Semantic Technology Tutorial

Part 4: Reasoning

Why Ontology Reasoning?

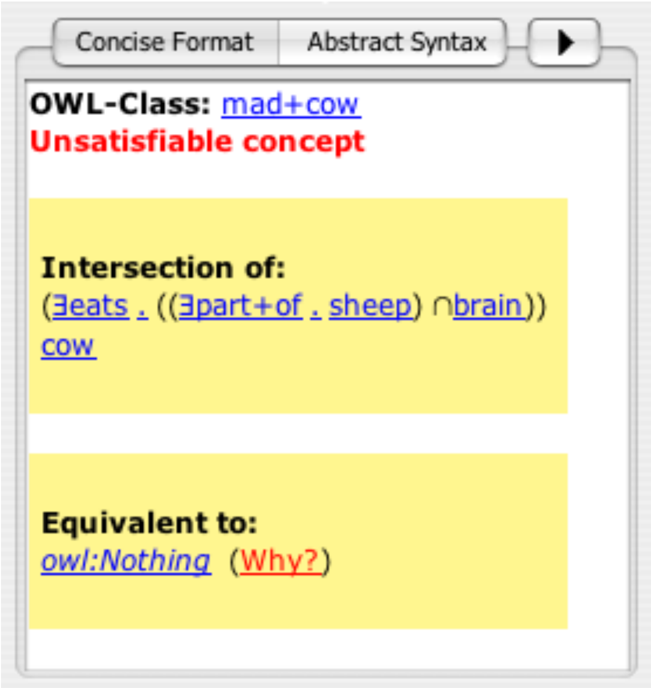
- Support for developing & maintaining ontologies
  - Known to be difficult/costly/time-consuming
  - Can be a major barrier to uptake of semantic technologies
- Fundamental service provided by semantic systems
  - Query answering over data, e.g.
    - For semantic data integration
    - For compliance verification and reporting
  - Schema queries, e.g.
    - For selecting components from large inventory
    - For identifying relevant advice based on customer profile
  - Recall that SPARQL allows for both schema and data queries, and even combined schema/data queries

Ontology Engineering

- Developing and maintaining quality ontologies is ***hard***

Ontology Engineering

- Developing and maintaining quality ontologies is ***hard***
- Reasoners allow domain experts to check if, e.g.:
  – classes are consistent (no “obvious” errors)



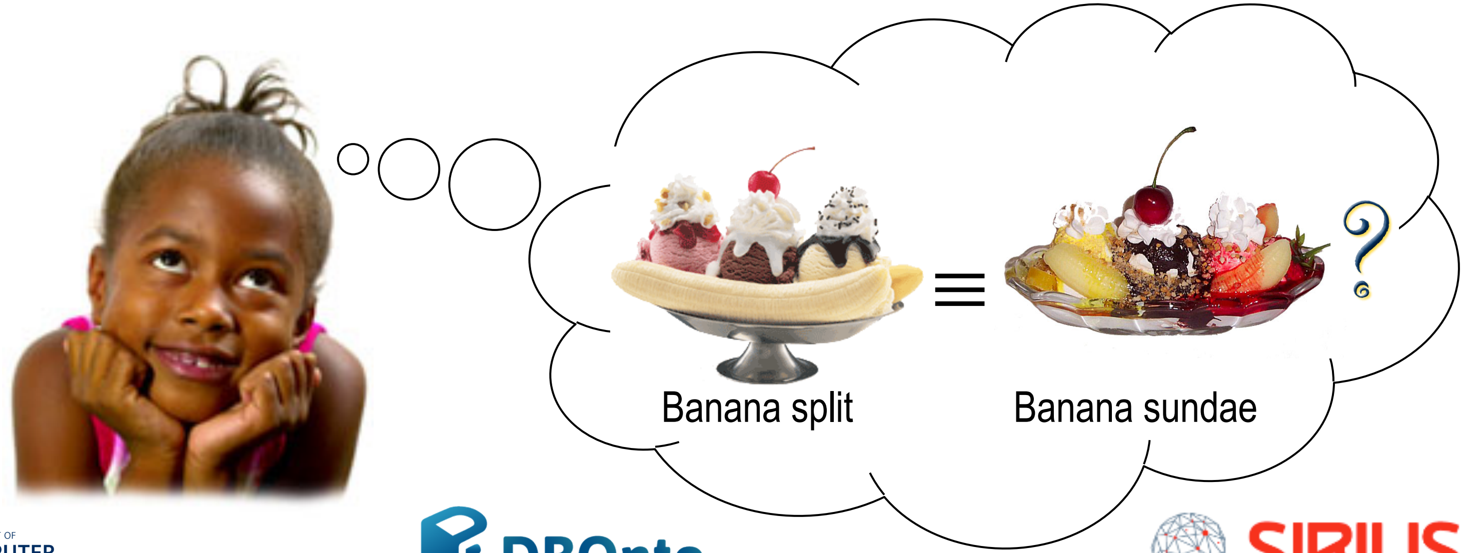
COMPUTER SCIENCE
DBOnto
SIRIUS

Ontology Engineering

- Developing and maintaining quality ontologies is ***hard***
- Reasoners allow domain experts to check if, e.g.:
  - classes are consistent (no "obvious" errors)
  - expected subsumptions hold (consistent with intuitions)

Ontology Engineering

- Developing and maintaining quality ontologies is ***hard***
- Reasoners allow domain experts to check if, e.g.:
  - classes are consistent (no “obvious” errors)
  - expected subsumptions hold (consistent with intuitions)
  - unexpected equivalences hold (unintended synonyms)



Ontology Engineering

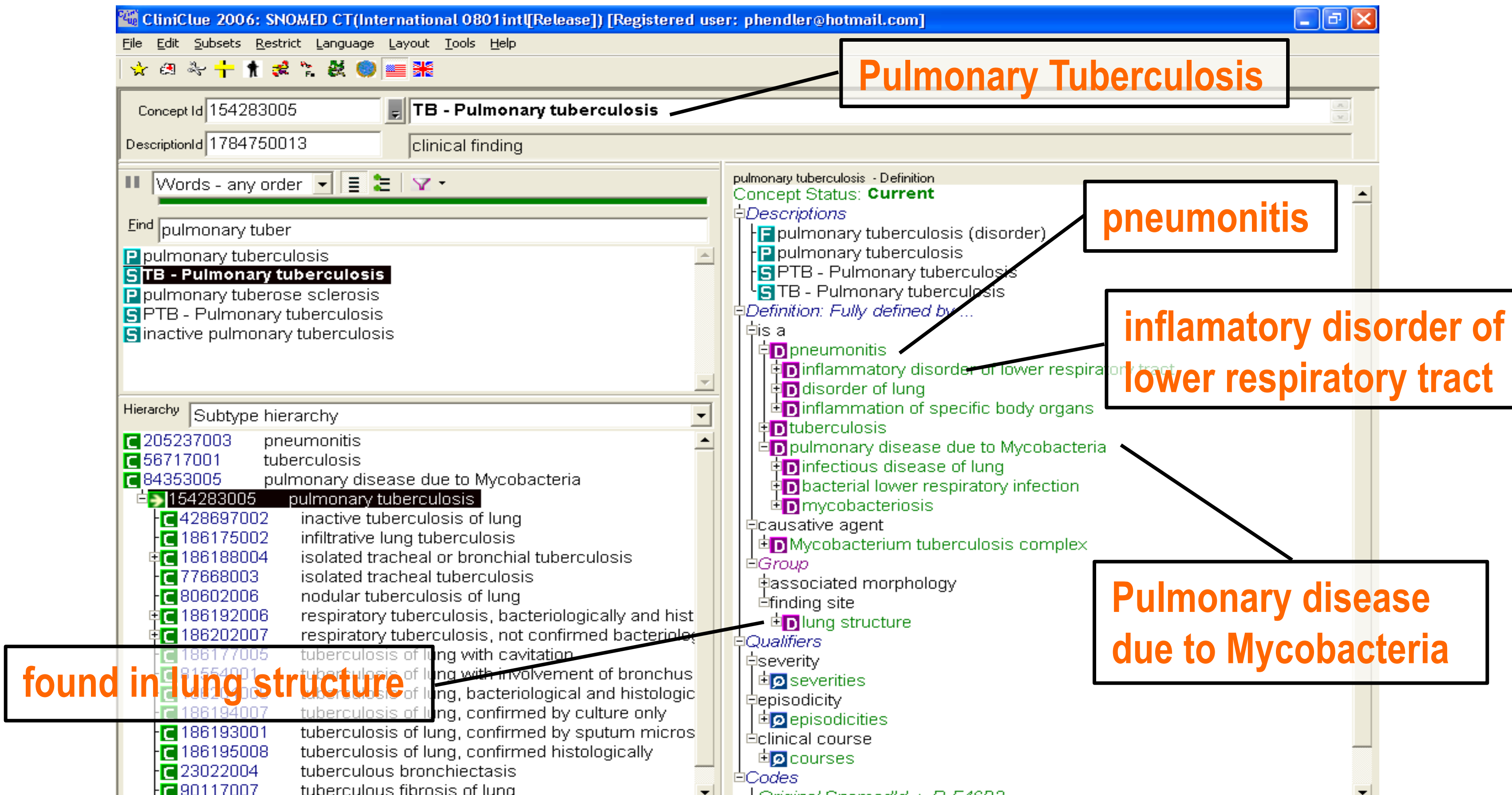
- Developing and maintaining quality ontologies is ***hard***
- Reasoners allow domain experts to check if, e.g.:
  - classes are consistent (no “obvious” errors)
  - expected subsumptions hold (consistent with intuitions)
  - unexpected equivalences hold (unintended synonyms)
- Reasoning also the basis for advanced tools, e.g.:
  - Ontology integration/reuse
  - Ontology module extraction
  - Explanation of (unexpected) inferences
  - …

Ontology Engineering: Case Study

SNOMED is **BIG** – over **400,000 concepts**

Ontology Engineering: Case Study

SNOMED is **BIG** – over **400,000 concepts**



Ontology Engineering: Case Study

- **Kaiser Permanente** extending SNOMED to express, e.g.:
  - *non-viral pneumonia* (negation)
  - *infectious pneumonia* is caused by a *virus* or a *bacterium* (disjunction)
  - *double pneumonia* occurs in two *lungs* (cardinalities)
- This is easy in **SNOMED-OWL**
  - but reasoner failed to find expected subsumptions, e.g., that *bacterial pneumonia* is a kind of *non-viral pneumonia*
- Ontology highly **under-constrained**: need to add disjointness axioms (at least)
  - *virus* and *bacterium* must be disjoint

Ontology Engineering: Case Study

- Adding disjointness led to **surprising results**
  - many classes become inconsistent, e.g., *percutanious embolization of hepatic artery using fluoroscopy guidance*
- Cause of **inconsistencies** identified as class *groin*
  - *groin* asserted to be subclass of both *abdomen* and *leg*
  - *abdomen and leg* are disjoint
  - modelling of *groin* (and other similar “junction” regions) identified as incorrect

Ontology Engineering: Case Study

- Correct modelling of groin is quite complex, e.g.:
  - groin has a part that is part of the abdomen, and has a part that is part of the leg (***inverse properties***)

$$\mathsf{Groin} \sqsubseteq \exists \mathsf{hasPart}.(\exists \mathsf{isPartOf}.\mathsf{Abdomen})$$
$$\mathsf{Groin} \sqsubseteq \exists \mathsf{hasPart}.(\exists \mathsf{isPartOf}.\mathsf{Leg})$$
$$\mathsf{hasPart} \equiv \mathsf{isPartOf}^{-}$$

  - all parts of the groin are part of the abdomen or the leg (**disjunction**)

$$\mathsf{Groin} \sqsubseteq \forall \mathsf{hasPart}.(\exists \mathsf{isPartOf}.(\mathsf{Abdomen} \sqcup \mathsf{Leg}))$$

  - ...

Ontology Engineering: Case Study

What we learned:

- Ontology engineering is **error prone**
  - errors of omission (e.g., disjointness) and commission (e.g., modelling of groin)
- **Expressive features** of OWL are sometimes needed
- Sophisticated tool support is **essential**
  - handling ontologies of this size is challenging
  - domain experts (and logicians!) often need help to understand the (root) cause of both inconsistencies and non-subsumptions
  - surprising and unexplained (non-) inferences are frustrating for users and may cause them to lose faith in the ontology and/or reasoner

How to provide reasoning services?

How to provide reasoning services?

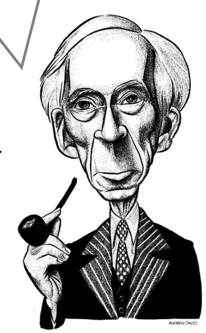
Recall what we said about semantics:

Why should I care about semantics? -- In fact I heard that a little goes a long way!

Well, from a philosophical POV, we need to specify the relationship between statements in the logic and the existential phenomena they describe.

That’s OK, but I don’t get paid for philosophy.

From a practical POV, in order to specify, build and test (ontology-based) tools/systems we need to precisely define relationships (like entailment) between logical statements – this defines the *intended* behaviour of tools/systems.



DL Semantics: Reasoning Problems

Given a knowledge base $\mathcal{K}$, and concepts C, D:

- **KB consistency**: $\mathcal{K}$ is consistent if there exists some model M s.t. $M \models \mathcal{K}$
- **Concept satisfiability**: C is satisfiable w.r.t. $\mathcal{K}$ if there exists a model $M = \langle D, \cdot^{\mathcal{I}} \rangle$ of K with $C^{\mathcal{I}} \neq \emptyset$
- **Concept subsumption**: C is subsumed by D w.r.t. $\mathcal{K}$, written $\mathcal{K} \models C \sqsubseteq D$, if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ in every model $\mathcal{I}$ of $\mathcal{K}$
- **Axiom entailment**: An axiom A is entailed by $\mathcal{K}$ (written $\mathcal{K} \models A$) if for every model M of $\mathcal{K}$, $M \models A$

\bigskip\noindent

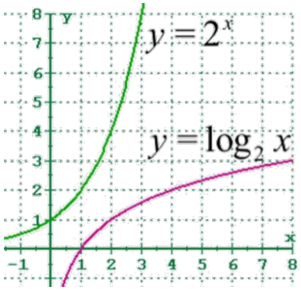
DL Semantics: Reasoning Problems

Given a knowledge base $\mathcal{K}$, and concepts C, D:

- **KB consistency**: $\mathcal{K}$ is consistent if there exists some model M s.t. $M \models \mathcal{K}$
- **Concept satisfiability**: C is satisfiable w.r.t. $\mathcal{K}$ if there exists a model $M = \langle D, \cdot^{\mathcal{I}} \rangle$ of K with $C^{\mathcal{I}} \neq \emptyset$
- **Concept subsumption**: C is subsumed by D w.r.t. $\mathcal{K}$, written $\mathcal{K} \models C \sqsubseteq D$, if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ in every model $\mathcal{I}$ of $\mathcal{K}$
- **Axiom entailment**: An axiom A is entailed by $\mathcal{K}$ (written $\mathcal{K} \models A$) if for every model M of $\mathcal{K}$, $M \models A$
- **CQ answering**: Given a KB $\mathcal{K}$ and a CQ q, compute $\mathsf{cert}(q, \mathcal{K})$

Theory $\rightsquigarrow$ Practice

- Most ontologies use **OWL** ontology language
- OWL based on **description logic** $\mathcal{SROIQ}$
  - ✔ Rich schema language
  - ✔ Clear semantics
  - ✔ Well understood computational properties (e.g., decidability, complexity)
  - ✗ N2ExpTime-comlete combined complexity
  - ✗ NP-hard data complexity (-v- AC^0 for databases)



Can we provide (empirically) scalable reasoning?

Various Approaches & Tradeoffs

1 Use full power of OWL and a complete reasoner:

- ✓ Well-suited for modeling complex domains
- ✓ Reliable answers
- ✗ High worst-case complexity
- ✗ Scalability problems for large ontologies & datasets

Complete OWL reasoners:

- E.g., FaCT++, **HermiT**, Pellet, ...
- Based on (hyper)tableau (model construction) theorem provers
- Highly optimised implementations effective on many ontologies

Various Approaches & Tradeoffs

2 Use a suitable "profile" and specialised reasoner:

OWL 2 defines language subsets, aka **profiles** that can be "more simply and/or efficiently implemented"

- **OWL 2 EL**
  - Based on $\mathcal{EL}^{++}$
  - PTime-complete for combined and data complexity
- **OWL 2 QL**
  - Based on DL-Lite
  - AC^0 data complexity (same as DBs)
- **OWL 2 RL**
  - Based on "**Description Logic Programs**" ($\approx$ DL $\cap$ LP)
  - PTime-complete for combined and data complexity

Various Approaches & Tradeoffs

2 Use a suitable “profile” and specialised reasoner:

- ✓ Tractable query answering
- ✓ Reliable answers (for inputs in the profile)
- ✗ Restricted expressivity of the ontology language
- ✗ Reasoners reject inputs outside profile

OWL 2 EL reasoners:

- E.g., CEL, ELK, ...
- Based on “consequence based” (deduction) theorem provers
- Target HCLS applications where many ontologies are (mainly) in the EL profile
- Usually support only schema reasoning (no query answering)

Various Approaches & Tradeoffs

2 Use a suitable "profile" and specialised reasoner:

- ✓ Tractable query answering
- ✓ Reliable answers (for inputs in the profile)
- ✗ Restricted expressivity of the ontology language
- ✗ Reasoners reject inputs outside profile

OWL 2 QL reasoners:

- E.g., Ontop, Mastro, ...
- Based on query rewriting
- Target applications where focus is query answering
- Data remains in RDBMs, but need ontology + mappings

Various Approaches & Tradeoffs

2 Use a suitable "profile" and specialised reasoner:

- ✓ Tractable query answering
- ✓ Reliable answers (for inputs in the profile)
- ✗ Restricted expressivity of the ontology language
- ✗ Reasoners reject inputs outside profile

OWL 2 RL reasoners:

- E.g., **RDFox**, Oracle, Sesame, Jena, OWLim, ...
- Often use chase-like materialisation techniques
- Widely used in practice to reason with large datasets
- Often incomplete even for RL (but RDFox is complete)

Various Approaches & Tradeoffs

3 Use full power of OWL and incomplete reasoner:

- ✓ Well-suited for modeling complex domains
- ✓ Favourable scalability properties
- ✓ Flexibility: no inputs rejected
- ✗ Incomplete answers (and degree of incompleteness not known)

OWL 2 RL ontology reasoners often used in this way:

- Accept any input but materialise only some entailed facts
- No way to know which if any entailments are missing (but see "Measuring & Repairing Incompleteness")
- Incompleteness can easily turn into unsoundness, e.g., via negation or aggregation

Tableau Reasoning

Tableau Algorithms

- Transform entailment to KB (in)consistency
  - $\mathcal{K} \models$ a:C iff $\mathcal{K} \cup \{$a:($\neg$C)$\}$ is *not* consistent (for new a)
  - $\mathcal{K} \models$ C $\sqsubseteq$ D iff $\mathcal{K} \cup \{$a:(C $\sqcap$ $\neg$D)$\}$ is *not* consistent (for new a)
- Start with facts explicitly asserted in ABox

  e.g., a:(C $\sqcap$ $\neg$D)
- Use expansion rules to derive new **ABox facts**

  e.g., a:C, a:$\neg$D
- Construction fails if obvious contradiction (clash)

  e.g., a:C, a:$\neg$C

Tableau Algorithms

- ABox is fully expanded if no more rules can be applied
- KB is consistent if there is some way to apply the rules so as to obtain a fully expanded and clash free Abox
  - Use backtracking search to explore all possible expansions
  - Fully expanded clash free ABox closely corresponds to model of KB
- KB is inconsistent if all possible expansions lead to a clash

