation, WS 2024/25

Recap

- For description logic knowledge bases, there are various relevant reasoning problems.
- All can be reduced to knowledge base (in)satisfiability.
- The basic description logic $\mathcal{ALC}$ can be extended in various ways:
  - Inverse Roles $\mathcal{I}$
  - (Qualified) Number Restrictions $(\mathcal{Q})\ \mathcal{N}$
  - Nominals $\mathcal{O}$
  - Role Hierarchies $\mathcal{H}$
  - Transitive Roles $\mathcal{ALC} \rightsquigarrow \mathcal{S},\ \cdot_{R^+}$
- Description Logics have close connections with propositional modal logic . . .
- . . . and with the two-variable fragments of first-order logic (with counting quantifiers)

Reasoning with Data

So far we have focused on terminological reasoning

- TBoxes represent general, conceptual domain knowledge
- Terminological reasoning is key to design error-free TBoxes

New Scenario: Ontology-based data access (OBDA)

- We have built an (error-free) TBox for our domain
- We want to populate TBox with data (add an ABox)
  - ABox & TBox should be compatible (no inconsistencies)
- Then, we can query the data
  - TBox provides vocabulary for queries
  - Answers reflect both TBox knowledge and ABox data

Compatibility of Data and Knowledge

The ABox data should be compatible with the TBox knowledge

$$\mathcal{T} = \{\text{GradSt} \sqcap \text{UnderGradSt} \sqsubseteq \bot\}$$
$$\mathcal{A} = \{\text{John} : \text{GradSt}, \text{John} : \text{UnderGradSt}\}$$

- Nothing wrong with the TBox
- Nothing wrong with the ABox
- There is an obvious error when putting them together

To detect these situations we use the following problem:

> **Knowledge Base satisfiability:**
> An instance is knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$.
> The answer is true iff a model $\mathcal{I} \models \mathcal{K}$ exists.

In a FOL setting, $\mathcal{K}$ is satisfiable if and only if $\pi(\mathcal{K})$ is satisfiable.

Tableau Algorithm for KB Consistency

Tableau-based knowledge base consistency algorithm:

- Input: Knowledge Base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$
- Output: true iff $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ is consistent

1. Start with input ABox $\mathcal{A}$
2. Apply expansion rules until completion or clash
3. Blocking only involves individuals not occurring in $\mathcal{A}$

Exploit forest-model property: construct forest-shaped ABox
root (ABox) individuals can be arbitrarily connected
tree individuals (introduced by $\exists$-rule) form trees

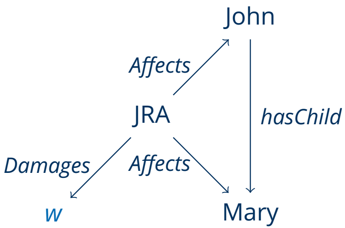
Typically, we are interested in tableau algorithms that are sound and complete w.r.t. the model theory, whence the terms satisfiable (model-theoretic) and consistent (proof-theoretic) coincide.

Tableau Example (Simplified)

$$
\begin{aligned}
(\mathsf{JRA}, \mathsf{John}) &: \mathit{Affects} \\
\mathsf{JRA} &: \mathsf{JuvArth} \\
(\mathsf{JRA}, \mathsf{Mary}) &: \mathit{Affects} \\
(\mathsf{John}, \mathsf{Mary}) &: \mathit{hasChild}
\end{aligned}
$$

$$
\begin{aligned}
\mathsf{JuvDis} &\sqsubseteq \exists \mathit{Affects}.\mathsf{Child} \sqcap \forall \mathit{Affects}.\mathsf{Child} \\
\exists \mathit{hasChild}.\top &\sqsubseteq \mathsf{Adult} \\
\mathsf{Adult} &\sqsubseteq \neg \mathsf{Child} \\
\mathsf{Arth} &\sqsubseteq \exists \mathit{Damages}.\mathsf{Joint} \\
\mathsf{JuvArth} &\sqsubseteq \mathsf{Arth} \sqcap \mathsf{JuvDis}
\end{aligned}
$$