Expansion Rules for $\mathcal{ALC}$

$\sqcap$-rule: if 1. $a : (C_1 \sqcap C_2) \in \mathcal{A}$, and
2. $\{a : C_1, a : C_2\} \not\subseteq \mathcal{A}$
then set $\mathcal{A}_1 = \mathcal{A} \cup \{a : C_1, a : C_2\}$

$\sqcup$-rule: if 1. $a : (C_1 \sqcup C_2) \in \mathcal{A}$, and
2. $\{a : C_1, a : C_2\} \cap \mathcal{A} = \emptyset$
then set $\mathcal{A}_1 = \mathcal{A} \cup \{a : C_1\}$ and $\mathcal{A}_2 = \mathcal{A} \cup \{a : C_2\}$

$\exists$-rule: if 1. $a : (\exists S.C) \in \mathcal{A}$, and
2. there is no b such that $\{\langle a, b\rangle : S, b : C\} \subseteq \mathcal{A}$,
then set $\mathcal{A}_1 = \mathcal{A} \cup \{\langle a, d\rangle : S, d : C\}$, where d is new in $\mathcal{A}$

$\forall$-rule: if 1. $\{a : (\forall S.C), \langle a, b\rangle : S\} \subseteq \mathcal{A}$, and
2. $b : C \notin \mathcal{A}$
then set $\mathcal{A}_1 = \mathcal{A} \cup \{b : C\}$

- some rules are nondeterministic, e.g., $\sqcup$, $\leq$
- implementations use backtracking search

Tableau Example

$$\begin{aligned}
\mathsf{Heart} &\sqsubseteq \mathsf{MuscularOrgan} \sqcap \\
&\quad \exists \mathsf{isPartOf}.\mathsf{CirculatorySystem} \\
\mathsf{HeartDisease} &\equiv \mathsf{Disease} \sqcap \\
&\quad \exists \mathsf{affects}.\mathsf{Heart} \\
\mathsf{VascularDisease} &\equiv \mathsf{Disease} \sqcap \\
&\quad \exists \mathsf{affects}.(\exists \mathsf{isPartOf}.\mathsf{CirculatorySystem})
\end{aligned}$$

$$\models \mathsf{HeartDisease} \sqsubseteq \mathsf{VascularDisease}\ ?$$

Tableau Example

$$
\begin{aligned}
\mathsf{Heart} &\sqsubseteq \mathsf{MuscularOrgan} \sqcap \\
&\quad \exists \mathsf{isPartOf}.\mathsf{CirculatorySystem} \\
\mathsf{HeartDisease} &\equiv \mathsf{Disease} \sqcap \\
&\quad \exists \mathsf{affects}.\mathsf{Heart} \\
\mathsf{VascularDisease} &\equiv \mathsf{Disease} \sqcap \\
&\quad \exists \mathsf{affects}.(\exists \mathsf{isPartOf}.\mathsf{CirculatorySystem})
\end{aligned}
$$

$$x : \mathsf{HeartDisease} \sqcap \neg \mathsf{VascularDisease}$$

Tableau Example

$$
\begin{aligned}
\mathsf{Heart} &\sqsubseteq \mathsf{MuscularOrgan} \sqcap \\
&\quad \exists \mathsf{isPartOf}.\mathsf{CirculatorySystem} \\
\mathsf{HeartDisease} &\equiv \mathsf{Disease} \sqcap \\
&\quad \exists \mathsf{affects}.\mathsf{Heart} \\
\mathsf{VascularDisease} &\equiv \mathsf{Disease} \sqcap \\
&\quad \exists \mathsf{affects}.(\exists \mathsf{isPartOf}.\mathsf{CirculatorySystem})
\end{aligned}
$$

$x : \mathsf{HeartDisease} \sqcap \neg \mathsf{VascularDisease}$

$x : \mathsf{HeartDisease}$

Tableau Example

$$\begin{aligned}
\mathsf{Heart} &\sqsubseteq \mathsf{MuscularOrgan} \sqcap \\
&\quad \exists \mathsf{isPartOf}.\mathsf{CirculatorySystem} \\
\mathsf{HeartDisease} &\equiv \mathsf{Disease} \sqcap \\
&\quad \exists \mathsf{affects}.\mathsf{Heart} \\
\mathsf{VascularDisease} &\equiv \mathsf{Disease} \sqcap \\
&\quad \exists \mathsf{affects}.(\exists \mathsf{isPartOf}.\mathsf{CirculatorySystem})
\end{aligned}$$

$x : \mathsf{HeartDisease} \sqcap \neg \mathsf{VascularDisease}$

$x : \mathsf{HeartDisease}$

Tableau Example

$$\begin{aligned}
\mathsf{Heart} &\sqsubseteq \mathsf{MuscularOrgan} \sqcap \\
&\quad \exists \mathsf{isPartOf}.\mathsf{CirculatorySystem} \\
\mathsf{HeartDisease} &\equiv \mathsf{Disease} \sqcap \\
&\quad \exists \mathsf{affects}.\mathsf{Heart} \\
\mathsf{VascularDisease} &\equiv \mathsf{Disease} \sqcap \\
&\quad \exists \mathsf{affects}.(\exists \mathsf{isPartOf}.\mathsf{CirculatorySystem})
\end{aligned}$$

$x : \mathsf{HeartDisease} \sqcap \neg \mathsf{VascularDisease}$

$x : \mathsf{HeartDisease}$

$x : \mathsf{Disease}$

$x : \exists \mathsf{affects}.\mathsf{Heart}$

Tableau Example

$$
\begin{aligned}
\mathsf{Heart} &\sqsubseteq \mathsf{MuscularOrgan} \sqcap \\
&\quad \exists \mathsf{isPartOf}.\mathsf{CirculatorySystem} \\
\mathsf{HeartDisease} &\equiv \mathsf{Disease} \sqcap \\
&\quad \exists \mathsf{affects}.\mathsf{Heart} \\
\mathsf{VascularDisease} &\equiv \mathsf{Disease} \sqcap \\
&\quad \exists \mathsf{affects}.(\exists \mathsf{isPartOf}.\mathsf{CirculatorySystem})
\end{aligned}
$$

$x : \mathsf{HeartDisease} \sqcap \neg \mathsf{VascularDisease}$

$x : \mathsf{HeartDisease}$

$x : \mathsf{Disease}$

$x : \exists \mathsf{affects}.\mathsf{Heart}$

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF OXFORD

DBOnto

SIRIUS

Tableau Example

$$\begin{aligned}
\mathsf{Heart} &\sqsubseteq \mathsf{MuscularOrgan} \sqcap \\
&\quad \exists \mathsf{isPartOf}.\mathsf{CirculatorySystem} \\
\mathsf{HeartDisease} &\equiv \mathsf{Disease} \sqcap \\
&\quad \exists \mathsf{affects}.\mathsf{Heart} \\
\mathsf{VascularDisease} &\equiv \mathsf{Disease} \sqcap \\
&\quad \exists \mathsf{affects}.(\exists \mathsf{isPartOf}.\mathsf{CirculatorySystem})
\end{aligned}$$

$$\begin{aligned}
x &: \mathsf{HeartDisease} \sqcap \neg \mathsf{VascularDisease} \\
x &: \mathsf{HeartDisease} \\
x &: \mathsf{Disease} \\
x &: \exists \mathsf{affects}.\mathsf{Heart} \\
(x, y) &: \mathsf{affects} \\
y &: \mathsf{Heart}
\end{aligned}$$

Tableau Example

$$
\begin{aligned}
\mathsf{Heart} &\sqsubseteq \mathsf{MuscularOrgan} \sqcap \\
&\quad \exists \mathsf{isPartOf}.\mathsf{CirculatorySystem} \\
\mathsf{HeartDisease} &\equiv \mathsf{Disease} \sqcap \\
&\quad \exists \mathsf{affects}.\mathsf{Heart} \\
\mathsf{VascularDisease} &\equiv \mathsf{Disease} \sqcap \\
&\quad \exists \mathsf{affects}.(\exists \mathsf{isPartOf}.\mathsf{CirculatorySystem})
\end{aligned}
$$

$$
\begin{aligned}
x &: \mathsf{HeartDisease} \sqcap \neg \mathsf{VascularDisease} \\
x &: \mathsf{HeartDisease} \\
x &: \mathsf{Disease} \\
x &: \exists \mathsf{affects}.\mathsf{Heart} \\
(x, y) &: \mathsf{affects} \\
y &: \mathsf{Heart}
\end{aligned}
$$

Tableau Example

$$
\begin{aligned}
\mathsf{Heart} &\sqsubseteq \mathsf{MuscularOrgan} \sqcap \\
&\quad \exists \mathsf{isPartOf}.\mathsf{CirculatorySystem} \\
\mathsf{HeartDisease} &\equiv \mathsf{Disease} \sqcap \\
&\quad \exists \mathsf{affects}.\mathsf{Heart} \\
\mathsf{VascularDisease} &\equiv \mathsf{Disease} \sqcap \\
&\quad \exists \mathsf{affects}.(\exists \mathsf{isPartOf}.\mathsf{CirculatorySystem})
\end{aligned}
$$

$$
\begin{aligned}
x &: \mathsf{HeartDisease} \sqcap \neg \mathsf{VascularDisease} \\
x &: \mathsf{HeartDisease} \\
x &: \mathsf{Disease} \\
x &: \exists \mathsf{affects}.\mathsf{Heart} \\
(x, y) &: \mathsf{affects} \\
y &: \mathsf{Heart} \\
y &: \mathsf{MuscularOrgan} \\
y &: \exists \mathsf{isPartOf}.\mathsf{CirculatorySystem}
\end{aligned}
$$

Tableau Example

$$\begin{aligned}
\mathsf{Heart} &\sqsubseteq \mathsf{MuscularOrgan} \sqcap \\
&\quad \exists \mathsf{isPartOf}.\mathsf{CirculatorySystem} \\
\mathsf{HeartDisease} &\equiv \mathsf{Disease} \sqcap \\
&\quad \exists \mathsf{affects}.\mathsf{Heart} \\
\mathsf{VascularDisease} &\equiv \mathsf{Disease} \sqcap \\
&\quad \exists \mathsf{affects}.(\exists \mathsf{isPartOf}.\mathsf{CirculatorySystem})
\end{aligned}$$

$$\begin{aligned}
x &: \mathsf{HeartDisease} \sqcap \neg \mathsf{VascularDisease} \\
x &: \mathsf{HeartDisease} \\
x &: \mathsf{Disease} \\
x &: \exists \mathsf{affects}.\mathsf{Heart} \\
(x, y) &: \mathsf{affects} \\
y &: \mathsf{Heart} \\
y &: \mathsf{MuscularOrgan} \\
y &: \exists \mathsf{isPartOf}.\mathsf{CirculatorySystem}
\end{aligned}$$

Tableau Example

$$\begin{aligned}
\mathsf{Heart} &\sqsubseteq \mathsf{MuscularOrgan} \sqcap \\
&\quad \exists \mathsf{isPartOf}.\mathsf{CirculatorySystem} \\
\mathsf{HeartDisease} &\equiv \mathsf{Disease} \sqcap \\
&\quad \exists \mathsf{affects}.\mathsf{Heart} \\
\mathsf{VascularDisease} &\equiv \mathsf{Disease} \sqcap \\
&\quad \exists \mathsf{affects}.(\exists \mathsf{isPartOf}.\mathsf{CirculatorySystem})
\end{aligned}$$

$$\begin{aligned}
x &: \mathsf{HeartDisease} \sqcap \neg \mathsf{VascularDisease} \\
x &: \mathsf{HeartDisease} \\
x &: \mathsf{Disease} \\
x &: \exists \mathsf{affects}.\mathsf{Heart} \\
(x, y) &: \mathsf{affects} \\
y &: \mathsf{Heart} \\
y &: \mathsf{MuscularOrgan} \\
y &: \exists \mathsf{isPartOf}.\mathsf{CirculatorySystem} \\
(y, z) &: \mathsf{isPartOf} \\
z &: \mathsf{CirculatorySystem}
\end{aligned}$$

Tableau Example

$$\begin{aligned}
\mathsf{Heart} &\sqsubseteq \mathsf{MuscularOrgan} \sqcap \\
&\quad \exists \mathsf{isPartOf}.\mathsf{CirculatorySystem} \\
\mathsf{HeartDisease} &\equiv \mathsf{Disease} \sqcap \\
&\quad \exists \mathsf{affects}.\mathsf{Heart} \\
\mathsf{VascularDisease} &\equiv \mathsf{Disease} \sqcap \\
&\quad \exists \mathsf{affects}.(\exists \mathsf{isPartOf}.\mathsf{CirculatorySystem})
\end{aligned}$$

$$\begin{aligned}
x &: \mathsf{HeartDisease} \sqcap \neg \mathsf{VascularDisease} \\
x &: \mathsf{HeartDisease} \\
x &: \mathsf{Disease} \\
x &: \exists \mathsf{affects}.\mathsf{Heart} \\
(x, y) &: \mathsf{affects} \\
y &: \mathsf{Heart} \\
y &: \mathsf{MuscularOrgan} \\
y &: \exists \mathsf{isPartOf}.\mathsf{CirculatorySystem} \\
(y, z) &: \mathsf{isPartOf} \\
z &: \mathsf{CirculatorySystem}
\end{aligned}$$

Tableau Example

$$
\begin{aligned}
\mathsf{Heart} &\sqsubseteq \mathsf{MuscularOrgan} \sqcap \\
&\quad \exists \mathsf{isPartOf}.\mathsf{CirculatorySystem} \\
\mathsf{HeartDisease} &\equiv \mathsf{Disease} \sqcap \\
&\quad \exists \mathsf{affects}.\mathsf{Heart} \\
\mathsf{VascularDisease} &\equiv \mathsf{Disease} \sqcap \\
&\quad \exists \mathsf{affects}.(\exists \mathsf{isPartOf}.\mathsf{CirculatorySystem})
\end{aligned}
$$

$x : \mathsf{HeartDisease} \sqcap \neg \mathsf{VascularDisease}$ $\quad$ $x : \neg \mathsf{VascularDisease}$

$$
\begin{aligned}
x &: \mathsf{HeartDisease} \\
x &: \mathsf{Disease} \\
x &: \exists \mathsf{affects}.\mathsf{Heart} \\
(x, y) &: \mathsf{affects} \\
y &: \mathsf{Heart} \\
y &: \mathsf{MuscularOrgan} \\
y &: \exists \mathsf{isPartOf}.\mathsf{CirculatorySystem} \\
(y, z) &: \mathsf{isPartOf} \\
z &: \mathsf{CirculatorySystem}
\end{aligned}
$$

Tableau Example

$$\begin{aligned}
\mathsf{Heart} &\sqsubseteq \mathsf{MuscularOrgan} \sqcap \\
&\quad \exists \mathsf{isPartOf}.\mathsf{CirculatorySystem} \\
\mathsf{HeartDisease} &\equiv \mathsf{Disease} \sqcap \\
&\quad \exists \mathsf{affects}.\mathsf{Heart} \\
\mathsf{VascularDisease} &\equiv \mathsf{Disease} \sqcap \\
&\quad \exists \mathsf{affects}.(\exists \mathsf{isPartOf}.\mathsf{CirculatorySystem})
\end{aligned}$$

$$\begin{aligned}
x &: \mathsf{HeartDisease} \sqcap \neg\mathsf{VascularDisease} \qquad x : \neg\mathsf{VascularDisease} \\
x &: \mathsf{HeartDisease} \\
x &: \mathsf{Disease} \\
x &: \exists \mathsf{affects}.\mathsf{Heart} \\
(x, y) &: \mathsf{affects} \\
y &: \mathsf{Heart} \\
y &: \mathsf{MuscularOrgan} \\
y &: \exists \mathsf{isPartOf}.\mathsf{CirculatorySystem} \\
(y, z) &: \mathsf{isPartOf} \\
z &: \mathsf{CirculatorySystem}
\end{aligned}$$

Tableau Example

$$\begin{aligned}
\mathsf{Heart} &\sqsubseteq \mathsf{MuscularOrgan} \sqcap \\
&\quad \exists \mathsf{isPartOf}.\mathsf{CirculatorySystem} \\
\mathsf{HeartDisease} &\equiv \mathsf{Disease} \sqcap \\
&\quad \exists \mathsf{affects}.\mathsf{Heart} \\
\mathsf{VascularDisease} &\equiv \mathsf{Disease} \sqcap \\
&\quad \exists \mathsf{affects}.(\exists \mathsf{isPartOf}.\mathsf{CirculatorySystem})
\end{aligned}$$

$$\begin{aligned}
x &: \mathsf{HeartDisease} \sqcap \neg \mathsf{VascularDisease} \\
x &: \mathsf{HeartDisease} \\
x &: \mathsf{Disease} \\
x &: \exists \mathsf{affects}.\mathsf{Heart} \\
(x, y) &: \mathsf{affects} \\
y &: \mathsf{Heart} \\
y &: \mathsf{MuscularOrgan} \\
y &: \exists \mathsf{isPartOf}.\mathsf{CirculatorySystem} \\
(y, z) &: \mathsf{isPartOf} \\
z &: \mathsf{CirculatorySystem}
\end{aligned}$$

$$\begin{aligned}
x &: \neg \mathsf{VascularDisease} \\
x &: \neg \mathsf{Disease} \sqcup \\
&\quad \neg \exists \mathsf{affects}.(\exists \mathsf{isPartOf}.\mathsf{CirculatorySystem})
\end{aligned}$$

Tableau Example

$$\begin{aligned}
\mathsf{Heart} &\sqsubseteq \mathsf{MuscularOrgan} \sqcap \\
&\quad \exists \mathsf{isPartOf}.\mathsf{CirculatorySystem} \\
\mathsf{HeartDisease} &\equiv \mathsf{Disease} \sqcap \\
&\quad \exists \mathsf{affects}.\mathsf{Heart} \\
\mathsf{VascularDisease} &\equiv \mathsf{Disease} \sqcap \\
&\quad \exists \mathsf{affects}.(\exists \mathsf{isPartOf}.\mathsf{CirculatorySystem})
\end{aligned}$$

$$\begin{aligned}
x &: \mathsf{HeartDisease} \sqcap \neg \mathsf{VascularDisease} \\
x &: \mathsf{HeartDisease} \\
x &: \mathsf{Disease} \\
x &: \exists \mathsf{affects}.\mathsf{Heart} \\
(x, y) &: \mathsf{affects} \\
y &: \mathsf{Heart} \\
y &: \mathsf{MuscularOrgan} \\
y &: \exists \mathsf{isPartOf}.\mathsf{CirculatorySystem} \\
(y, z) &: \mathsf{isPartOf} \\
z &: \mathsf{CirculatorySystem}
\end{aligned}$$

$$\begin{aligned}
x &: \neg \mathsf{VascularDisease} \\
x &: \neg \mathsf{Disease} \sqcup \\
&\quad \neg \exists \mathsf{affects}.(\exists \mathsf{isPartOf}.\mathsf{CirculatorySystem})
\end{aligned}$$

Tableau Example

$$
\begin{aligned}
\mathsf{Heart} &\sqsubseteq \mathsf{MuscularOrgan} \sqcap \\
&\quad \exists \mathsf{isPartOf}.\mathsf{CirculatorySystem} \\
\mathsf{HeartDisease} &\equiv \mathsf{Disease} \sqcap \\
&\quad \exists \mathsf{affects}.\mathsf{Heart} \\
\mathsf{VascularDisease} &\equiv \mathsf{Disease} \sqcap \\
&\quad \exists \mathsf{affects}.(\exists \mathsf{isPartOf}.\mathsf{CirculatorySystem})
\end{aligned}
$$

$$
\begin{aligned}
x &: \mathsf{HeartDisease} \sqcap \neg \mathsf{VascularDisease} \\
x &: \mathsf{HeartDisease} \\
x &: \mathsf{Disease} \\
x &: \exists \mathsf{affects}.\mathsf{Heart} \\
(x, y) &: \mathsf{affects} \\
y &: \mathsf{Heart} \\
y &: \mathsf{MuscularOrgan} \\
y &: \exists \mathsf{isPartOf}.\mathsf{CirculatorySystem} \\
(y, z) &: \mathsf{isPartOf} \\
z &: \mathsf{CirculatorySystem}
\end{aligned}
$$

$$
\begin{aligned}
x &: \neg \mathsf{VascularDisease} \\
x &: \neg \mathsf{Disease} \sqcup \\
&\quad \neg \exists \mathsf{affects}.(\exists \mathsf{isPartOf}.\mathsf{CirculatorySystem}) \\
x &: \neg \mathsf{Disease}
\end{aligned}
$$

Tableau Example

$$
\begin{aligned}
\mathsf{Heart} &\sqsubseteq \mathsf{MuscularOrgan} \sqcap \\
&\quad \exists \mathsf{isPartOf}.\mathsf{CirculatorySystem} \\
\mathsf{HeartDisease} &\equiv \mathsf{Disease} \sqcap \\
&\quad \exists \mathsf{affects}.\mathsf{Heart} \\
\mathsf{VascularDisease} &\equiv \mathsf{Disease} \sqcap \\
&\quad \exists \mathsf{affects}.(\exists \mathsf{isPartOf}.\mathsf{CirculatorySystem})
\end{aligned}
$$

$$
\begin{aligned}
x &: \mathsf{HeartDisease} \sqcap \neg\mathsf{VascularDisease} \\
x &: \mathsf{HeartDisease} \\
x &: \mathsf{Disease} \\
x &: \exists \mathsf{affects}.\mathsf{Heart} \\
(x, y) &: \mathsf{affects} \\
y &: \mathsf{Heart} \\
y &: \mathsf{MuscularOrgan} \\
y &: \exists \mathsf{isPartOf}.\mathsf{CirculatorySystem} \\
(y, z) &: \mathsf{isPartOf} \\
z &: \mathsf{CirculatorySystem}
\end{aligned}
$$

$$
\begin{aligned}
x &: \neg\mathsf{VascularDisease} \\
x &: \neg\mathsf{Disease} \sqcup \\
&\quad \neg\exists \mathsf{affects}.(\exists \mathsf{isPartOf}.\mathsf{CirculatorySystem}) \\
x &: \neg\mathsf{Disease}
\end{aligned}
$$

Tableau Example

$$
\begin{aligned}
\mathsf{Heart} &\sqsubseteq \mathsf{MuscularOrgan} \sqcap \\
&\quad \exists \mathsf{isPartOf}.\mathsf{CirculatorySystem} \\
\mathsf{HeartDisease} &\equiv \mathsf{Disease} \sqcap \\
&\quad \exists \mathsf{affects}.\mathsf{Heart} \\
\mathsf{VascularDisease} &\equiv \mathsf{Disease} \sqcap \\
&\quad \exists \mathsf{affects}.(\exists \mathsf{isPartOf}.\mathsf{CirculatorySystem})
\end{aligned}
$$

$$
\begin{aligned}
x &: \mathsf{HeartDisease} \sqcap \neg \mathsf{VascularDisease} \\
x &: \mathsf{HeartDisease} \\
x &: \mathsf{Disease} \\
x &: \exists \mathsf{affects}.\mathsf{Heart} \\
(x, y) &: \mathsf{affects} \\
y &: \mathsf{Heart} \\
y &: \mathsf{MuscularOrgan} \\
y &: \exists \mathsf{isPartOf}.\mathsf{CirculatorySystem} \\
(y, z) &: \mathsf{isPartOf} \\
z &: \mathsf{CirculatorySystem}
\end{aligned}
$$

$$
\begin{aligned}
x &: \neg \mathsf{VascularDisease} \\
x &: \neg \mathsf{Disease} \sqcup \\
&\quad \neg \exists \mathsf{affects}.(\exists \mathsf{isPartOf}.\mathsf{CirculatorySystem})
\end{aligned}
$$

Tableau Example

$$
\begin{aligned}
\mathsf{Heart} &\sqsubseteq \mathsf{MuscularOrgan} \sqcap \\
&\quad \exists \mathsf{isPartOf}.\mathsf{CirculatorySystem} \\
\mathsf{HeartDisease} &\equiv \mathsf{Disease} \sqcap \\
&\quad \exists \mathsf{affects}.\mathsf{Heart} \\
\mathsf{VascularDisease} &\equiv \mathsf{Disease} \sqcap \\
&\quad \exists \mathsf{affects}.(\exists \mathsf{isPartOf}.\mathsf{CirculatorySystem})
\end{aligned}
$$

$$
\begin{aligned}
x &: \mathsf{HeartDisease} \sqcap \neg \mathsf{VascularDisease} \\
x &: \mathsf{HeartDisease} \\
x &: \mathsf{Disease} \\
x &: \exists \mathsf{affects}.\mathsf{Heart} \\
(x, y) &: \mathsf{affects} \\
y &: \mathsf{Heart} \\
y &: \mathsf{MuscularOrgan} \\
y &: \exists \mathsf{isPartOf}.\mathsf{CirculatorySystem} \\
(y, z) &: \mathsf{isPartOf} \\
z &: \mathsf{CirculatorySystem}
\end{aligned}
$$

$$
\begin{aligned}
x &: \neg \mathsf{VascularDisease} \\
x &: \neg \mathsf{Disease} \sqcup \\
&\quad \neg \exists \mathsf{affects}.(\exists \mathsf{isPartOf}.\mathsf{CirculatorySystem}) \\
x &: \neg \exists \mathsf{affects}.(\exists \mathsf{isPartOf}.\mathsf{CirculatorySystem})
\end{aligned}
$$

Tableau Example

$$\mathsf{Heart} \sqsubseteq \mathsf{MuscularOrgan} \sqcap \exists \mathsf{isPartOf}.\mathsf{CirculatorySystem}$$

$$\mathsf{HeartDisease} \equiv \mathsf{Disease} \sqcap \exists \mathsf{affects}.\mathsf{Heart}$$

$$\mathsf{VascularDisease} \equiv \mathsf{Disease} \sqcap \exists \mathsf{affects}.(\exists \mathsf{isPartOf}.\mathsf{CirculatorySystem})$$

$x : \mathsf{HeartDisease} \sqcap \neg\mathsf{VascularDisease}$

$x : \mathsf{HeartDisease}$

$x : \mathsf{Disease}$

$x : \exists \mathsf{affects}.\mathsf{Heart}$

$(x, y) : \mathsf{affects}$

$y : \mathsf{Heart}$

$y : \mathsf{MuscularOrgan}$

$y : \exists \mathsf{isPartOf}.\mathsf{CirculatorySystem}$

$(y, z) : \mathsf{isPartOf}$

$z : \mathsf{CirculatorySystem}$

$x : \neg\mathsf{VascularDisease}$

$x : \neg\mathsf{Disease} \sqcup \neg\exists \mathsf{affects}.(\exists \mathsf{isPartOf}.\mathsf{CirculatorySystem})$

$x : \neg\exists \mathsf{affects}.(\exists \mathsf{isPartOf}.\mathsf{CirculatorySystem})$

Tableau Example

$$
\begin{aligned}
\mathsf{Heart} &\sqsubseteq \mathsf{MuscularOrgan} \sqcap \\
&\quad \exists \mathsf{isPartOf}.\mathsf{CirculatorySystem} \\
\mathsf{HeartDisease} &\equiv \mathsf{Disease} \sqcap \\
&\quad \exists \mathsf{affects}.\mathsf{Heart} \\
\mathsf{VascularDisease} &\equiv \mathsf{Disease} \sqcap \\
&\quad \exists \mathsf{affects}.(\exists \mathsf{isPartOf}.\mathsf{CirculatorySystem})
\end{aligned}
$$

$$
\begin{aligned}
x &: \mathsf{HeartDisease} \sqcap \neg \mathsf{VascularDisease} \\
x &: \mathsf{HeartDisease} \\
x &: \mathsf{Disease} \\
x &: \exists \mathsf{affects}.\mathsf{Heart} \\
(x, y) &: \mathsf{affects} \\
y &: \mathsf{Heart} \\
y &: \mathsf{MuscularOrgan} \\
y &: \exists \mathsf{isPartOf}.\mathsf{CirculatorySystem} \\
(y, z) &: \mathsf{isPartOf} \\
z &: \mathsf{CirculatorySystem}
\end{aligned}
$$

$$
\begin{aligned}
x &: \neg \mathsf{VascularDisease} \\
x &: \neg \mathsf{Disease} \sqcup \\
&\quad \neg \exists \mathsf{affects}.(\exists \mathsf{isPartOf}.\mathsf{CirculatorySystem}) \\
x &: \neg \exists \mathsf{affects}.(\exists \mathsf{isPartOf}.\mathsf{CirculatorySystem}) \\
x &: \forall \mathsf{affects}.(\forall \mathsf{isPartOf}.\neg \mathsf{CirculatorySystem})
\end{aligned}
$$

Tableau Example

$$\begin{aligned}
\mathsf{Heart} &\sqsubseteq \mathsf{MuscularOrgan} \sqcap \\
&\quad \exists \mathsf{isPartOf}.\mathsf{CirculatorySystem} \\
\mathsf{HeartDisease} &\equiv \mathsf{Disease} \sqcap \\
&\quad \exists \mathsf{affects}.\mathsf{Heart} \\
\mathsf{VascularDisease} &\equiv \mathsf{Disease} \sqcap \\
&\quad \exists \mathsf{affects}.(\exists \mathsf{isPartOf}.\mathsf{CirculatorySystem})
\end{aligned}$$

$x : \mathsf{HeartDisease} \sqcap \neg\mathsf{VascularDisease}$

$x : \mathsf{HeartDisease}$

$x : \mathsf{Disease}$

$x : \exists \mathsf{affects}.\mathsf{Heart}$

$(x, y) : \mathsf{affects}$

$y : \mathsf{Heart}$

$y : \mathsf{MuscularOrgan}$

$y : \exists \mathsf{isPartOf}.\mathsf{CirculatorySystem}$

$(y, z) : \mathsf{isPartOf}$

$z : \mathsf{CirculatorySystem}$

$x : \neg\mathsf{VascularDisease}$

$x : \neg\mathsf{Disease} \sqcup \neg\exists \mathsf{affects}.(\exists \mathsf{isPartOf}.\mathsf{CirculatorySystem})$

$x : \neg\exists \mathsf{affects}.(\exists \mathsf{isPartOf}.\mathsf{CirculatorySystem})$

$x : \forall \mathsf{affects}.(\forall \mathsf{isPartOf}.\neg\mathsf{CirculatorySystem})$

Tableau Example

$$
\begin{aligned}
\mathsf{Heart} &\sqsubseteq \mathsf{MuscularOrgan} \sqcap \\
&\quad \exists \mathsf{isPartOf}.\mathsf{CirculatorySystem} \\
\mathsf{HeartDisease} &\equiv \mathsf{Disease} \sqcap \\
&\quad \exists \mathsf{affects}.\mathsf{Heart} \\
\mathsf{VascularDisease} &\equiv \mathsf{Disease} \sqcap \\
&\quad \exists \mathsf{affects}.(\exists \mathsf{isPartOf}.\mathsf{CirculatorySystem})
\end{aligned}
$$

$$
\begin{aligned}
x &: \mathsf{HeartDisease} \sqcap \neg\mathsf{VascularDisease} \\
x &: \mathsf{HeartDisease} \\
x &: \mathsf{Disease} \\
x &: \exists \mathsf{affects}.\mathsf{Heart} \\
(x, y) &: \mathsf{affects} \\
y &: \mathsf{Heart} \\
y &: \mathsf{MuscularOrgan} \\
y &: \exists \mathsf{isPartOf}.\mathsf{CirculatorySystem} \\
(y, z) &: \mathsf{isPartOf} \\
z &: \mathsf{CirculatorySystem}
\end{aligned}
$$

$$
\begin{aligned}
x &: \neg\mathsf{VascularDisease} \\
x &: \neg\mathsf{Disease} \sqcup \\
&\quad \neg\exists \mathsf{affects}.(\exists \mathsf{isPartOf}.\mathsf{CirculatorySystem}) \\
x &: \neg\exists \mathsf{affects}.(\exists \mathsf{isPartOf}.\mathsf{CirculatorySystem}) \\
x &: \forall \mathsf{affects}.(\forall \mathsf{isPartOf}.\neg\mathsf{CirculatorySystem}) \\
y &: \forall \mathsf{isPartOf}.\neg\mathsf{CirculatorySystem}
\end{aligned}
$$

Tableau Example

$$\begin{aligned}
\mathsf{Heart} &\sqsubseteq \mathsf{MuscularOrgan} \sqcap \\
&\quad \exists \mathsf{isPartOf}.\mathsf{CirculatorySystem} \\
\mathsf{HeartDisease} &\equiv \mathsf{Disease} \sqcap \\
&\quad \exists \mathsf{affects}.\mathsf{Heart} \\
\mathsf{VascularDisease} &\equiv \mathsf{Disease} \sqcap \\
&\quad \exists \mathsf{affects}.(\exists \mathsf{isPartOf}.\mathsf{CirculatorySystem})
\end{aligned}$$

$$\begin{aligned}
x &: \mathsf{HeartDisease} \sqcap \neg\mathsf{VascularDisease} \\
x &: \mathsf{HeartDisease} \\
x &: \mathsf{Disease} \\
x &: \exists \mathsf{affects}.\mathsf{Heart} \\
(x, y) &: \mathsf{affects} \\
y &: \mathsf{Heart} \\
y &: \mathsf{MuscularOrgan} \\
y &: \exists \mathsf{isPartOf}.\mathsf{CirculatorySystem} \\
(y, z) &: \mathsf{isPartOf} \\
z &: \mathsf{CirculatorySystem}
\end{aligned}$$

$$\begin{aligned}
x &: \neg\mathsf{VascularDisease} \\
x &: \neg\mathsf{Disease} \sqcup \\
&\quad \neg\exists \mathsf{affects}.(\exists \mathsf{isPartOf}.\mathsf{CirculatorySystem}) \\
x &: \neg\exists \mathsf{affects}.(\exists \mathsf{isPartOf}.\mathsf{CirculatorySystem}) \\
x &: \forall \mathsf{affects}.(\forall \mathsf{isPartOf}.\neg\mathsf{CirculatorySystem}) \\
y &: \forall \mathsf{isPartOf}.\neg\mathsf{CirculatorySystem}
\end{aligned}$$

Tableau Example

$$
\begin{aligned}
\mathsf{Heart} &\sqsubseteq \mathsf{MuscularOrgan} \sqcap \\
&\quad \exists \mathsf{isPartOf}.\mathsf{CirculatorySystem} \\
\mathsf{HeartDisease} &\equiv \mathsf{Disease} \sqcap \\
&\quad \exists \mathsf{affects}.\mathsf{Heart} \\
\mathsf{VascularDisease} &\equiv \mathsf{Disease} \sqcap \\
&\quad \exists \mathsf{affects}.(\exists \mathsf{isPartOf}.\mathsf{CirculatorySystem})
\end{aligned}
$$

$$
\begin{aligned}
x &: \mathsf{HeartDisease} \sqcap \neg\mathsf{VascularDisease} \\
x &: \mathsf{HeartDisease} \\
x &: \mathsf{Disease} \\
x &: \exists \mathsf{affects}.\mathsf{Heart} \\
(x, y) &: \mathsf{affects} \\
y &: \mathsf{Heart} \\
y &: \mathsf{MuscularOrgan} \\
y &: \exists \mathsf{isPartOf}.\mathsf{CirculatorySystem} \\
(y, z) &: \mathsf{isPartOf} \\
z &: \mathsf{CirculatorySystem}
\end{aligned}
$$

$$
\begin{aligned}
x &: \neg\mathsf{VascularDisease} \\
x &: \neg\mathsf{Disease} \sqcup \\
&\quad \neg\exists \mathsf{affects}.(\exists \mathsf{isPartOf}.\mathsf{CirculatorySystem}) \\
x &: \neg\exists \mathsf{affects}.(\exists \mathsf{isPartOf}.\mathsf{CirculatorySystem}) \\
x &: \forall \mathsf{affects}.(\forall \mathsf{isPartOf}.\neg\mathsf{CirculatorySystem}) \\
y &: \forall \mathsf{isPartOf}.\neg\mathsf{CirculatorySystem} \\
z &: \neg\mathsf{CirculatorySystem}
\end{aligned}
$$

Tableau Example

$$\begin{aligned}
\mathsf{Heart} &\sqsubseteq \mathsf{MuscularOrgan} \sqcap \\
&\quad \exists \mathsf{isPartOf}.\mathsf{CirculatorySystem} \\
\mathsf{HeartDisease} &\equiv \mathsf{Disease} \sqcap \\
&\quad \exists \mathsf{affects}.\mathsf{Heart} \\
\mathsf{VascularDisease} &\equiv \mathsf{Disease} \sqcap \\
&\quad \exists \mathsf{affects}.(\exists \mathsf{isPartOf}.\mathsf{CirculatorySystem})
\end{aligned}$$

$$\begin{aligned}
x &: \mathsf{HeartDisease} \sqcap \neg \mathsf{VascularDisease} \\
x &: \mathsf{HeartDisease} \\
x &: \mathsf{Disease} \\
x &: \exists \mathsf{affects}.\mathsf{Heart} \\
(x, y) &: \mathsf{affects} \\
y &: \mathsf{Heart} \\
y &: \mathsf{MuscularOrgan} \\
y &: \exists \mathsf{isPartOf}.\mathsf{CirculatorySystem} \\
(y, z) &: \mathsf{isPartOf} \\
z &: \mathsf{CirculatorySystem}
\end{aligned}$$

$$\begin{aligned}
x &: \neg \mathsf{VascularDisease} \\
x &: \neg \mathsf{Disease} \sqcup \\
&\quad \neg \exists \mathsf{affects}.(\exists \mathsf{isPartOf}.\mathsf{CirculatorySystem}) \\
x &: \neg \exists \mathsf{affects}.(\exists \mathsf{isPartOf}.\mathsf{CirculatorySystem}) \\
x &: \forall \mathsf{affects}.(\forall \mathsf{isPartOf}.\neg \mathsf{CirculatorySystem}) \\
y &: \forall \mathsf{isPartOf}.\neg \mathsf{CirculatorySystem} \\
z &: \neg \mathsf{CirculatorySystem}
\end{aligned}$$

Tableau Example

$$
\begin{aligned}
\mathsf{Heart} &\sqsubseteq \mathsf{MuscularOrgan} \sqcap \\
&\quad \exists \mathsf{isPartOf}.\mathsf{CirculatorySystem} \\
\mathsf{HeartDisease} &\equiv \mathsf{Disease} \sqcap \\
&\quad \exists \mathsf{affects}.\mathsf{Heart} \\
\mathsf{VascularDisease} &\equiv \mathsf{Disease} \sqcap \\
&\quad \exists \mathsf{affects}.(\exists \mathsf{isPartOf}.\mathsf{CirculatorySystem})
\end{aligned}
$$

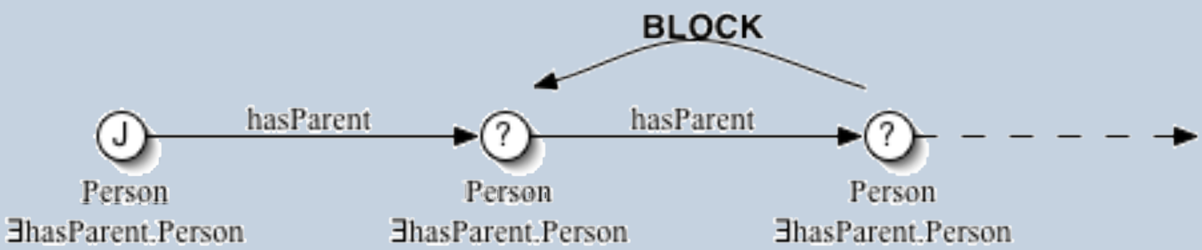
$$
\begin{aligned}
x &: \mathsf{HeartDisease} \sqcap \neg \mathsf{VascularDisease} \\
x &: \mathsf{HeartDisease} \\
x &: \mathsf{Disease} \\
x &: \exists \mathsf{affects}.\mathsf{Heart} \\
(x, y) &: \mathsf{affects} \\
y &: \mathsf{Heart} \\
y &: \mathsf{MuscularOrgan} \\
y &: \exists \mathsf{isPartOf}.\mathsf{CirculatorySystem} \\
(y, z) &: \mathsf{isPartOf} \\
z &: \mathsf{CirculatorySystem}
\end{aligned}
$$

$$
\begin{aligned}
x &: \neg \mathsf{VascularDisease} \\
x &: \neg \mathsf{Disease} \sqcup \\
&\quad \neg \exists \mathsf{affects}.(\exists \mathsf{isPartOf}.\mathsf{CirculatorySystem}) \\
x &: \neg \exists \mathsf{affects}.(\exists \mathsf{isPartOf}.\mathsf{CirculatorySystem}) \\
x &: \forall \mathsf{affects}.(\forall \mathsf{isPartOf}.\neg \mathsf{CirculatorySystem}) \\
y &: \forall \mathsf{isPartOf}.\neg \mathsf{CirculatorySystem} \\
z &: \neg \mathsf{CirculatorySystem}
\end{aligned}
$$

Note similarity to chase!