Tableau expansion (simplified):



$$
\begin{aligned}
\mathsf{con}_{\mathcal{A}}(\mathsf{JRA}) &= \{\mathsf{JuvArth}, \mathsf{Arth}, \mathsf{JuvDis}, \exists \mathit{Damages}.\mathsf{Joint}, \\
&\quad \exists \mathit{Affects}.\mathsf{Child}, \forall \mathit{Affects}.\mathsf{Child}\} \\
\mathsf{con}_{\mathcal{A}}(\mathsf{John}) &= \{\mathsf{Child}, \mathsf{Adult}, \neg \mathsf{Child}\} \\
\mathsf{con}_{\mathcal{A}}(\mathsf{Mary}) &= \{\mathsf{Child}\} \\
\mathsf{con}_{\mathcal{A}}(w) &= \{\mathsf{Joint}\}
\end{aligned}
$$

Querying the Data

It does not make sense to query an inconsistent $\mathcal{K}$ (previous example).

- An inconsistent ($\triangleq$ unsatisfiable) $\mathcal{K}$ entails all formulas.
- We (typically) fix inconsistencies before we start asking queries.

Once we have determined that $\mathcal{K}$ is consistent, we want to query the data:

- Which children are affected by a juvenile arthritis?
- Which drugs are used to treat JRA?
- Who is affected by an arthritis and is allergic to steroids?

Similar to the types of queries one would pose to a database.

```
SELECT Child.cname
   FROM Child, Affects, JuvArth
   WHERE Child.cname = Affects.cname AND
         Affects.dname = JuvArth.dname
```

Querying the Data: Simple Queries (1)

The basic data queries ask for all the instances of a concept:

$q_1(x) = \text{Child}(x)$ — Set of children?

$q_2(x) = (\text{Dis} \sqcap \exists \textit{Damages}.\text{Joint})(x)$ — Set of diseases affecting a joint?

How to (naively) answer these queries? Try each individual name.

ABox $\mathcal{A}$

$(\text{JRA}, \text{John}) : \textit{Affects}$

$\text{JRA} : \text{JuvArth}$

$(\text{JRA}, \text{Mary}) : \textit{Affects}$

TBox $\mathcal{T}$ $\quad (\mathcal{K} = (\mathcal{T}, \mathcal{A}))$

$\text{JuvDis} \sqsubseteq \exists \textit{Affects}.\text{Child} \sqcap \forall \textit{Affects}.\text{Child}$

$\text{Adult} \sqsubseteq \neg \text{Child}$

$\text{Arth} \sqsubseteq \exists \textit{Damages}.\text{Joint}$

$\text{JuvArth} \sqsubseteq \text{Arth} \sqcap \text{JuvDis}$

$\mathcal{K} \models \text{JRA} : \text{Child}$? *No.* JRA is not an answer to q_1

$\mathcal{K} \models \text{John} : \text{Child}$? *Yes!* John is an answer to q_1

$\mathcal{K} \models \text{Mary} : \text{Child}$? *Yes!* Mary is an answer to q_1

Querying the Data: Simple Queries (2)

So, we are interested in the following decision problem:

Concept Instance Checking:
Given individual name a, concept C and KB $\mathcal{K}$,
an instance is a triple $\langle a, C, \mathcal{K} \rangle$.
The answer is true iff $\mathcal{K} \models a : C$

In $\mathcal{ALC}$ (and extensions) this problem is reducible to KB satisfiability:

$(\mathcal{T}, \mathcal{A}) \models a : C \quad$ iff $\quad (\mathcal{T}, \mathcal{A} \cup \{a : \neg C\})$ satisfiable

Note that we can assume, w.l.o.g., that *C* is a concept name:

$$(\mathcal{T}, \mathcal{A}) \models a : C \qquad \text{iff} \qquad (\mathcal{T} \cup \{X \equiv C\}, \mathcal{A}) \models a : X$$

where X is a concept name that does not occur in $\mathcal{T}$ or $\mathcal{A}$.

Querying the Data: Simple Queries (3)

What about instances of a role:

$$q_2(x,y) = \mathit{hasChild}(x,y) \quad \text{Set of parent-child tuples?}$$

How to (naively) answer these queries? Try each pair of individuals!

ABox $\mathcal{A}$

$$\begin{aligned} \mathsf{JRA} &: \mathsf{JuvArth} \\ (\mathsf{JRA}, \mathsf{Mary}) &: \mathit{Affects} \\ (\mathsf{John}, \mathsf{Mary}) &: \mathit{hasChild} \end{aligned}$$

TBox $\mathcal{T}$ $(\mathcal{K} = (\mathcal{T}, \mathcal{A}))$

$$\begin{aligned} \mathsf{JuvDis} &\sqsubseteq \exists \mathit{Affects}.\mathsf{Child} \sqcap \forall \mathit{Affects}.\mathsf{Child} \\ \mathsf{Adult} &\sqsubseteq \neg \mathsf{Child} \\ \mathsf{Arth} &\sqsubseteq \exists \mathit{Damages}.\mathsf{Joint} \\ \mathsf{JuvArth} &\sqsubseteq \mathsf{Arth} \sqcap \mathsf{JuvDis} \end{aligned}$$

$\mathcal{K} \models (\mathsf{John}, \mathsf{John}) : \mathit{hasChild}$? *No.* (John, John) is not an answer to q_2

$\mathcal{K} \models (\mathsf{John}, \mathsf{Mary}) : \mathit{hasChild}$? *Yes!* (John, Mary) is an answer to q_2

$\mathcal{K} \models (\mathsf{John}, \mathsf{JRA}) : \mathit{hasChild}$? *No.* (John, John) is not an answer to q_2

...

Querying the Data: Simple Queries (4)

So, we are interested in the following decision problem:

Role Instance Checking:
Given a pair of individual names (a, b), role R and KB $\mathcal{K}$, an instance is a triple $\langle(\mathrm{a},\mathrm{b}), R, \mathcal{K}\rangle$.
The answer is true iff $\mathcal{K} \models (\mathrm{a},\mathrm{b}) : R$

Can this problem be reduced to knowledge base consistency?

$$(\mathcal{T},\mathcal{A}) \models (\mathrm{a},\mathrm{b}) : R \quad \text{iff} \quad (\mathcal{T}, \mathcal{A} \cup \{\mathrm{a} : \forall R.\mathrm{X}, \mathrm{b} : \neg\mathrm{X}\}) \text{ is inconsistent}$$

where X is a concept name that does not occur in $\mathcal{T}$ or $\mathcal{A}$.

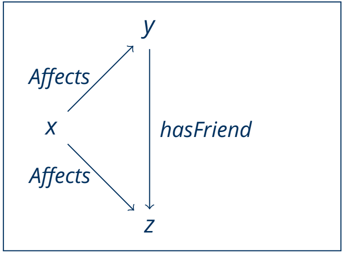
Limitations of Concept-based Queries

Some natural queries cannot be expressed using a concept:

$$q(y) = \exists x \exists z (\mathit{Affects}(x, y) \land \mathit{Affects}(x, z) \land \mathit{hasFriend}(y, z))$$

Set of people (y) affected by the same disease as a friend?

Query Graph:



We can only represent tree-like queries as concepts

Related to the tree model property of DLs

We need a more expressive query language . . .

Conjunctive Queries

The language of conjunctive queries:

- Generalises concept-based queries in a natural way
  - arbitrarily-shaped queries vs. tree-like queries
- Widely used as a query language in databases
  - Corresponds to Select-Project-Join fragment of relational algebra
  - Fragment of relational calculus using only $\exists$ and $\land$
- Implemented in most DBMS

We next study the problem of CQ answering over DL knowledge bases

We will not study the problem of answering FOL queries over DL KBs

$\rightsquigarrow$ Corresponds to general relational calculus queries.

$\rightsquigarrow$ Leads to an undecidable decision problem.