Termination

- Simplest DLs are naturally terminating
  - Rules produce strictly smaller concepts
- Most DLs require some form of **blocking**
  - E.g., {Person ⊑ ∃hasParent.Person, John:Person}



- Expressive DLs need more complex blocking

Correctness

A **decision procedure** for KB consistency

Will always give an answer, and will always give the ***right*** answer
i.e., it is correct (sound and complete) and terminating

Sound: if clash-free ABox is constructed, then KB is consistent

Given fully expanded clash-free ABox, we can trivially construct a model

Complete: if KB is consistent, then clash-free ABox is constructed

Given a model, we can use it to guide application of non-deterministic rules

Terminating: the algorithm will always produce an answer

Upper bound on number of new individuals we can create,
so ABox construction will always terminate

Highly Optimised Implementations

- Lazy unfolding (used in above example)
- Simplification and rewriting
  - Absorption: $A \sqcap B \sqsubseteq C \quad \longrightarrow \quad A \sqsubseteq C \sqcup \neg B$
- Detection of tractable fragments ($\mathcal{EL}$)
- Fast semi-decision procedures
  - Told subsumer, model merging, …
- Search optimisations
  - Dependency directed backtracking
- Reuse of previous computations
  - Of (un)satisfiable sets of concepts (conjunctions)
- Heuristics
  - Ordering don't know and don't care non-determinism

Tableau — Issues

1 Complexity

- Problem has inherently high worst case complexity
- Algorithms typically not optimal even w.r.t. worst case complexity

2 Scalability

- Highly optimised implementations often effective in practice (for schema reasoning)
- But one-by-one entailment checking can be problematical with very large ontologies
- Unclear how to extend one-by-one entailment checking approach to support scalable query answering

Hypertableau Reasoning

- Proof procedure
  - Decides truth/falsehood
  - No direct answer retrieval
- Disjunctions produce alternatives
  - Explore via backtracking

EXAMPLE

$$:Country \sqsubseteq \exists :headedBy.(:King \sqcup :President)$$
$$:Kingdom \sqsubseteq :Country \sqcap \forall :headedBy.:King$$
$$:King \sqcap :President \sqsubseteq \bot$$
$$:King \sqsubseteq :Monarch$$
$$:Country \sqcap \exists :headedBy.:Monarch \sqsubseteq :Monarchy$$

Goal: prove that every kingdom is a monarchy!

- Proof procedure
  - Decides truth/falsehood
  - No direct answer retrieval
- Disjunctions produce alternatives
  - Explore via backtracking

EXAMPLE

$$:Country \sqsubseteq \exists :headedBy.(:King \sqcup :President)$$

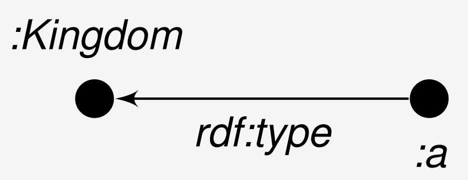
$$:Kingdom \sqsubseteq :Country \sqcap \forall :headedBy.:King$$

$$:King \sqcap :President \sqsubseteq \bot$$

$$:King \sqsubseteq :Monarch$$

$$:Country \sqcap \exists :headedBy.:Monarch \sqsubseteq :Monarchy$$

Goal: prove that every kingdom is a monarchy!

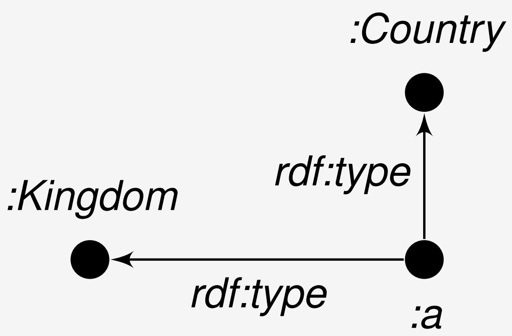


- Proof procedure
  - Decides truth/falsehood
  - No direct answer retrieval
- Disjunctions produce alternatives
  - Explore via backtracking

EXAMPLE

$$:Country \sqsubseteq \exists :headedBy.(:King \sqcup :President)$$
$$:Kingdom \sqsubseteq :Country \sqcap \forall :headedBy.:King$$
$$:King \sqcap :President \sqsubseteq \bot$$
$$:King \sqsubseteq :Monarch$$
$$:Country \sqcap \exists :headedBy.:Monarch \sqsubseteq :Monarchy$$

Goal: prove that every kingdom is a monarchy!

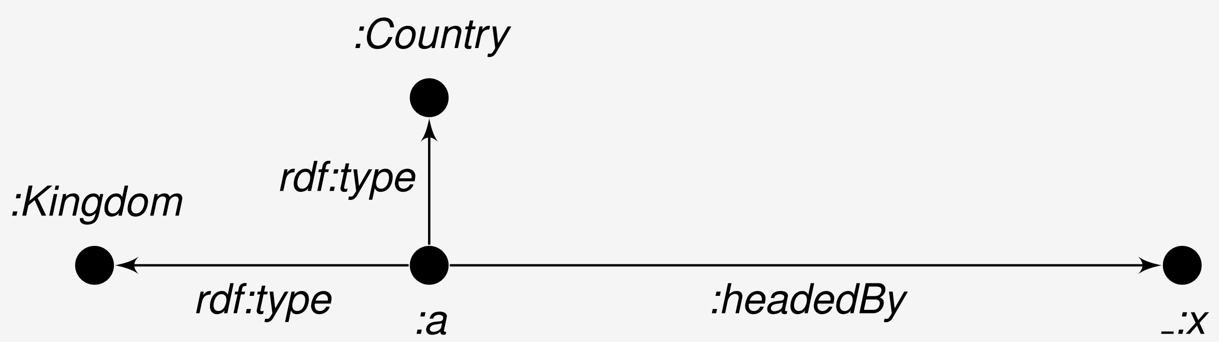


- Proof procedure
  - Decides truth/falsehood
  - No direct answer retrieval
- Disjunctions produce alternatives
  - Explore via backtracking

EXAMPLE

$$:Country \sqsubseteq \exists :headedBy.(:King \sqcup :President)$$
$$:Kingdom \sqsubseteq :Country \sqcap \forall :headedBy.:King$$
$$:King \sqcap :President \sqsubseteq \bot$$
$$:King \sqsubseteq :Monarch$$
$$:Country \sqcap \exists :headedBy.:Monarch \sqsubseteq :Monarchy$$

Goal: prove that every kingdom is a monarchy!

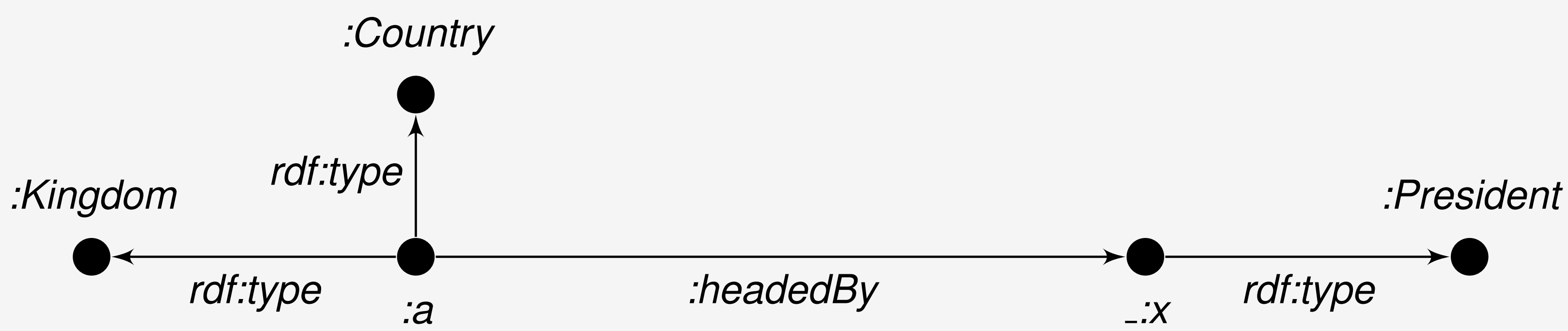


- Proof procedure
  - Decides truth/falsehood
  - No direct answer retrieval
- Disjunctions produce alternatives
  - Explore via backtracking

EXAMPLE

$$:Country \sqsubseteq \exists :headedBy.(:King \sqcup :President)$$
$$:Kingdom \sqsubseteq :Country \sqcap \forall :headedBy.:King$$
$$:King \sqcap :President \sqsubseteq \bot$$
$$:King \sqsubseteq :Monarch$$
$$:Country \sqcap \exists :headedBy.:Monarch \sqsubseteq :Monarchy$$

Goal: prove that every kingdom is a monarchy!

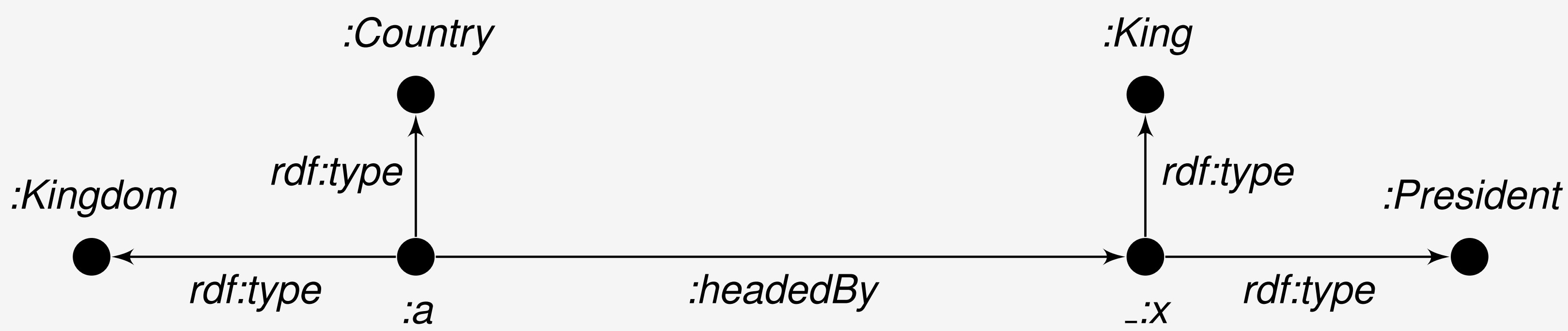


- Proof procedure
  - Decides truth/falsehood
  - No direct answer retrieval
- Disjunctions produce alternatives
  - Explore via backtracking

EXAMPLE

$$:Country \sqsubseteq \exists :headedBy.(:King \sqcup :President)$$
$$:Kingdom \sqsubseteq :Country \sqcap \forall :headedBy.:King$$
$$:King \sqcap :President \sqsubseteq \bot$$
$$:King \sqsubseteq :Monarch$$
$$:Country \sqcap \exists :headedBy.:Monarch \sqsubseteq :Monarchy$$

Goal: prove that every kingdom is a monarchy!

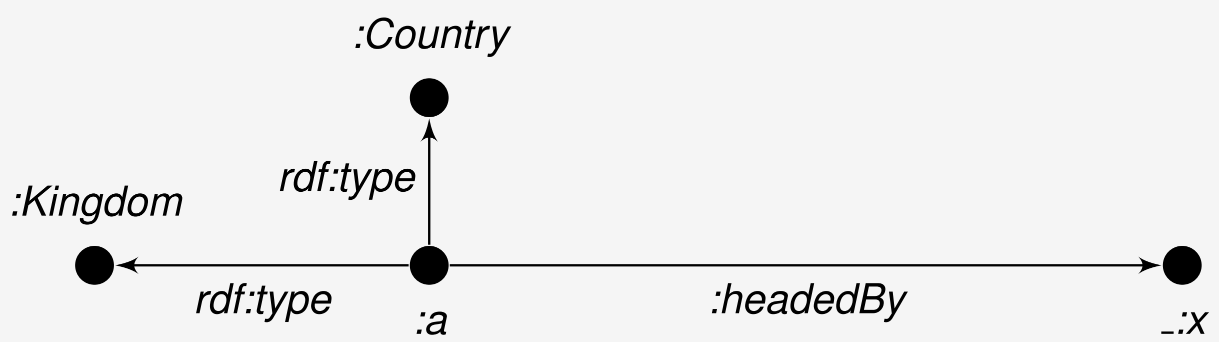


- Proof procedure
  - Decides truth/falsehood
  - No direct answer retrieval
- Disjunctions produce alternatives
  - Explore via backtracking

EXAMPLE

$$:Country \sqsubseteq \exists :headedBy.(:King \sqcup :President)$$
$$:Kingdom \sqsubseteq :Country \sqcap \forall :headedBy.:King$$
$$:King \sqcap :President \sqsubseteq \bot$$
$$:King \sqsubseteq :Monarch$$
$$:Country \sqcap \exists :headedBy.:Monarch \sqsubseteq :Monarchy$$

Goal: prove that every kingdom is a monarchy!

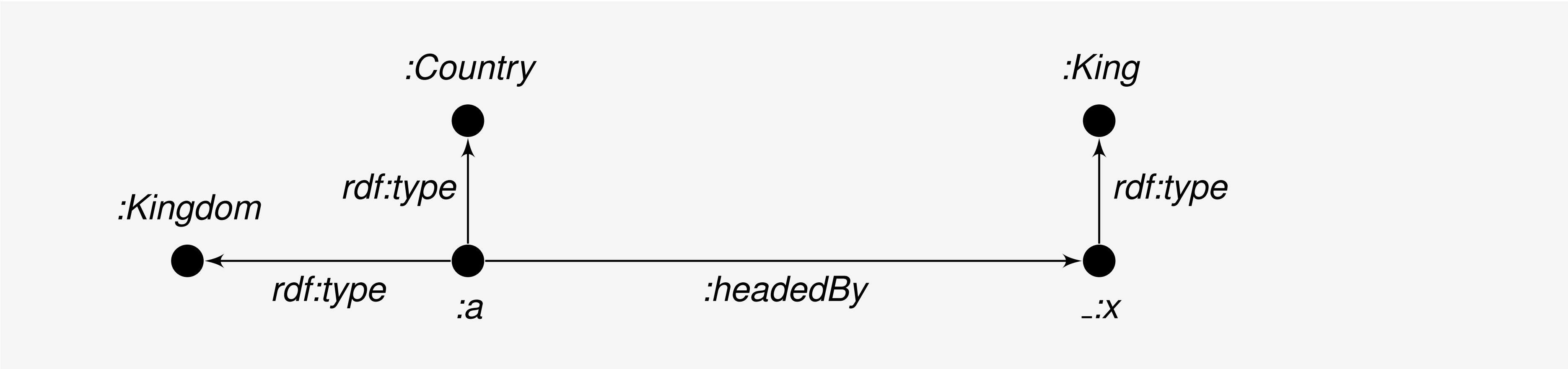


- Proof procedure
  - Decides truth/falsehood
  - No direct answer retrieval
- Disjunctions produce alternatives
  - Explore via backtracking

EXAMPLE

$$:Country \sqsubseteq \exists :headedBy.(:King \sqcup :President)$$
$$:Kingdom \sqsubseteq :Country \sqcap \forall :headedBy.:King$$
$$:King \sqcap :President \sqsubseteq \bot$$
$$:King \sqsubseteq :Monarch$$
$$:Country \sqcap \exists :headedBy.:Monarch \sqsubseteq :Monarchy$$

Goal: prove that every kingdom is a monarchy!

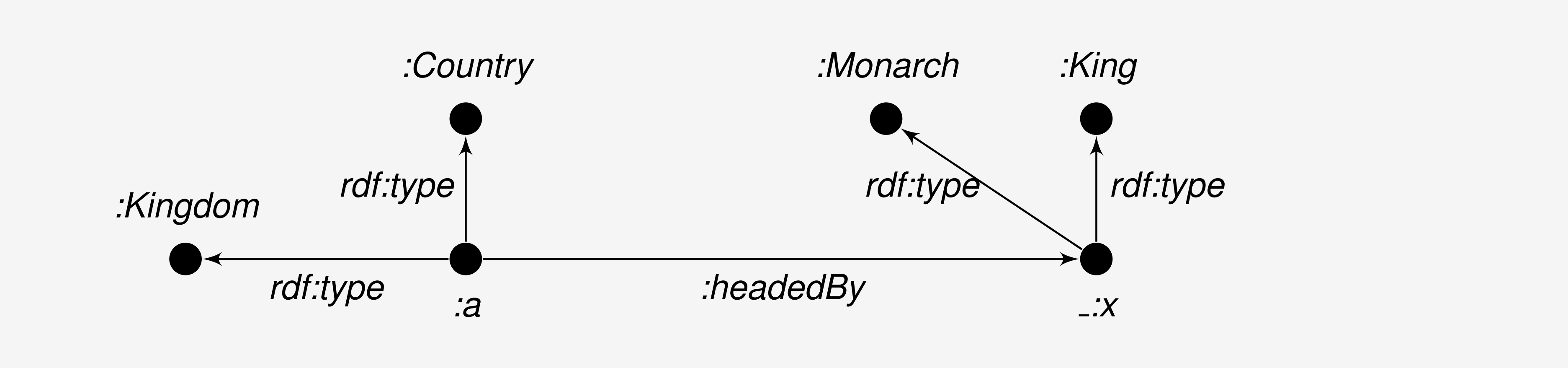


- Proof procedure
  - Decides truth/falsehood
  - No direct answer retrieval
- Disjunctions produce alternatives
  - Explore via backtracking

EXAMPLE

$$:Country \sqsubseteq \exists :headedBy.(:King \sqcup :President)$$
$$:Kingdom \sqsubseteq :Country \sqcap \forall :headedBy.:King$$
$$:King \sqcap :President \sqsubseteq \bot$$
$$:King \sqsubseteq :Monarch$$
$$:Country \sqcap \exists :headedBy.:Monarch \sqsubseteq :Monarchy$$

Goal: prove that every kingdom is a monarchy!

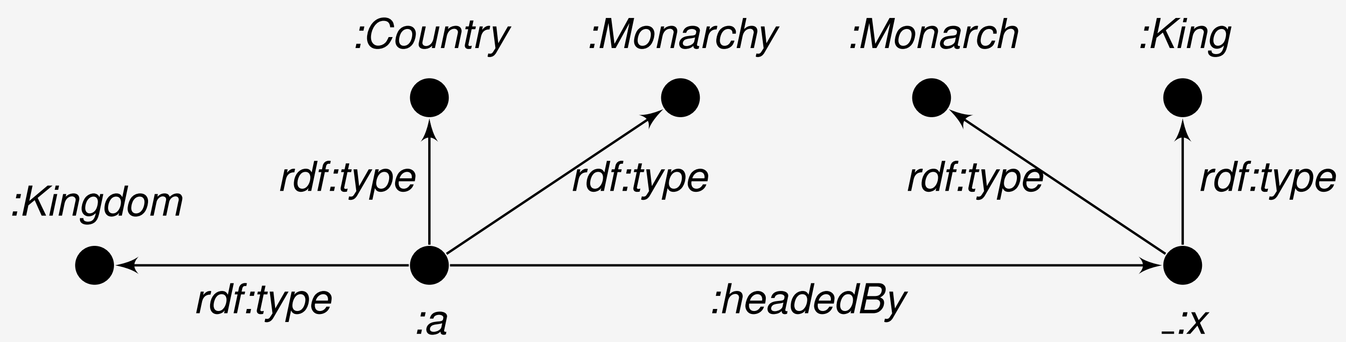


- Proof procedure
  - Decides truth/falsehood
  - No direct answer retrieval
- Disjunctions produce alternatives
  - Explore via backtracking

EXAMPLE

$$:Country \sqsubseteq \exists :headedBy.(:King \sqcup :President)$$
$$:Kingdom \sqsubseteq :Country \sqcap \forall :headedBy.:King$$
$$:King \sqcap :President \sqsubseteq \bot$$
$$:King \sqsubseteq :Monarch$$
$$:Country \sqcap \exists :headedBy.:Monarch \sqsubseteq :Monarchy$$

Goal: prove that every kingdom is a monarchy!



Consequence Based Reasoning

Consequence Based — How Does It Work?

- Normalise ontology axioms to standard form:

$$A \sqsubseteq B \quad A \sqcap B \sqsubseteq C \quad A \sqsubseteq \exists R.B \quad \exists R.B \sqsubseteq C$$

- Saturate using inference rules (for $\mathcal{EL}$):

$$\frac{A \sqsubseteq B \quad B \sqsubseteq C}{A \sqsubseteq C} \qquad \frac{A \sqsubseteq B \quad A \sqsubseteq C \quad B \sqcap C \sqsubseteq D}{A \sqsubseteq D}$$

$$\frac{A \sqsubseteq \exists R.B \quad B \sqsubseteq C \quad \exists R.C \sqsubseteq D}{A \sqsubseteq D}$$

- Extension to $\mathcal{EL}^{++}$ requires (many) more rules

Consequence Based — Example

$$\mathsf{OrganTransplant} \equiv \mathsf{Transplant} \sqcap \exists \mathsf{site}.\mathsf{Organ}$$

$$\mathsf{HeartTransplant} \equiv \mathsf{Transplant} \sqcap \exists \mathsf{site}.\mathsf{Heart}$$

$$\mathsf{Heart} \sqsubseteq \mathsf{Organ}$$

$$\models \mathsf{HeartTransplant} \sqsubseteq \mathsf{OrganTransplant}\ ?$$

Consequence Based — Example

$$\begin{aligned} \mathsf{OrganTransplant} &\equiv \mathsf{Transplant} \sqcap \exists \mathsf{site}.\mathsf{Organ} \\ \mathsf{HeartTransplant} &\equiv \mathsf{Transplant} \sqcap \exists \mathsf{site}.\mathsf{Heart} \\ \mathsf{Heart} &\sqsubseteq \mathsf{Organ} \end{aligned}$$

Consequence Based — Example

$$\mathsf{OrganTransplant} \equiv \mathsf{Transplant} \sqcap \exists \mathsf{site}.\mathsf{Organ}$$

$$\mathsf{HeartTransplant} \equiv \mathsf{Transplant} \sqcap \exists \mathsf{site}.\mathsf{Heart}$$

$$\mathsf{Heart} \sqsubseteq \mathsf{Organ}$$

Consequence Based — Example

$$\mathsf{OrganTransplant} \equiv \mathsf{Transplant} \sqcap \exists \mathsf{site}.\mathsf{Organ}$$
$$\mathsf{HeartTransplant} \equiv \mathsf{Transplant} \sqcap \exists \mathsf{site}.\mathsf{Heart}$$
$$\mathsf{Heart} \sqsubseteq \mathsf{Organ}$$

$$\mathsf{OrganTransplant} \sqsubseteq \mathsf{Transplant}$$
$$\mathsf{OrganTransplant} \sqsubseteq \exists \mathsf{site}.\mathsf{Organ}$$

Consequence Based — Example

$$\mathsf{OrganTransplant} \equiv \mathsf{Transplant} \sqcap \exists \mathsf{site}.\mathsf{Organ}$$

$$\mathsf{HeartTransplant} \equiv \mathsf{Transplant} \sqcap \exists \mathsf{site}.\mathsf{Heart}$$

$$\mathsf{Heart} \sqsubseteq \mathsf{Organ}$$

$$\mathsf{OrganTransplant} \sqsubseteq \mathsf{Transplant}$$

$$\mathsf{OrganTransplant} \sqsubseteq \exists \mathsf{site}.\mathsf{Organ}$$

$$\exists \mathsf{site}.\mathsf{Organ} \sqsubseteq \mathsf{SO}$$

Consequence Based — Example

$$\mathsf{OrganTransplant} \equiv \mathsf{Transplant} \sqcap \exists \mathsf{site}.\mathsf{Organ}$$

$$\mathsf{HeartTransplant} \equiv \mathsf{Transplant} \sqcap \exists \mathsf{site}.\mathsf{Heart}$$

$$\mathsf{Heart} \sqsubseteq \mathsf{Organ}$$

$$\mathsf{OrganTransplant} \sqsubseteq \mathsf{Transplant}$$

$$\mathsf{OrganTransplant} \sqsubseteq \exists \mathsf{site}.\mathsf{Organ}$$

$$\exists \mathsf{site}.\mathsf{Organ} \sqsubseteq \mathsf{SO}$$

$$\mathsf{Transplant} \sqcap \mathsf{SO} \sqsubseteq \mathsf{OrganTransplant}$$

Consequence Based — Example

$$\mathsf{OrganTransplant} \equiv \mathsf{Transplant} \sqcap \exists \mathsf{site}.\mathsf{Organ}$$
$$\mathsf{HeartTransplant} \equiv \mathsf{Transplant} \sqcap \exists \mathsf{site}.\mathsf{Heart}$$
$$\mathsf{Heart} \sqsubseteq \mathsf{Organ}$$

$$\mathsf{OrganTransplant} \sqsubseteq \mathsf{Transplant}$$
$$\mathsf{OrganTransplant} \sqsubseteq \exists \mathsf{site}.\mathsf{Organ}$$
$$\exists \mathsf{site}.\mathsf{Organ} \sqsubseteq \mathsf{SO}$$
$$\mathsf{Transplant} \sqcap \mathsf{SO} \sqsubseteq \mathsf{OrganTransplant}$$

Consequence Based — Example

$$\mathsf{OrganTransplant} \equiv \mathsf{Transplant} \sqcap \exists \mathsf{site}.\mathsf{Organ}$$
$$\mathsf{HeartTransplant} \equiv \mathsf{Transplant} \sqcap \exists \mathsf{site}.\mathsf{Heart}$$
$$\mathsf{Heart} \sqsubseteq \mathsf{Organ}$$

$$\mathsf{OrganTransplant} \sqsubseteq \mathsf{Transplant}$$
$$\mathsf{OrganTransplant} \sqsubseteq \exists \mathsf{site}.\mathsf{Organ}$$
$$\exists \mathsf{site}.\mathsf{Organ} \sqsubseteq \mathsf{SO}$$
$$\mathsf{Transplant} \sqcap \mathsf{SO} \sqsubseteq \mathsf{OrganTransplant}$$
$$\mathsf{HeartTransplant} \sqsubseteq \mathsf{Transplant}$$
$$\mathsf{HeartTransplant} \sqsubseteq \exists \mathsf{site}.\mathsf{Heart}$$

Consequence Based — Example

$$\mathsf{OrganTransplant} \equiv \mathsf{Transplant} \sqcap \exists \mathsf{site}.\mathsf{Organ}$$
$$\mathsf{HeartTransplant} \equiv \mathsf{Transplant} \sqcap \exists \mathsf{site}.\mathsf{Heart}$$
$$\mathsf{Heart} \sqsubseteq \mathsf{Organ}$$

$$\mathsf{OrganTransplant} \sqsubseteq \mathsf{Transplant}$$
$$\mathsf{OrganTransplant} \sqsubseteq \exists \mathsf{site}.\mathsf{Organ}$$
$$\exists \mathsf{site}.\mathsf{Organ} \sqsubseteq \mathsf{SO}$$
$$\mathsf{Transplant} \sqcap \mathsf{SO} \sqsubseteq \mathsf{OrganTransplant}$$
$$\mathsf{HeartTransplant} \sqsubseteq \mathsf{Transplant}$$
$$\mathsf{HeartTransplant} \sqsubseteq \exists \mathsf{site}.\mathsf{Heart}$$
$$\exists \mathsf{site}.\mathsf{Heart} \sqsubseteq \mathsf{SH}$$
$$\mathsf{Transplant} \sqcap \mathsf{SH} \sqsubseteq \mathsf{HeartTransplant}$$

Consequence Based — Example

$$\mathsf{OrganTransplant} \equiv \mathsf{Transplant} \sqcap \exists \mathsf{site}.\mathsf{Organ}$$
$$\mathsf{HeartTransplant} \equiv \mathsf{Transplant} \sqcap \exists \mathsf{site}.\mathsf{Heart}$$
$$\mathsf{Heart} \sqsubseteq \mathsf{Organ}$$

$$\mathsf{OrganTransplant} \sqsubseteq \mathsf{Transplant}$$
$$\mathsf{OrganTransplant} \sqsubseteq \exists \mathsf{site}.\mathsf{Organ}$$
$$\exists \mathsf{site}.\mathsf{Organ} \sqsubseteq \mathsf{SO}$$
$$\mathsf{Transplant} \sqcap \mathsf{SO} \sqsubseteq \mathsf{OrganTransplant}$$
$$\mathsf{HeartTransplant} \sqsubseteq \mathsf{Transplant}$$
$$\mathsf{HeartTransplant} \sqsubseteq \exists \mathsf{site}.\mathsf{Heart}$$
$$\exists \mathsf{site}.\mathsf{Heart} \sqsubseteq \mathsf{SH}$$
$$\mathsf{Transplant} \sqcap \mathsf{SH} \sqsubseteq \mathsf{HeartTransplant}$$

Consequence Based — Example

$$\begin{aligned}
\mathsf{OrganTransplant} &\equiv \mathsf{Transplant} \sqcap \exists \mathsf{site}.\mathsf{Organ} \\
\mathsf{HeartTransplant} &\equiv \mathsf{Transplant} \sqcap \exists \mathsf{site}.\mathsf{Heart} \\
\mathsf{Heart} &\sqsubseteq \mathsf{Organ}
\end{aligned}$$

$$\begin{aligned}
\mathsf{OrganTransplant} &\sqsubseteq \mathsf{Transplant} \\
\mathsf{OrganTransplant} &\sqsubseteq \exists \mathsf{site}.\mathsf{Organ} \\
\exists \mathsf{site}.\mathsf{Organ} &\sqsubseteq \mathsf{SO} \\
\mathsf{Transplant} \sqcap \mathsf{SO} &\sqsubseteq \mathsf{OrganTransplant} \\
\mathsf{HeartTransplant} &\sqsubseteq \mathsf{Transplant} \\
\mathsf{HeartTransplant} &\sqsubseteq \exists \mathsf{site}.\mathsf{Heart} \\
\exists \mathsf{site}.\mathsf{Heart} &\sqsubseteq \mathsf{SH} \\
\mathsf{Transplant} \sqcap \mathsf{SH} &\sqsubseteq \mathsf{HeartTransplant} \\
\mathsf{Heart} &\sqsubseteq \mathsf{Organ}
\end{aligned}$$

Consequence Based — Example

$$\mathsf{OrganTransplant} \equiv \mathsf{Transplant} \sqcap \exists \mathsf{site}.\mathsf{Organ}$$
$$\mathsf{HeartTransplant} \equiv \mathsf{Transplant} \sqcap \exists \mathsf{site}.\mathsf{Heart}$$
$$\mathsf{Heart} \sqsubseteq \mathsf{Organ}$$

$$\mathsf{OrganTransplant} \sqsubseteq \mathsf{Transplant}$$
$$\mathsf{OrganTransplant} \sqsubseteq \exists \mathsf{site}.\mathsf{Organ}$$
$$\exists \mathsf{site}.\mathsf{Organ} \sqsubseteq \mathsf{SO}$$
$$\mathsf{Transplant} \sqcap \mathsf{SO} \sqsubseteq \mathsf{OrganTransplant}$$
$$\mathsf{HeartTransplant} \sqsubseteq \mathsf{Transplant}$$
$$\mathsf{HeartTransplant} \sqsubseteq \exists \mathsf{site}.\mathsf{Heart}$$
$$\exists \mathsf{site}.\mathsf{Heart} \sqsubseteq \mathsf{SH}$$
$$\mathsf{Transplant} \sqcap \mathsf{SH} \sqsubseteq \mathsf{HeartTransplant}$$
$$\mathsf{Heart} \sqsubseteq \mathsf{Organ}$$

Consequence Based — Example

$$\mathsf{OrganTransplant} \equiv \mathsf{Transplant} \sqcap \exists \mathsf{site}.\mathsf{Organ}$$
$$\mathsf{HeartTransplant} \equiv \mathsf{Transplant} \sqcap \exists \mathsf{site}.\mathsf{Heart}$$
$$\mathsf{Heart} \sqsubseteq \mathsf{Organ}$$

$$\frac{A \sqsubseteq \exists R.B \quad B \sqsubseteq C \quad \exists R.C \sqsubseteq D}{A \sqsubseteq D}$$

$$\mathsf{OrganTransplant} \sqsubseteq \mathsf{Transplant}$$
$$\mathsf{OrganTransplant} \sqsubseteq \exists \mathsf{site}.\mathsf{Organ}$$
$$\exists \mathsf{site}.\mathsf{Organ} \sqsubseteq \mathsf{SO}$$
$$\mathsf{Transplant} \sqcap \mathsf{SO} \sqsubseteq \mathsf{OrganTransplant}$$
$$\mathsf{HeartTransplant} \sqsubseteq \mathsf{Transplant}$$
$$\mathsf{HeartTransplant} \sqsubseteq \exists \mathsf{site}.\mathsf{Heart}$$
$$\exists \mathsf{site}.\mathsf{Heart} \sqsubseteq \mathsf{SH}$$
$$\mathsf{Transplant} \sqcap \mathsf{SH} \sqsubseteq \mathsf{HeartTransplant}$$
$$\mathsf{Heart} \sqsubseteq \mathsf{Organ}$$

Consequence Based — Example

$\mathsf{OrganTransplant} \equiv \mathsf{Transplant} \sqcap \exists \mathsf{site}.\mathsf{Organ}$

$\mathsf{HeartTransplant} \equiv \mathsf{Transplant} \sqcap \exists \mathsf{site}.\mathsf{Heart}$

$\mathsf{Heart} \sqsubseteq \mathsf{Organ}$

$$\frac{A \sqsubseteq \exists R.B \quad B \sqsubseteq C \quad \exists R.C \sqsubseteq D}{A \sqsubseteq D}$$

$\mathsf{OrganTransplant} \sqsubseteq \mathsf{Transplant}$

$\mathsf{OrganTransplant} \sqsubseteq \exists \mathsf{site}.\mathsf{Organ}$

$\exists \mathsf{site}.\mathsf{Organ} \sqsubseteq \mathsf{SO}$

$\mathsf{Transplant} \sqcap \mathsf{SO} \sqsubseteq \mathsf{OrganTransplant}$

$\mathsf{HeartTransplant} \sqsubseteq \mathsf{Transplant}$

$\mathsf{HeartTransplant} \sqsubseteq \exists \mathsf{site}.\mathsf{Heart}$

$\exists \mathsf{site}.\mathsf{Heart} \sqsubseteq \mathsf{SH}$

$\mathsf{Transplant} \sqcap \mathsf{SH} \sqsubseteq \mathsf{HeartTransplant}$

$\mathsf{Heart} \sqsubseteq \mathsf{Organ}$

$\mathsf{HeartTransplant} \sqsubseteq \mathsf{SO}$

Consequence Based — Example

$$\mathsf{OrganTransplant} \equiv \mathsf{Transplant} \sqcap \exists \mathsf{site}.\mathsf{Organ}$$
$$\mathsf{HeartTransplant} \equiv \mathsf{Transplant} \sqcap \exists \mathsf{site}.\mathsf{Heart}$$
$$\mathsf{Heart} \sqsubseteq \mathsf{Organ}$$

$$\frac{A \sqsubseteq B \quad A \sqsubseteq C \quad B \sqcap C \sqsubseteq D}{A \sqsubseteq D}$$

$$\mathsf{OrganTransplant} \sqsubseteq \mathsf{Transplant}$$
$$\mathsf{OrganTransplant} \sqsubseteq \exists \mathsf{site}.\mathsf{Organ}$$
$$\exists \mathsf{site}.\mathsf{Organ} \sqsubseteq \mathsf{SO}$$
$$\mathsf{Transplant} \sqcap \mathsf{SO} \sqsubseteq \mathsf{OrganTransplant}$$
$$\mathsf{HeartTransplant} \sqsubseteq \mathsf{Transplant}$$
$$\mathsf{HeartTransplant} \sqsubseteq \exists \mathsf{site}.\mathsf{Heart}$$
$$\exists \mathsf{site}.\mathsf{Heart} \sqsubseteq \mathsf{SH}$$
$$\mathsf{Transplant} \sqcap \mathsf{SH} \sqsubseteq \mathsf{HeartTransplant}$$
$$\mathsf{Heart} \sqsubseteq \mathsf{Organ}$$

$$\mathsf{HeartTransplant} \sqsubseteq \mathsf{SO}$$

Consequence Based — Example

$\mathsf{OrganTransplant} \equiv \mathsf{Transplant} \sqcap \exists \mathsf{site}.\mathsf{Organ}$