Conjunctive Queries – Definition

Conjunctive query

Let **V** be a set of **variables**.
A **term** t is a variable from **V** or an individual name from **I**.

A **conjunctive query** (CQ) q has the form $\exists x_1 \cdots \exists x_k (\alpha_1 \wedge \cdots \wedge \alpha_n)$ where:

- $k \geq 0$, $n \geq 1$, $x_1, \ldots, x_k \in \mathbf{V}$;
- each α_i is a concept atom $A(t)$ or a role atom $r(t, t')$, that is, $A \in \mathbf{C}$, $r \in \mathbf{R}$, and t, t' are terms;
- $x_1, \ldots, x_k$ are called **quantified variables**; all other variables in q are called **answer variables**;
- the **arity** of q is the number of answer variables;
- q is called **Boolean** if it has arity zero.

To indicate that the answer variables in a CQ q are $\vec{x}$, we often write $q(\vec{x})$ instead of just q.

Example Conjunctive Queries

1. Return all pairs of individual names (a, b) such that a is a professor who supervises student b:

$$q_1(x_1, x_2) = \text{Professor}(\underline{x_1}) \wedge \text{supervises}(\underline{x_1}, \underline{x_2}) \wedge \text{Student}(\underline{x_2}).$$

2. Return all individual names a such that a is a student supervised by some professor:

$$q_2(x) = \exists y\, (\text{Professor}(y) \wedge \text{supervises}(y, \underline{x}) \wedge \text{Student}(\underline{x})).$$

3. Return all pairs of students supervised by the same professor:

$$q_3(x_1, x_2) = \exists y\, (\text{Professor}(y) \wedge \text{supervises}(y, \underline{x_1}) \wedge \text{supervises}(y, \underline{x_2}) \wedge \text{Student}(\underline{x_1}) \wedge \text{Student}(\underline{x_2})).$$

4. Return all students supervised by professor smith (an individual name):

$$q_4(x) = \text{supervises}(\text{smith}, \underline{x}) \wedge \text{Student}(\underline{x}).$$

Answers on an Interpretation

We first define query answers on a given interpretation $\mathcal{I}$.

Definition

Let q be a conjunctive query and $\mathcal{I}$ an interpretation.
We use $\mathrm{term}(q)$ to denote the terms in q.

A **match of q in $\mathcal{I}$** is a mapping $\pi : \mathrm{term}(q) \to \Delta^{\mathcal{I}}$ such that

- $\pi(a) = a^{\mathcal{I}}$ for all $a \in \mathrm{term}(q) \cap \mathbf{I}$,
- $\pi(t) \in A^{\mathcal{I}}$ for all concept atoms $A(t)$ in q, and
- $(\pi(t_1), \pi(t_2)) \in r^{\mathcal{I}}$ for all role atoms $r(t_1, t_2)$ in q.

Let $\vec{x} = x_1, \ldots, x_k$ be the answer variables in q and $\vec{a} = a_1, \ldots, a_k$ be individual names from $\mathbf{I}$.
We call the match π of q in $\mathcal{I}$ an **$\vec{a}$-match** if $\pi(x_i) = a_i^{\mathcal{I}}$ for $1 \leq i \leq k$.

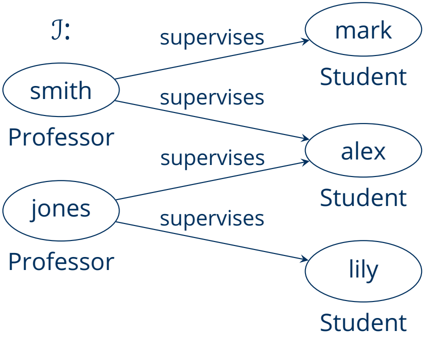
We say that $\vec{a}$ is an **answer to q on $\mathcal{I}$** if there is an $\vec{a}$-match π of q in $\mathcal{I}$.

We use $\mathrm{ans}(q, \mathcal{I})$ to denote the set of all answers to q on $\mathcal{I}$.