$\mathsf{HeartTransplant} \equiv \mathsf{Transplant} \sqcap \exists \mathsf{site}.\mathsf{Heart}$

$\mathsf{Heart} \sqsubseteq \mathsf{Organ}$

$$\frac{A \sqsubseteq B \quad A \sqsubseteq C \quad B \sqcap C \sqsubseteq D}{A \sqsubseteq D}$$

$\mathsf{OrganTransplant} \sqsubseteq \mathsf{Transplant}$

$\mathsf{OrganTransplant} \sqsubseteq \exists \mathsf{site}.\mathsf{Organ}$

$\exists \mathsf{site}.\mathsf{Organ} \sqsubseteq \mathsf{SO}$

$\mathsf{Transplant} \sqcap \mathsf{SO} \sqsubseteq \mathsf{OrganTransplant}$

$\mathsf{HeartTransplant} \sqsubseteq \mathsf{Transplant}$

$\mathsf{HeartTransplant} \sqsubseteq \exists \mathsf{site}.\mathsf{Heart}$

$\exists \mathsf{site}.\mathsf{Heart} \sqsubseteq \mathsf{SH}$

$\mathsf{Transplant} \sqcap \mathsf{SH} \sqsubseteq \mathsf{HeartTransplant}$

$\mathsf{Heart} \sqsubseteq \mathsf{Organ}$

$\mathsf{HeartTransplant} \sqsubseteq \mathsf{SO}$

$\mathsf{HeartTransplant} \sqsubseteq \mathsf{OrganTransplant}$

Consequence Based — Example

$$\mathsf{OrganTransplant} \equiv \mathsf{Transplant} \sqcap \exists \mathsf{site}.\mathsf{Organ}$$

$$\mathsf{HeartTransplant} \equiv \mathsf{Transplant} \sqcap \exists \mathsf{site}.\mathsf{Heart}$$

$$\mathsf{Heart} \sqsubseteq \mathsf{Organ}$$

$$\mathsf{OrganTransplant} \sqsubseteq \mathsf{Transplant}$$

$$\mathsf{OrganTransplant} \sqsubseteq \exists \mathsf{site}.\mathsf{Organ}$$

$$\exists \mathsf{site}.\mathsf{Organ} \sqsubseteq \mathsf{SO}$$

$$\mathsf{Transplant} \sqcap \mathsf{SO} \sqsubseteq \mathsf{OrganTransplant}$$

$$\mathsf{HeartTransplant} \sqsubseteq \mathsf{Transplant}$$

$$\mathsf{HeartTransplant} \sqsubseteq \exists \mathsf{site}.\mathsf{Heart}$$

$$\exists \mathsf{site}.\mathsf{Heart} \sqsubseteq \mathsf{SH}$$

$$\mathsf{Transplant} \sqcap \mathsf{SH} \sqsubseteq \mathsf{HeartTransplant}$$

$$\mathsf{Heart} \sqsubseteq \mathsf{Organ}$$

$$\mathsf{HeartTransplant} \sqsubseteq \mathsf{SO}$$

$$\mathsf{HeartTransplant} \sqsubseteq \mathsf{OrganTransplant}$$

UNIVERSITY OF OXFORD
DEPARTMENT OF COMPUTER SCIENCE
DBOnto
SIRIUS

Correctness

A **decision procedure** for classification

Will always give an answer, and will always give the ***right*** answer
i.e., it is correct (sound and complete) and terminating

Sound: if $C \sqsubseteq D$ is derived, then KB entails $C \sqsubseteq D$

Completion rules are locally correct (preserve entailments)

Complete: if $C \sqsubseteq D$ is entailed by KB, then $C \sqsubseteq D$ is derived

Completion rules cover all cases

Terminating: the algorithm will always produce an answer

Upper bound on number of axioms of the form $C \sqsubseteq D$ or $C \sqsubseteq \exists r.D$,
so completion will always "saturate"

Consequence-Based — Issues

1 Expressivity

- Existing systems mainly focus on EL profile
- Prototypical extensions to SHIQ, but not yet clear how well they will work in practice

2 Scalability

- Existing systems support only schema reasoning
- Unclear how to extend the approach to support scalable query answering

Query Rewriting

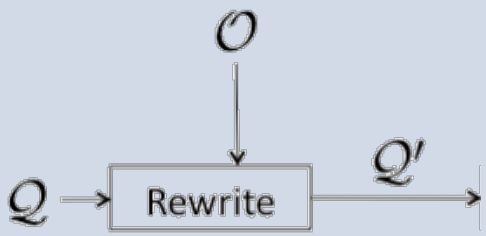
OWL 2 QL and Query Rewriting

Given QL ontology $\mathcal{O}$ query $\mathcal{Q}$ and mappings $\mathcal{M}$:

OWL 2 QL and Query Rewriting

Given QL ontology $\mathcal{O}$ query $\mathcal{Q}$ and mappings $\mathcal{M}$:

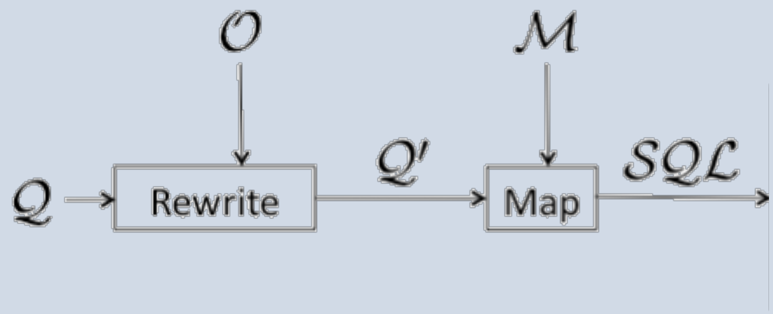
- Use $\mathcal{O}$ to **rewrite** $\mathcal{Q} \rightarrow \mathcal{Q}'$ s.t. answering $\mathcal{Q}'$ without $\mathcal{O}$ is equivalent to answering $\mathcal{Q}$ w.r.t. $\mathcal{O}$ *for any dataset*



OWL 2 QL and Query Rewriting

Given QL ontology $\mathcal{O}$ query $\mathcal{Q}$ and mappings $\mathcal{M}$:

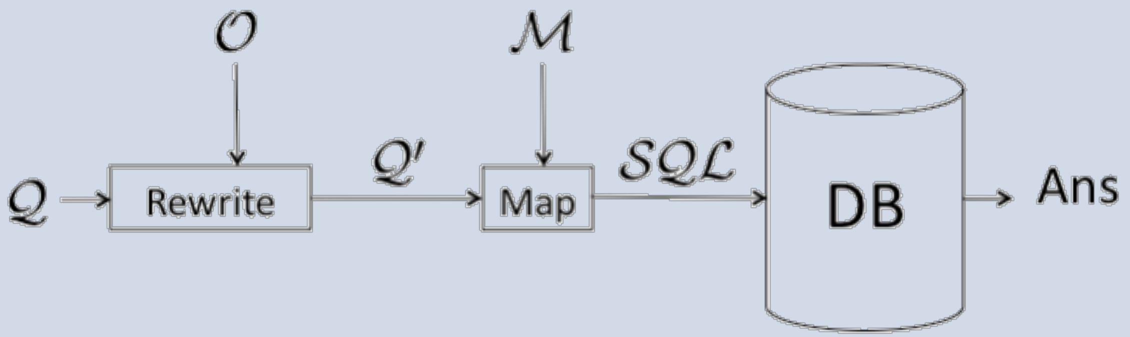
- Use $\mathcal{O}$ to **rewrite** $\mathcal{Q} \rightarrow \mathcal{Q}'$ s.t. answering $\mathcal{Q}'$ without $\mathcal{O}$ is equivalent to answering $\mathcal{Q}$ w.r.t. $\mathcal{O}$ *for any dataset*
- **Map** ontology queries $\rightarrow$ DB queries (typically SQL) using mappings $\mathcal{M}$ to rewrite $\mathcal{Q}'$ into a DB query



OWL 2 QL and Query Rewriting

Given QL ontology $\mathcal{O}$ query $\mathcal{Q}$ and mappings $\mathcal{M}$:

- Use $\mathcal{O}$ to **rewrite** $\mathcal{Q} \rightarrow \mathcal{Q}'$ s.t. answering $\mathcal{Q}'$ without $\mathcal{O}$ is equivalent to answering $\mathcal{Q}$ w.r.t. $\mathcal{O}$ *for any dataset*
- **Map** ontology queries $\rightarrow$ DB queries (typically SQL) using mappings $\mathcal{M}$ to rewrite $\mathcal{Q}'$ into a DB query
- **Evaluate** (SQL) query against DB



DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF OXFORD
DBOnto
SIRIUS

UNIVERSITY OF OXFORD
DEPARTMENT OF COMPUTER SCIENCE

DBOnto

SIRIUS

$$
\begin{aligned}
Q(?x) \leftarrow\ & (?x, \text{rdf:type}, \text{:Pipeline}) \wedge \\
& (?x, \text{:fromFacility}, ?y) \wedge \\
& (?y, \text{rdf:type}, \text{:OilFacility})
\end{aligned}
$$

$$
\begin{aligned}
Q(?x) \leftarrow\ & (?x, \text{rdf:type}, \text{:Pipeline}) \land \\
& (?x, \text{:fromFacility}, ?y) \land \\
& (?y, \text{rdf:type}, \text{:OilFacility})
\end{aligned}
$$

(:p1, rdf:type, :Pipeline)
(:p1, :fromFacility, :f1)
(:f1, rdf:type, :OilFacility)
(:p2, rdf:type, :OilPipeline)
(:p2, :fromFacility, :f2)
(:f2, rdf:type, :OilFacility)
(:p3, rdf:type, :OilPipeline)

$$Q(?x) \leftarrow (?x, \text{rdf:type}, \text{:Pipeline}) \wedge (?x, \text{:fromFacility}, ?y) \wedge (?y, \text{rdf:type}, \text{:OilFacility})$$

(:p1, rdf:type, :Pipeline)
(:p1, :fromFacility, :f1)
(:f1, rdf:type, :OilFacility)
(:p2, rdf:type, :OilPipeline)
(:p2, :fromFacility, :f2)
(:f2, rdf:type, :OilFacility)
(:p3, rdf:type, :OilPipeline)

$$Q(?x) \leftarrow (?x, \text{rdf:type}, \text{:Pipeline}) \wedge$$
$$(?x, \text{:fromFacility}, ?y) \wedge$$
$$(?y, \text{rdf:type}, \text{:OilFacility})$$

:p1

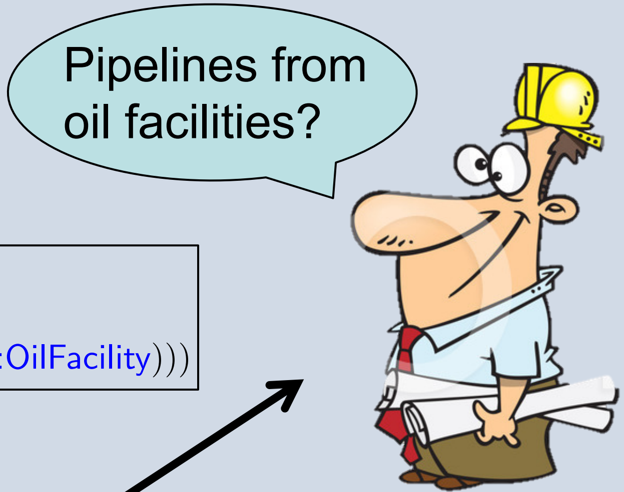
(:p1, rdf:type, :Pipeline)
(:p1, :fromFacility, :f1)
(:f1, rdf:type, :OilFacility)
(:p2, rdf:type, :OilPipeline)
(:p2, :fromFacility, :f2)
(:f2, rdf:type, :OilFacility)
(:p3, rdf:type, :OilPipeline)

UNIVERSITY OF OXFORD DEPARTMENT OF COMPUTER SCIENCE

DBOnto

SIRIUS

$$Q(?x) \leftarrow (?x, \text{rdf:type}, \text{:Pipeline}) \wedge (?x, \text{:fromFacility}, ?y) \wedge (?y, \text{rdf:type}, \text{:OilFacility})$$

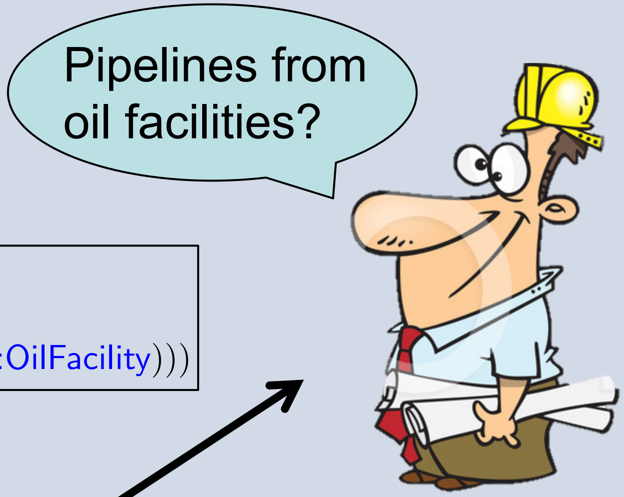



```
SubClassOf(:OilPipeline
  ObjectIntersectionOf(:Pipeline
    ObjectSomeValuesFrom(:fromFacility :OilFacility)))
```

(:p1, rdf:type, :Pipeline)
(:p1, :fromFacility, :f1)
(:f1, rdf:type, :OilFacility)
(:p2, rdf:type, :OilPipeline)
(:p2, :fromFacility, :f2)
(:f2, rdf:type, :OilFacility)
(:p3, rdf:type, :OilPipeline)

:p1

$$Q(?x) \leftarrow (?x, \text{rdf:type}, \text{:Pipeline}) \wedge (?x, \text{:fromFacility}, ?y) \wedge (?y, \text{rdf:type}, \text{:OilFacility})$$

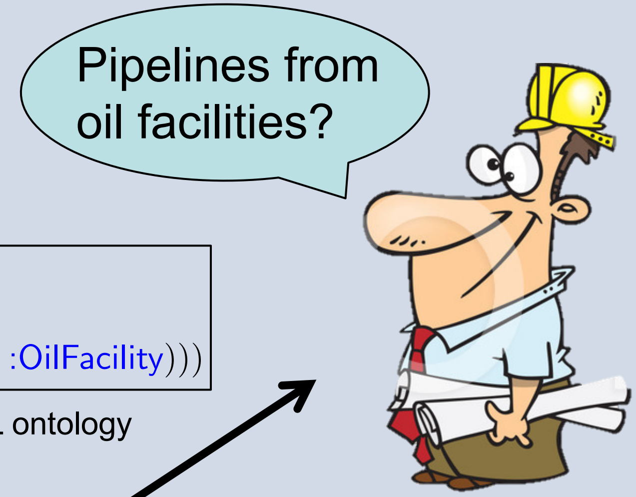



```
SubClassOf(:OilPipeline
  ObjectIntersectionOf(:Pipeline
    ObjectSomeValuesFrom(:fromFacility :OilFacility)))
```

(:p1, rdf:type, :Pipeline)
(:p1, :fromFacility, :f1)
(:f1, rdf:type, :OilFacility)
(:p2, rdf:type, :OilPipeline)
(:p2, :fromFacility, :f2)
(:f2, rdf:type, :OilFacility)
(:p3, rdf:type, :OilPipeline)

:p1, :p2, :p3

$$Q(?x) \leftarrow (?x, \text{rdf:type}, \text{:Pipeline}) \wedge (?x, \text{:fromFacility}, ?y) \wedge (?y, \text{rdf:type}, \text{:OilFacility})$$




```
SubClassOf(:OilPipeline
  ObjectIntersectionOf(:Pipeline
    ObjectSomeValuesFrom(:fromFacility :OilFacility)))
```

OWL 2 QL ontology

(:p1, rdf:type, :Pipeline)
(:p1, :fromFacility, :f1)
(:f1, rdf:type, :OilFacility)
(:p2, rdf:type, :OilPipeline)
(:p2, :fromFacility, :f2)
(:f2, rdf:type, :OilFacility)
(:p3, rdf:type, :OilPipeline)

:p1, :p2, :p3

$$Q(?x) \leftarrow (?x, \text{rdf:type}, \text{:Pipeline}) \wedge (?x, \text{:fromFacility}, ?y) \wedge (?y, \text{rdf:type}, \text{:OilFacility})$$

rewrite

SubClassOf(:OilPipeline
  ObjectIntersectionOf(:Pipeline
    ObjectSomeValuesFrom(:fromFacility :OilFacility)))

OWL 2 QL ontology

$$Q'(?x) \leftarrow (?x, \text{rdf:type}, \text{:Pipeline}) \wedge (?x, \text{:fromFacility}, ?y) \wedge (?y, \text{rdf:type}, \text{:OilFacility}) \vee (?x, \text{rdf:type}, \text{:OilPipeline})$$

(:p1, rdf:type, :Pipeline)
(:p1, :fromFacility, :f1)
(:f1, rdf:type, :OilFacility)
(:p2, rdf:type, :OilPipeline)
(:p2, :fromFacility, :f2)
(:f2, rdf:type, :OilFacility)
(:p3, rdf:type, :OilPipeline)

:p1, :p2, :p3

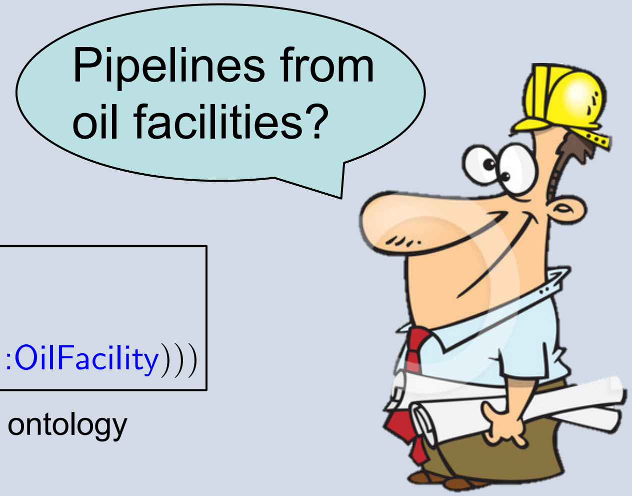
Pipelines from oil facilities?

UNIVERSITY OF OXFORD DEPARTMENT OF COMPUTER SCIENCE

DBOnto

SIRIUS

$$Q(?x) \leftarrow (?x, \text{rdf:type}, \text{:Pipeline}) \wedge (?x, \text{:fromFacility}, ?y) \wedge (?y, \text{rdf:type}, \text{:OilFacility})$$



rewrite

```
SubClassOf(:OilPipeline
  ObjectIntersectionOf(:Pipeline
    ObjectSomeValuesFrom(:fromFacility :OilFacility)))
```

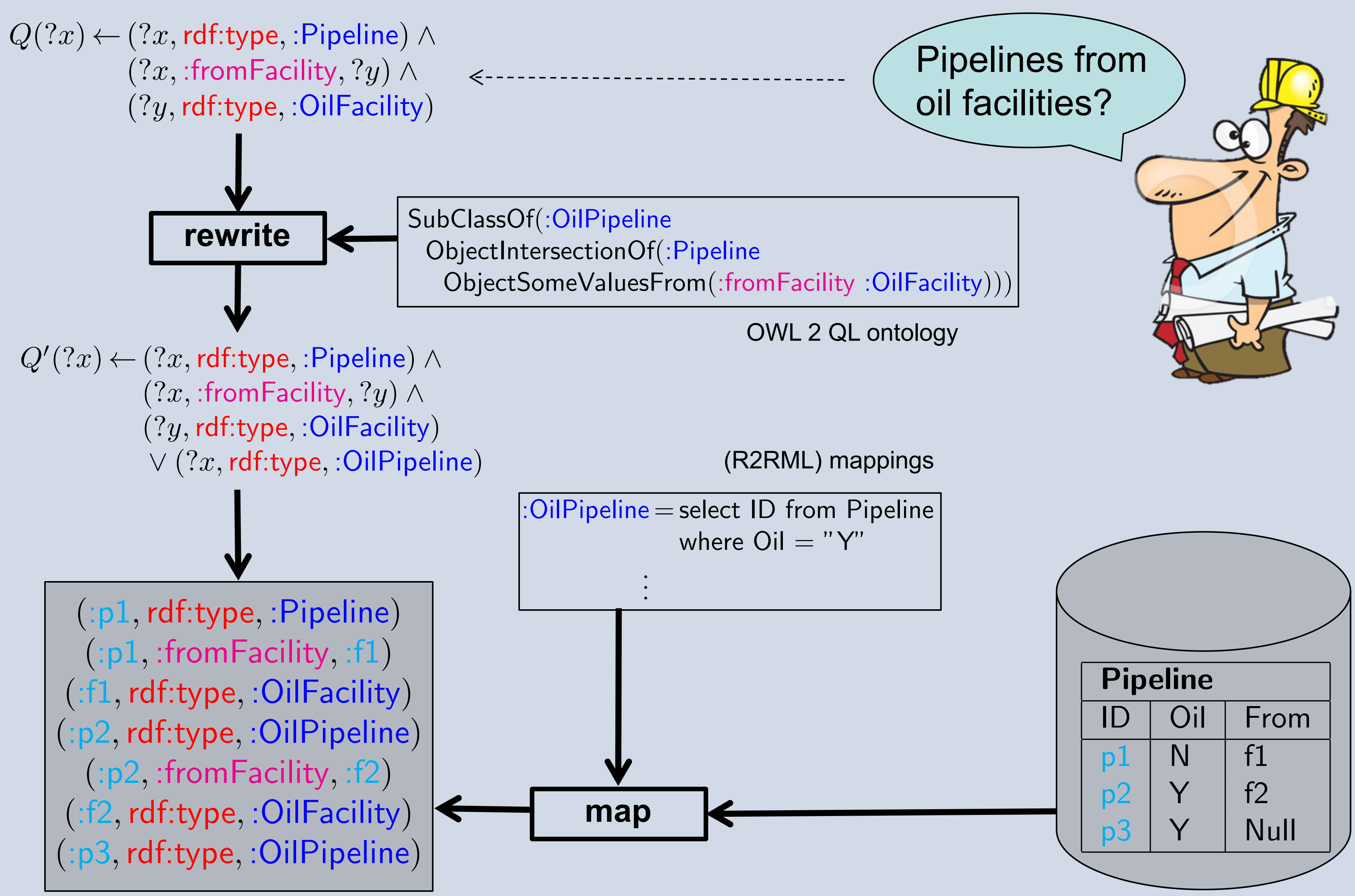
OWL 2 QL ontology

$$Q'(?x) \leftarrow (?x, \text{rdf:type}, \text{:Pipeline}) \wedge (?x, \text{:fromFacility}, ?y) \wedge (?y, \text{rdf:type}, \text{:OilFacility}) \vee (?x, \text{rdf:type}, \text{:OilPipeline})$$

Pipeline

ID	Oil	From
p1	N	f1
p2	Y	f2
p3	Y	Null

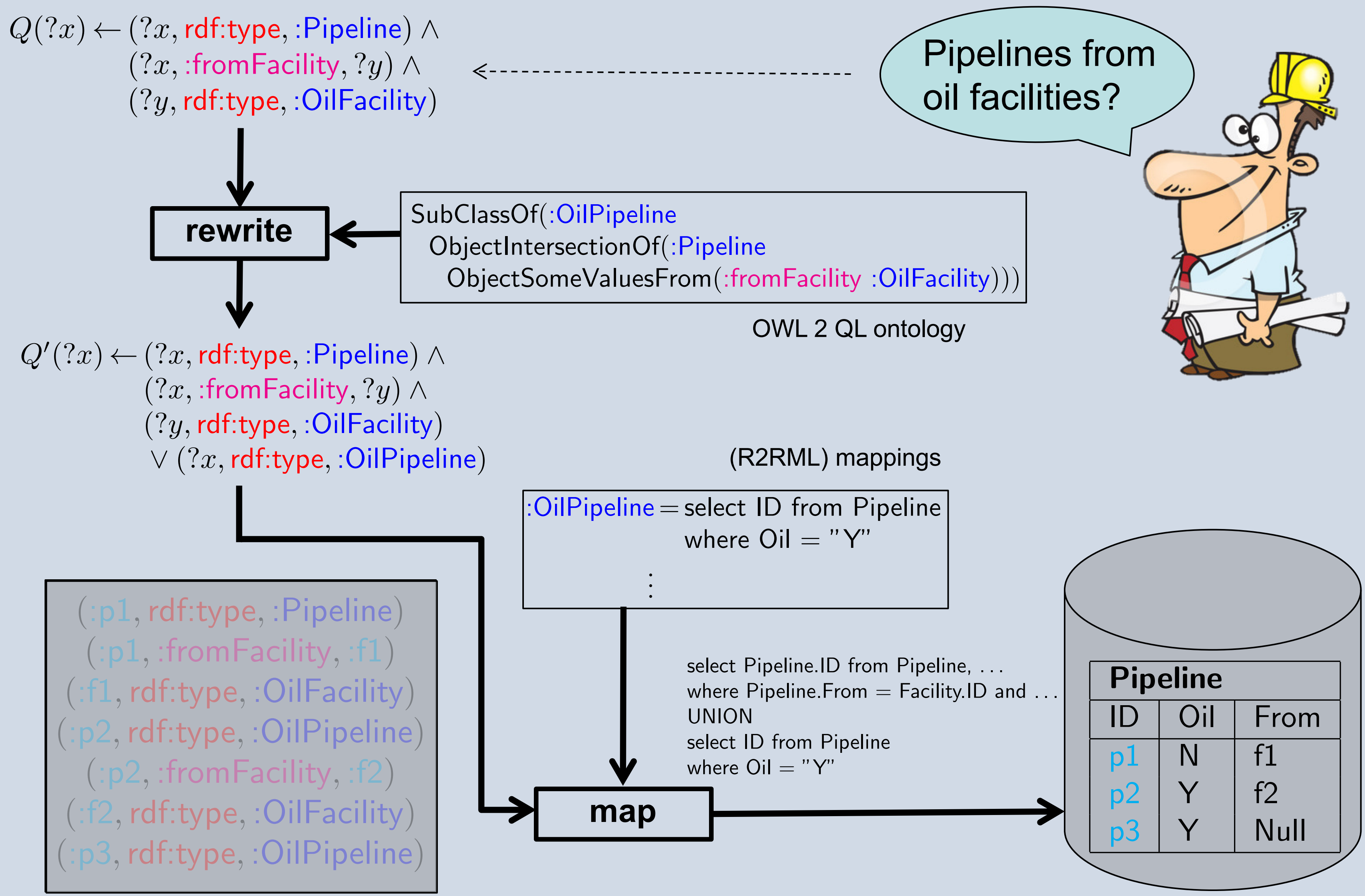
UNIVERSITY OF OXFORD DEPARTMENT OF COMPUTER SCIENCE
DBOnto
SIRIUS



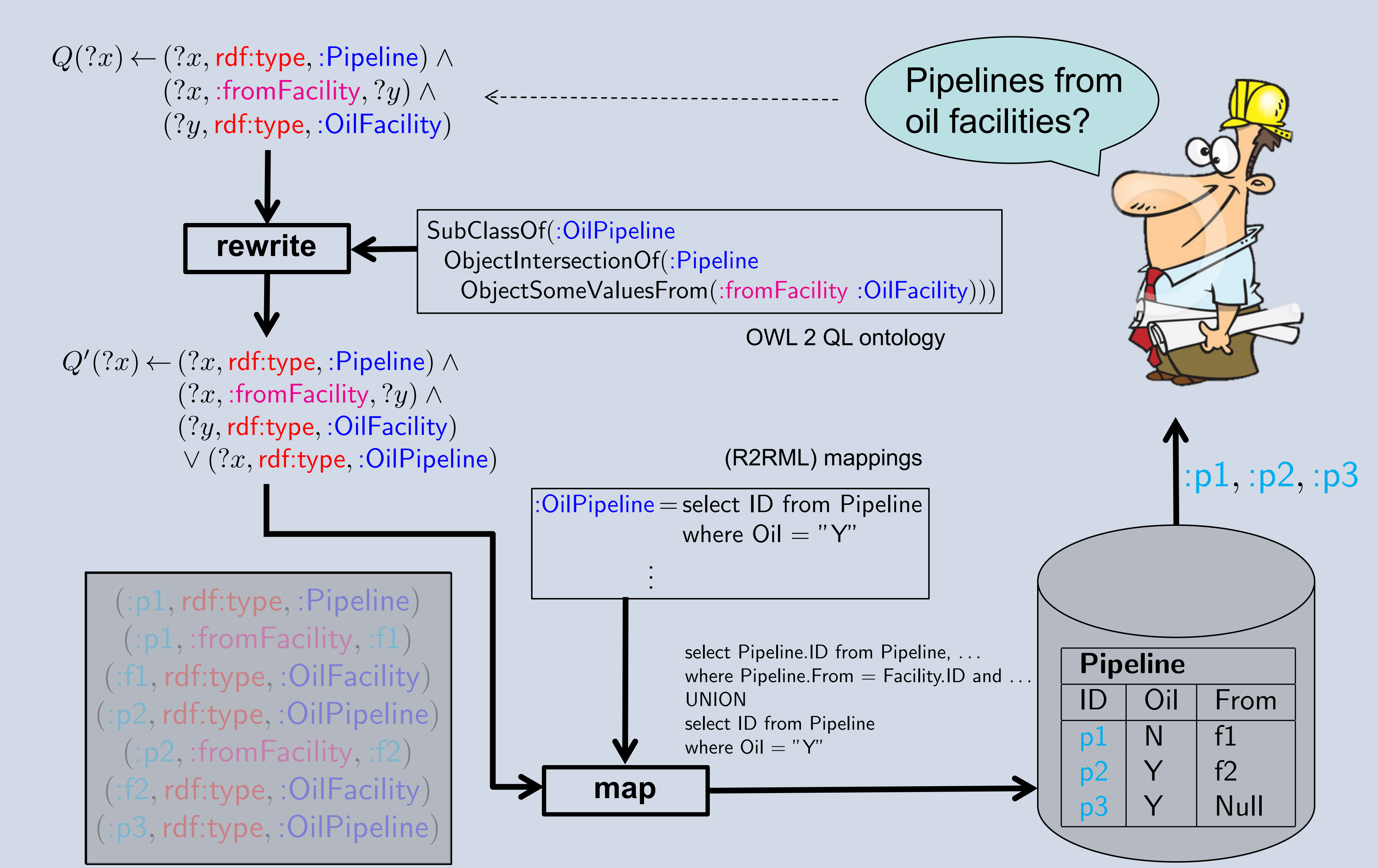
DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF OXFORD

DBOnto

SIRIUS



Pipelines from oil facilities?
Q(?x) ← (?x, rdf:type, :Pipeline) ∧ (?x, :fromFacility, ?y) ∧ (?y, rdf:type, :OilFacility)
rewrite
SubClassOf(:OilPipeline ObjectIntersectionOf(:Pipeline ObjectSomeValuesFrom(:fromFacility :OilFacility)))
OWL 2 QL ontology
Q′(?x) ← (?x, rdf:type, :Pipeline) ∧ (?x, :fromFacility, ?y) ∧ (?y, rdf:type, :OilFacility) ∨ (?x, rdf:type, :OilPipeline)
(R2RML) mappings
:OilPipeline = select ID from Pipeline where Oil = "Y"
select Pipeline.ID from Pipeline, . . .
where Pipeline.From = Facility.ID and . . .
UNION
select ID from Pipeline
where Oil = "Y"
map
(:p1, rdf:type, :Pipeline)
(:p1, :fromFacility, :f1)
(:f1, rdf:type, :OilFacility)
(:p2, rdf:type, :OilPipeline)
(:p2, :fromFacility, :f2)
(:f2, rdf:type, :OilFacility)
(:p3, rdf:type, :OilPipeline)
Pipeline
ID Oil From
p1 N f1
p2 Y f2
p3 Y Null



DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF OXFORD

DBOnto

SIRIUS

Correctness

- Rewriting can be shown to be correct

  i.e., $\mathsf{ans}(\mathcal{Q}, \mathcal{O}, \mathsf{DB}) = \mathsf{ans}(\mathcal{Q}', \emptyset, \mathsf{DB})$

- Query answer is correct iff system used to compute $\mathsf{ans}(\mathcal{Q}', \emptyset, \mathsf{DB})$ is correct
  - i.e., if DBMS is sound complete and terminating

Query Rewriting — Issues

1 Rewriting

- May be large (worst case exponential in size of ontology)
- Queries may be hard for existing DBMSs

2 Mappings

- May be difficult to develop and maintain

3 Expressivity

- OWL 2 QL (necessarily) has (very) restricted expressive power, e.g.:
  - No functional or transitive properties
  - No universal (for-all) restrictions
  - …

Materialisation Based Reasoning

Materialisation — How Does It Work?

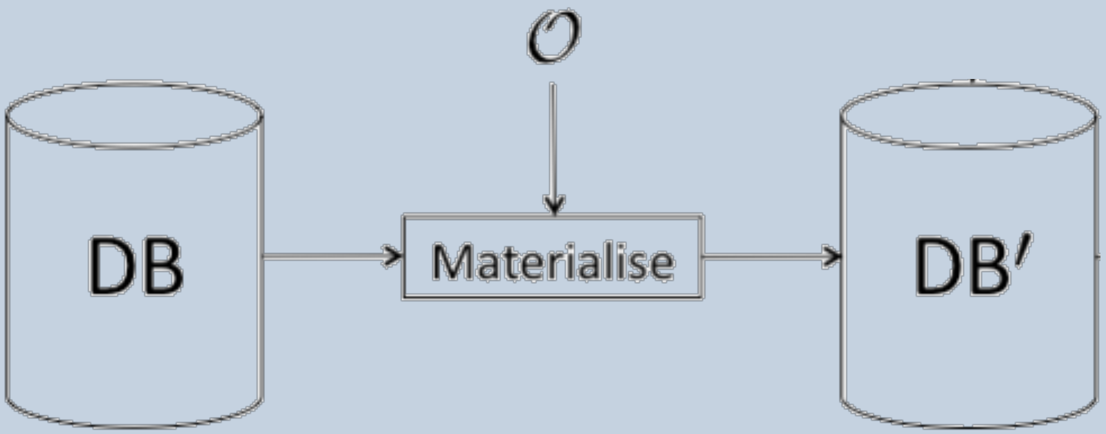
Given (RDF) data DB, ontology $\mathcal{O}$ and query $\mathcal{Q}$:

Materialisation — How Does It Work?

Given (RDF) data DB, ontology $\mathcal{O}$ and query $\mathcal{Q}$:

- **Materialise** (RDF) data DB $\rightarrow$ DB$'$ s.t. evaluating $\mathcal{Q}$ w.r.t. DB$'$ equivalent to answering $\mathcal{Q}$ w.r.t. DB and $\mathcal{O}$

  nb: Closely related to **chase** procedure used with DB dependencies



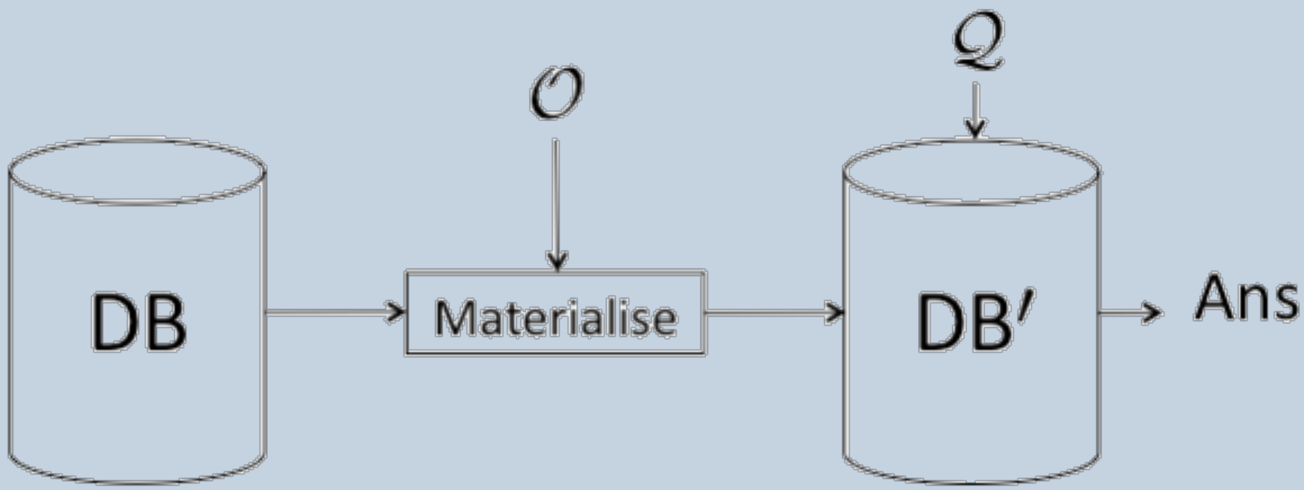
Materialisation — How Does It Work?