Answers on Interpretation $\mathcal{J}$ (1)

$q_2(x)$:

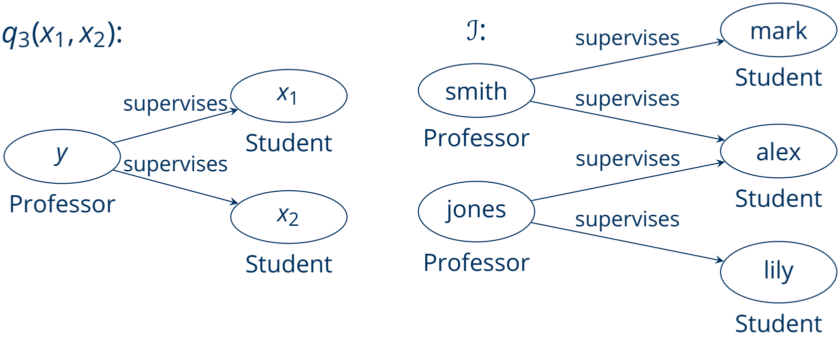
$\mathcal{J}$:



$$q_2(x) = \exists y(\text{Professor}(y) \land \text{supervises}(y, \underline{x}) \land \text{Student}(\underline{x}))$$

There are 3 answers to $q_2(x)$ on $\mathcal{J}$: mark, alex, and lily.
Note that a match is a homomorphism from the query to the interpretation (both viewed as a graphs).

Answers on Interpretation $\mathcal{J}$ (2)



$$q_3(x_1, x_2) = \exists y\, (\text{Professor}(y) \land \text{supervises}(y, \underline{x_1}) \land \text{supervises}(y, \underline{x_2}) \land \text{Student}(\underline{x_1}) \land \text{Student}(\underline{x_2})).$$

There are 7 answers to $q_3(x_1, x_2)$ on $\mathcal{J}$, including (mark, alex), (alex, lily), (lily, alex) and (mark, mark). Note that a match need not be injective.

Certain Answers

Usually we are interested in answers on a KB, which may have many models.

In this case, so-called certain answers provide a natural semantics.

Definition

Let q be a CQ and $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a KB.

We say that $\vec{a}$ is a **certain answer to** q **on** $\mathcal{K}$ if

- all individual names from $\vec{a}$ occur in $\mathcal{A}$, and
- $\vec{a} \in \mathrm{ans}(q, \mathcal{I})$ for every model $\mathcal{I}$ of $\mathcal{K}$.

We use $\mathrm{cert}(q, \mathcal{K})$ to denote the set of all certain answers to q on $\mathcal{K}$:

$$\mathrm{cert}(q, \mathcal{K}) = \bigcap_{\mathcal{I} \models \mathcal{K}} \mathrm{ans}(q, \mathcal{I})$$

Certain Answers: Examples

Consider the $\mathcal{ALCI}$ KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$:

$$\mathcal{T} = \{\text{Student} \sqsubseteq \exists \mathit{supervises}^{-}.\text{Professor}\},$$
$$\mathcal{A} = \{\text{smith} : \text{Professor}, \text{mark} : \text{Student}, \text{alex} : \text{Student}, \text{lily} : \text{Student}, (\text{smith}, \text{mark}) : \mathit{supervises}, (\text{smith}, \text{alex}) : \mathit{supervises}\}.$$

- $q_4(x) = \text{supervises}(\text{smith}, \underline{x}) \land \text{Student}(\underline{x})$; $\text{cert}(q_4, \mathcal{K}) = \{\text{mark}, \text{alex}\}$: there are models of $\mathcal{K}$ in which smith supervises other students, but only mark and alex are supervised by smith in *all* models.
- $q_2(x) = \exists y(\text{Professor}(y) \land \text{supervises}(y, \underline{x}) \land \text{Student}(\underline{x}))$; $\text{cert}(q_2, \mathcal{K}) = \{\text{mark}, \text{alex}, \text{lily}\}$: note that lily is included because she is a student and thus the TBox enforces that in every model of $\mathcal{K}$ she has a supervisor who is a professor.
- $q_1(x_1, x_2) = \text{Professor}(\underline{x_1}) \land \text{supervises}(\underline{x_1}, \underline{x_2}) \land \text{Student}(\underline{x_2})$; $\text{cert}(q_1, \mathcal{K}) = \{(\text{smith}, \text{mark}), (\text{smith}, \text{alex})\}$: lily always has a supervisor, but there is no supervisor (known by name) on which all models agree.

Boolean Conjunctive Query Answering

(Arbitrary) CQ answering reduces to Boolean CQ answering.

Given query q of arity n and $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ in which m individual names occur:

- Iterate through m^n tuples of arity n.
- For each tuple $\vec{a} = (a_1, \ldots, a_n)$ create a Boolean query $q_{\vec{a}}$ by replacing the ith answer variable with a_i, for all $1 \leq i \leq n$.
- $\vec{a} \in \mathrm{cert}(q, \mathcal{K})$ iff $\mathcal{K} \models q_{\vec{a}}$.

> **Boolean Conjunctive Query Entailment:**
> An instance is a pair $\langle \mathcal{K}, q \rangle$
> with $\mathcal{K}$ a KB and q a Boolean CQ.
> The answer is true iff $\mathcal{I} \models q$ for each $\mathcal{I} \models \mathcal{K}$.

This problem is not trivially reducible to knowledge base satisfiability.

It is ExpTime-complete for $\mathcal{ALC}$, the same as satisfiability.
(A proof is beyond the scope of this course.)

Boolean Conjunctive Query Answering

Many types of query can be reduced to KB satisfiability:

- Concept and role instance queries, e.g., $q() = C(a)$ and $q() = r(a, b)$.
- Fully ground queries, e.g., $q() = C(a) \wedge D(b) \wedge r(a, b)$ (idea: check each atom independently).
- Forest shaped queries, e.g., $q() = \exists x(C(a) \wedge D(x) \wedge r(a, x))$ (idea: roll up the tree parts of the query).

Reduction may or may not be possible in general (possible for $\mathcal{SHIQ}$; open problem for $\mathcal{SHOIQ}$).

Conjunctive Query Answering (1)

How to interpret the answer to a Boolean Query? $(\mathcal{K} = (\mathcal{T}, \mathcal{A}))$

ABox $\mathcal{A}$:

$$\begin{aligned}(\mathsf{JRA}, \mathsf{John}) &: \mathit{Affects} \\ \mathsf{JRA} &: \mathsf{JuvArth} \\ (\mathsf{JRA}, \mathsf{Mary}) &: \mathit{Affects}\end{aligned}$$

TBox $\mathcal{T}$:

$$\begin{aligned}\mathsf{JuvDis} &\sqsubseteq \exists \mathit{Affects}.\mathsf{Child} \sqcap \forall \mathit{Affects}.\mathsf{Child} \\ \mathsf{Adult} &\sqsubseteq \neg \mathsf{Child} \\ \mathsf{Arth} &\sqsubseteq \exists \mathit{Damages}.\mathsf{Joint} \\ \mathsf{JuvArth} &\sqsubseteq \mathsf{Arth} \sqcap \mathsf{JuvDis}\end{aligned}$$

$$\begin{aligned}q_1 &= \mathit{Affects}(\mathsf{JRA}, \mathsf{Mary}) \\ q_2 &= \mathsf{Child}(\mathsf{Mary}) \\ q_3 &= \mathsf{Adult}(\mathsf{Mary}) \\ q_4 &= \exists y (\mathit{Damages}(\mathsf{JRA}, y) \land \mathsf{Organ}(y))\end{aligned}$$

$\mathcal{A} \models q_1$	Yes
$\mathcal{A} \not\models q_2, \mathcal{A} \not\models \neg q_2$	???
$\mathcal{K} \models q_2$	Yes
$\mathcal{A} \not\models q_3, \mathcal{A} \not\models \neg q_3$	???
$\mathcal{K} \models \neg q_3$	No
$\mathcal{A} \not\models q_4, \mathcal{A} \not\models \neg q_4$	???
$\mathcal{K} \not\models q_4, \mathcal{K} \not\models \neg q_4$	???

Conjunctive Query Answering (2)

$\mathcal{A}$ is seen as a FOL knowledge base, but $\mathcal{D}$ is seen as a FOL model:

ABox $\mathcal{A}$

$$\begin{aligned}(\mathrm{JRA}, \mathrm{John}) &: \mathit{Affects} \\ \mathrm{JRA} &: \mathrm{JuvArth} \\ (\mathrm{JRA}, \mathrm{Mary}) &: \mathit{Affects}\end{aligned}$$

Database $\mathcal{D}$

Affects	
JRA	John
JRA	Mary

JuvArthritis
JRA

$$\begin{aligned} q_1 &= \mathit{Affects}(\mathrm{JRA}, \mathrm{Mary}) \\ q_2 &= \mathrm{Child}(\mathrm{Mary}) \\ q_3 &= \mathrm{Adult}(\mathrm{Mary}) \\ q_4 &= \exists y(\mathit{Damages}(\mathrm{JRA}, y) \wedge \mathrm{Organ}(y)) \end{aligned}$$

$\mathcal{A} \models q_1$	Yes
$\mathcal{D} \models q_1$	Yes
$\mathcal{A} \not\models q_2, \mathcal{A} \not\models \neg q_2$	???
$\mathcal{D} \not\models q_2$	*No*
$\mathcal{A} \not\models q_3, \mathcal{A} \not\models \neg q_3$	???
$\mathcal{D} \not\models q_3$	*No*
$\mathcal{A} \not\models q_4, \mathcal{A} \not\models \neg q_4$	???
$\mathcal{D} \not\models q_4$	*No*

Ontologies vs. Database Systems

Conceptual DB-Schema:

- Typically formulated as an ER or UML diagram (used in DB design)
- Schema leads to a set of FOL-based constraints
- Constraints are used to check conformance of the data
- Constraints are disregarded for query answering

$\rightsquigarrow$ In databases, query answering is a FOL **model checking** problem.

Description Logic TBoxes:

- Formulated in a Description Logic (fragment of FOL)
- TBox axioms are used to check conformance of the data
    The way this is done differs from DBs
- TBox axioms participate in query answering

$\rightsquigarrow$ In description logics, query answering is a FOL **entailment** problem.

KB Consistency: Practicality Issues

- Addition of ABox may greatly exacerbate practicality problems
  - No obvious limit to size of data – could be millions or even billions of individuals
  - Tableau algorithm applied to whole ABox
- Optimisations can ameliorate but not eliminate the problem
- Can exploit decomposition of an ABox:
  - $\mathcal{A}$ can be decomposed into a set of disjoint connected components $\{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$ such that:

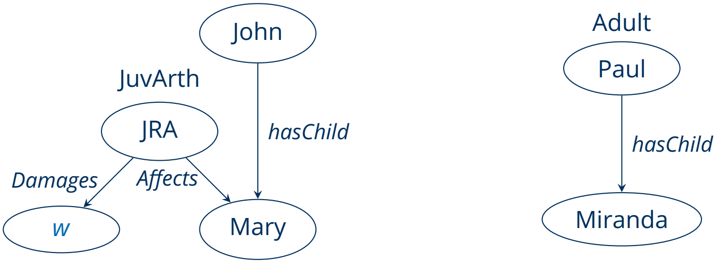
$$\mathcal{A} = \mathcal{A}_1 \cup \ldots \cup \mathcal{A}_n$$
$$\forall_{1 \leq i < j \leq n} \, \mathsf{ind}(\mathcal{A}_i) \cap \mathsf{ind}(\mathcal{A}_j) = \emptyset$$

    where $\mathsf{ind}(\mathcal{A}_i)$ is the set of individuals (constants) occurring in $\mathcal{A}_i$
- An $\mathcal{ALC}$ KB $(\mathcal{T}, \mathcal{A})$ is consistent iff $(\mathcal{T}, \mathcal{A}_i)$ is consistent for each $\mathcal{A}_i$ in a decomposition $\{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$ of $\mathcal{A}$

ABox Decomposition: Example

$$
\begin{aligned}
\mathsf{JRA} &: \mathsf{JuvArth} \\
(\mathsf{JRA}, \mathsf{Mary}) &: \mathit{Affects} \\
(\mathsf{John}, \mathsf{Mary}) &: \mathit{hasChild} \\
(\mathsf{Paul}, \mathsf{Miranda}) &: \mathit{hasChild} \\
\mathsf{Paul} &: \mathsf{Adult}
\end{aligned}
\qquad
\begin{aligned}
\mathsf{JuvDis} &\sqsubseteq \exists \mathit{Affects}.\mathsf{Child} \sqcap \forall \mathit{Affects}.\mathsf{Child} \\
\exists \mathit{hasChild}.\top &\sqsubseteq \mathsf{Adult} \\
\mathsf{Adult} &\sqsubseteq \neg \mathsf{Child} \\
\mathsf{Arth} &\sqsubseteq \exists \mathit{Damages}.\mathsf{Joint} \\
\mathsf{JuvArth} &\sqsubseteq \mathsf{Arth} \sqcap \mathsf{JuvDis}
\end{aligned}
$$

Perform separate consistency tests on the disjoint connected components:



Query Answering: Practicality Issues

- Recall our example query:

$$q(y) = \exists x \exists z (\mathit{Affects}(x,y) \land \mathit{Affects}(x,z) \land \mathit{hasFriend}(y,z))$$

- To answer this query we have to:
  - check for each individual a occurring in $\mathcal{A}$ if $(\mathcal{T},\mathcal{A}) \models q_{[y/a]}$, where $q_{[y/a]}$ is the Boolean CQ

$$q() = \exists x \exists z (\mathit{Affects}(x,a) \land \mathit{Affects}(x,z) \land \mathit{hasFriend}(a,z))$$

  - checking $(\mathcal{T},\mathcal{A}) \models q_{[y/a]}$ involves performing (possibly many) consistency tests;
  - each test could be very costly.
- And what if we change the query to

$$q(x,y,z) = \mathit{Affects}(x,y) \land \mathit{Affects}(x,z) \land \mathit{hasFriend}(y,z)?$$

- In general, there are m^n "candidate" answer tuples, where m is the number of individuals occurring in $\mathcal{A}$ and n the arity of the query.

Optimised Query Answering

Many optimisations are possible, for example:

- Exploit the fact that we cannot entail ABox roles in $\mathcal{ALC}$, that is:

$$(\mathcal{T}, \mathcal{A}) \models R(a, b) \text{ iff } R(a, b) \in \mathcal{A}$$

- Only check candidate tuples with relevant relational structure
- So for

$$q(y, z) = \exists x\, (\mathsf{JuvArth}(x) \land \mathit{Affects}(x, y) \land \mathit{hasFriend}(y, z))$$

  only check tuples (a, b) such that

$$\mathit{hasFriend}(a, b) \in \mathcal{A}$$

  and for these we only need to check the Boolean CQ:

$$\exists x\, (\mathsf{JuvArth}(x) \land \mathit{Affects}(x, a) \land \mathit{Affects}(x, b))$$

Conflicting Requirements

Ontology-based data access applications require:

1. Very expressive ontology languages

   As large fragment of FOL as possible

2. Powerful query languages

   As large fragment of SQL as possible

3. Efficient query answering algorithms

   Low complexity, easy to optimise

The requirements are in conflict!

$\rightsquigarrow$ We need to make compromises.

Conclusion

- DL KB consistency can be decided using tableau algorithms
  $\rightsquigarrow$ Idea: Make implicit inconsistencies explicit/construct model
- Query answering for DL KBs is understood as FOL *entailment*
- Conjunctive Queries (CQs) constitute natural query language
- CQs induce answers on a single interpretation, and *certain answers* on a KB
- Boolean CQ Entailment is not trivially reducible to KB consistency
- In contrast, CQ Entailment in databases is understood as FOL *model checking*