Given (RDF) data DB, ontology $\mathcal{O}$ and query $\mathcal{Q}$:

- **Materialise** (RDF) data DB $\rightarrow$ DB$'$ s.t. evaluating $\mathcal{Q}$ w.r.t. DB$'$ equivalent to answering $\mathcal{Q}$ w.r.t. DB and $\mathcal{O}$

  nb: Closely related to **chase** procedure used with DB dependencies

- **Evaluate** $\mathcal{Q}$ against DB$'$



Materialisation — Example

$$\mathcal{O} \left\{ \begin{array}{r} \exists \text{treats}.\text{Patient} \sqsubseteq \text{Doctor} \\ \text{Consulatant} \sqsubseteq \text{Doctor} \end{array} \right.$$

Materialisation — Example

$$\mathcal{O} \begin{cases} \exists \text{treats}.\text{Patient} \sqsubseteq \text{Doctor} \\ \text{Consulatant} \sqsubseteq \text{Doctor} \end{cases}$$

$$\text{DB} \begin{cases} \text{treats}(d_1, p_1) \\ \text{Patient}(p_1) \\ \text{Doctor}(d_2) \\ \text{Consultant}(c_1) \end{cases}$$

Materialisation — Example

$$\mathcal{O}\left\{\begin{array}{r}\exists \mathsf{treats}.\mathsf{Patient} \sqsubseteq \mathsf{Doctor}\\ \mathsf{Consulatant} \sqsubseteq \mathsf{Doctor}\end{array}\right.$$

$$\text{DB}\left\{\begin{array}{l}\mathsf{treats}(d_1, p_1)\\ \mathsf{Patient}(p_1)\\ \mathsf{Doctor}(d_2)\\ \mathsf{Consultant}(c_1)\end{array}\right.$$

$$\mathcal{Q}_1 \quad Q(x) \leftarrow \mathsf{Doctor}(y)$$

Materialisation — Example

$$\mathcal{O} \left\{ \begin{array}{r} \exists \text{treats}.\text{Patient} \sqsubseteq \text{Doctor} \\ \text{Consulatant} \sqsubseteq \text{Doctor} \end{array} \right. \qquad \begin{array}{r} \text{treats}(x, y) \wedge \text{Patient}(y) \rightarrow \text{Doctor}(x) \\ \text{Consulatant}(x) \rightarrow \text{Doctor}(x) \end{array}$$

$$\text{DB} \left\{ \begin{array}{l} \text{treats}(d_1, p_1) \\ \text{Patient}(p_1) \\ \text{Doctor}(d_2) \\ \text{Consultant}(c_1) \end{array} \right.$$

$$\mathcal{Q}_1 \quad Q(x) \leftarrow \text{Doctor}(y)$$

Materialisation — Example

$$\mathcal{O} \left\{ \begin{array}{r} \exists \text{treats}.\text{Patient} \sqsubseteq \text{Doctor} \\ \text{Consulatant} \sqsubseteq \text{Doctor} \end{array} \right. \qquad \begin{array}{r} \text{treats}(x, y) \wedge \text{Patient}(y) \rightarrow \text{Doctor}(x) \\ \text{Consulatant}(x) \rightarrow \text{Doctor}(x) \end{array}$$

$$\text{DB} \left\{ \begin{array}{l} \text{treats}(d_1, p_1) \\ \text{Patient}(p_1) \\ \text{Doctor}(d_2) \\ \text{Consultant}(c_1) \end{array} \right. \qquad \text{DB}' \left\{ \begin{array}{l} \text{treats}(d_1, p_1) \\ \text{Patient}(p_1) \\ \text{Doctor}(d_2) \\ \text{Consultant}(c_1) \\ \text{Doctor}(d_1) \\ \text{Doctor}(c_1) \end{array} \right.$$

$$\mathcal{Q}_1 \quad Q(x) \leftarrow \text{Doctor}(y)$$

Materialisation — Example

$$\mathcal{O} \begin{cases} \exists \text{treats}.\text{Patient} \sqsubseteq \text{Doctor} \\ \text{Consulatant} \sqsubseteq \text{Doctor} \end{cases}$$

$$\text{treats}(x, y) \wedge \text{Patient}(y) \rightarrow \text{Doctor}(x)$$
$$\text{Consulatant}(x) \rightarrow \text{Doctor}(x)$$

$$\text{DB} \begin{cases} \text{treats}(d_1, p_1) \\ \text{Patient}(p_1) \\ \text{Doctor}(d_2) \\ \text{Consultant}(c_1) \end{cases} \qquad \text{DB}' \begin{cases} \text{treats}(d_1, p_1) \\ \text{Patient}(p_1) \\ \text{Doctor}(d_2) \\ \text{Consultant}(c_1) \\ \text{Doctor}(d_1) \\ \text{Doctor}(c_1) \end{cases}$$

$$\mathcal{Q}_1 \quad Q(x) \leftarrow \text{Doctor}(y) \qquad \rightsquigarrow \qquad \{d_2, d_1, c_1\}$$

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF OXFORD
DBOnto
SIRIUS

Materialisation — Example

$\mathcal{O} \begin{cases} \text{Doctor} \equiv \exists \text{treats.Patient} \\ \text{Consulatant} \sqsubseteq \text{Doctor} \end{cases}$

$\text{DB} \begin{cases} \text{treats}(d_1, p_1) \\ \text{Patient}(p_1) \\ \text{Doctor}(d_2) \\ \text{Consultant}(c_1) \end{cases}$

$\text{DB}' \begin{cases} \text{treats}(d_1, p_1) \\ \text{Patient}(p_1) \\ \text{Doctor}(d_2) \\ \text{Consultant}(c_1) \\ \text{Doctor}(d_1) \\ \text{Doctor}(c_1) \end{cases}$

$\mathcal{Q}_1 \quad Q(x) \leftarrow \text{Doctor}(y) \quad \rightsquigarrow \quad \{d_2, d_1, c_1\}$

$\mathcal{Q}_2 \quad Q(x) \leftarrow \text{treats}(x, y) \wedge \text{Patient}(y) \quad \rightsquigarrow \quad \{d_1\}$

Materialisation — Issues

1 Scalability

- Ptime complete
- Efficiently implementable in practice?

2 Updates

- Additions relatively easy (continue materialisation)
- But what about retraction?

3 Migrating data to RDF

- Materialisation assumes data in "special" (RDF triple) store
- How can legacy data be migrated?

4 Expressivity

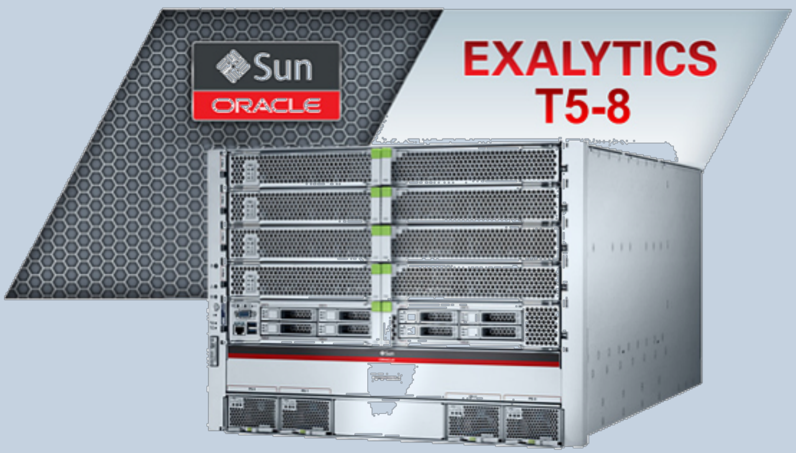
- QL $\not\subseteq$ RL; in particular, no RHS existentials (aka TGDs)

Materialisation: Scalability

- Efficient **Datalog/RL** engine is critical
- Existing approaches mainly target distributed “shared-nothing” architectures, often via **map reduce**
  - High communication overhead
  - Typically focus on small fragments (e.g., RDFS), so don’t really address expressivity issue
  - Even then, query answering over (distributed) materialized data is non-trivial and may require considerable communication

RDFox Datalog Engine

- Targets SOTA **main-memory, mulit-core** architecture
  - Optimized in-memory storage with ‘mostly’ lock-free parallel inserts
  - Memory efficient: commodity server with 128 GB can store $>10^9$ triples
  - Exploits multi-core architecture: 10-20 x speedup with 32/16 threads/cores
  - LUBM 120K ($>10^{10}$ triples) in 251s (20M t/s) on T5-8 (4TB/1024 threads)



UNIVERSITY OF OXFORD DEPARTMENT OF COMPUTER SCIENCE

DBOnto

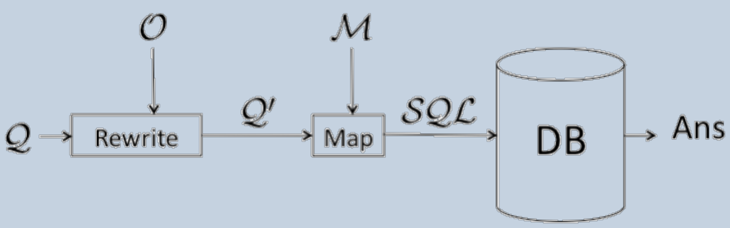
SIRIUS

RDFox Datalog Engine

- **Incremental addition and retraction** of triples
  - Retraction via novel FBF “view maintenance” algorithm
  - Retraction of 5,000 triples from materialised LUBM 50k in less than 1s
- Many other **novel features**
  - Handles more general (than RL) Dalalog and SWRL rules
  - SPARQL features such as BIND and FILTER in rule bodies
  - Native equality handling (owl:sameAs) via rewriting
  - Stratified negation as failure (NAF)

Materialisation: Data Migration

- Need to specify a suitable **migration** process
  - Use **R2RML** mappings to extract data and transform into RDF
  - But where do these mappings come from?
- Recall query rewriting:
  - **Mappings $\mathcal{M}$** *are* R2RML mappings
  - Run mappings **in reverse** to extract and transform data



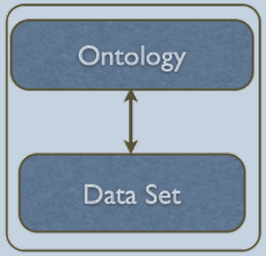
- "**Lazy ETL**"
  - Deploy query rewriting (OBDA) system
  - Extend $\mathcal{O}$ and $\mathcal{M}$ as needed
  - Use $\mathcal{M}$ to ETL data into RDF store

Materialisation: Expressivity

- RL is more powerful than QL, but $\mathsf{QL} \not\subseteq \mathsf{RL}$
  - In particular, no RHS existentials (aka TGDs)
  - Can't express, e.g., $\mathsf{OilPipeline} \sqsubseteq \mathsf{Pipeline} \sqcap \exists \mathsf{fromFacility}.\mathsf{OilFacility}$
- Recall **OWL 2 EL**
  - Based on $\mathcal{EL}^{++}$
  - Implementable via Datalog query answering plus "filtration"

OWL 2 EL via Datalog + Filtration

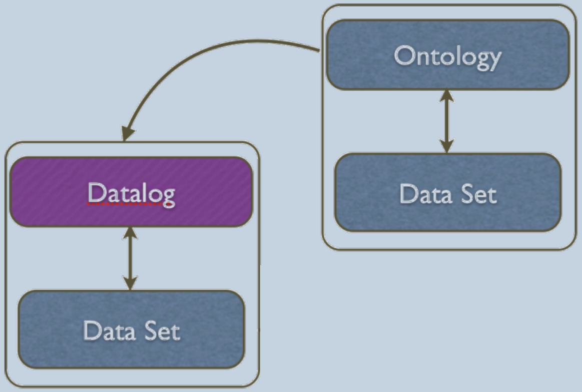
Given (RDF) Data Set, EL ontology $\mathcal{O}$ and query $\mathcal{Q}$:



OWL 2 EL via Datalog + Filtration

Given (RDF) Data Set, EL ontology $\mathcal{O}$ and query $\mathcal{Q}$:

- **Over-approximate** $\mathcal{O}$ into Datalog program D

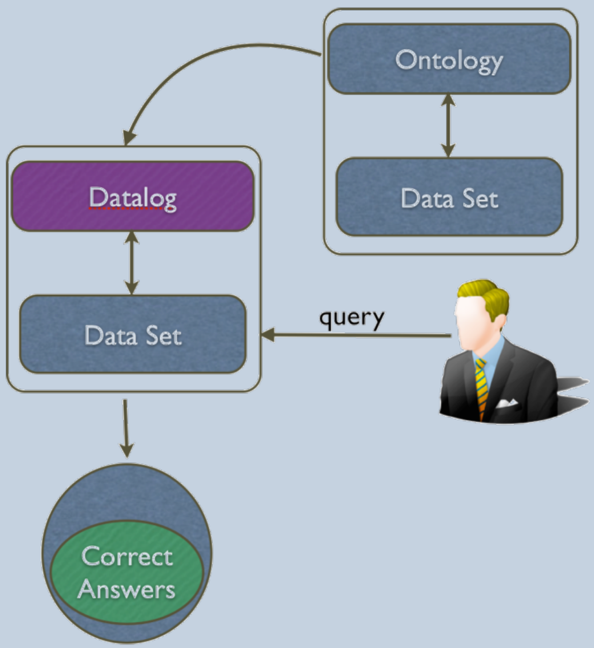


DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF OXFORD
DBOnto
SIRIUS

OWL 2 EL via Datalog + Filtration

Given (RDF) Data Set, EL ontology $\mathcal{O}$ and query $\mathcal{Q}$:

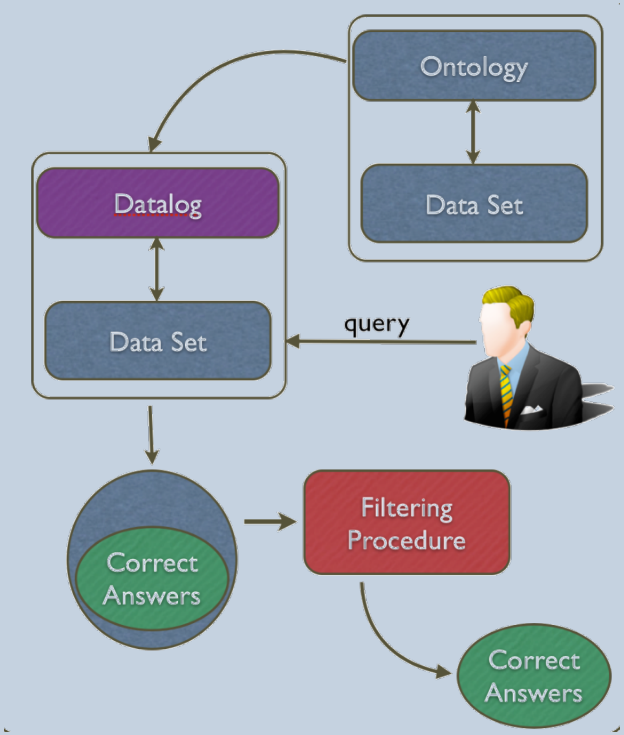
- **Over-approximate** $\mathcal{O}$ into Datalog program D
- **Evaluate** $\mathcal{Q}$ over D + Data Set (via materialisation)



OWL 2 EL via Datalog + Filtration

Given (RDF) Data Set, EL ontology $\mathcal{O}$ and query $\mathcal{Q}$:

- **Over-approximate** $\mathcal{O}$ into Datalog program D
- **Evaluate** $\mathcal{Q}$ over D + Data Set (via materialisation)
- Use (polynomial) **Filtering Procedure** to eliminate spurious answers



DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF OXFORD

Materialisation: Expressivity

- Materialisation based reasoning complete for **OWL 2 RL** profile
- Easily (and often) applied to ontologies **outside the profile**, but:
  - Reasoning may be incomplete
  - Incompleteness difficult to measure via empirical testing
- Possible solutions offered by recent work:
  - **Measuring and repairing incompleteness**
  - **Chase materialisation**
  - **Computing upper and lower bounds**

Measuring and Repairing Incompleteness

- Use ontology $\mathcal{O}$ (and query $\mathcal{Q}$) to generate a test suite

Measuring and Repairing Incompleteness

- Use ontology $\mathcal{O}$ (and query $\mathcal{Q}$) to generate a test suite
- A **test suite** for $\mathcal{O}$ is a pair $\mathbf{S} = \langle \mathbf{S}_\bot, \mathbf{S}_Q \rangle$
  - $\mathbf{S}_\bot$ a set of ABoxes that are unsatisfiable w.r.t. $\mathcal{O}$
  - $\mathbf{S}_Q$ a set of paris $\langle \mathcal{A}, \mathcal{Y} \rangle$ with $\mathcal{A}$ an ABox and $\mathcal{Y}$ a query

Measuring and Repairing Incompleteness

- Use ontology $\mathcal{O}$ (and query $\mathcal{Q}$) to generate a test suite
- A **test suite** for $\mathcal{O}$ is a pair $\mathbf{S} = \langle \mathbf{S}_\bot, \mathbf{S}_Q \rangle$
  - $\mathbf{S}_\bot$ a set of ABoxes that are unsatisfiable w.r.t. $\mathcal{O}$
  - $\mathbf{S}_Q$ a set of paris $\langle \mathcal{A}, \mathcal{Y} \rangle$ with $\mathcal{A}$ an ABox and $\mathcal{Y}$ a query
- A **reasoner** $\mathcal{R}$ passes $\mathbf{S}$ if:
  - $\mathcal{R}$ finds $\mathcal{O} \cup \mathcal{A}$ unsatisfiable for each $\mathcal{A} \in \mathbf{S}_\bot$
  - $\mathcal{R}$ complete for $\mathcal{Y}$ w.r.t. $\mathcal{O} \cup \mathcal{A}$ for each $\langle \mathcal{A}, \mathcal{Y} \rangle \in \mathbf{S}_Q$

[7] Cuenca Grau, Motik, Stoilos, and Horrocks. Completeness Guarantees for Incomplete Ontology Reasoners: Theory and Practice. JAIR, 43:419-476, 2012.

Chase Materialisation

- Applicable to **acyclic** ontologies
  - Acyclicity can be checked using, e.g., graph based techniques (weak acyclicity, joint acyclicity, etc.)
  - Many realistic ontologies turn out to be acyclic

- Given acyclic ontology $\mathcal{O}$, can apply chase materialisation:
  - Ontology translated into existential rules (aka dependencies)
  - Existential rules can introduce fresh Skolem individuals
  - Termination guaranteed for acyclic ontologies

[8] Cuenca Grau et al. Acyclicity Conditions and their Application to Query Answering in Description Logics. In Proc. of KR 2012.

Chase Materialisation — Example

$$\mathcal{O} \begin{cases} \text{Doctor} \equiv \exists \text{treats}.\text{Patient} \\ \text{Consulatant} \sqsubseteq \text{Doctor} \end{cases}$$

← **Now an equivalence!**

$$\text{DB} \begin{cases} \text{treats}(d_1, p_1) \\ \text{Patient}(p_1) \\ \text{Doctor}(d_2) \\ \text{Consultant}(c_1) \end{cases}$$

Chase Materialisation — Example

$$\mathcal{O} \begin{cases} \mathsf{Doctor} \equiv \exists \mathsf{treats}.\mathsf{Patient} \\ \mathsf{Consulatant} \sqsubseteq \mathsf{Doctor} \end{cases}$$

$$\mathsf{DB} \begin{cases} \mathsf{treats}(d_1, p_1) \\ \mathsf{Patient}(p_1) \\ \mathsf{Doctor}(d_2) \\ \mathsf{Consultant}(c_1) \end{cases}$$

$$\mathsf{DB}' \begin{cases} \mathsf{treats}(d_1, p_1) \\ \mathsf{Patient}(p_1) \\ \mathsf{Doctor}(d_2) \\ \mathsf{Consultant}(c_1) \\ \mathsf{Doctor}(d_1) \\ \mathsf{Doctor}(c_1) \\ \mathsf{treats}(d_2, f(d_2)) \\ \mathsf{Patient}(f(d_2)) \\ \mathsf{treats}(c_1, f(c_1)) \\ \mathsf{Patient}(f(c_1)) \end{cases}$$

Skolems

Chase Materialisation — Example

$$\mathcal{O} \begin{cases} \text{Doctor} \equiv \exists \text{treats}.\text{Patient} \\ \text{Consulatant} \sqsubseteq \text{Doctor} \end{cases}$$

$$\text{DB} \begin{cases} \text{treats}(d_1, p_1) \\ \text{Patient}(p_1) \\ \text{Doctor}(d_2) \\ \text{Consultant}(c_1) \end{cases}$$

$$\text{DB}' \begin{cases} \text{treats}(d_1, p_1) \\ \text{Patient}(p_1) \\ \text{Doctor}(d_2) \\ \text{Consultant}(c_1) \\ \text{Doctor}(d_1) \\ \text{Doctor}(c_1) \\ \text{treats}(d_2, f(d_2)) \\ \text{Patient}(f(d_2)) \\ \text{treats}(c_1, f(c_1)) \\ \text{Patient}(f(c_1)) \end{cases}$$

Skolems

$$\mathcal{Q}_1 \quad Q(x) \leftarrow \text{Doctor}(y)$$

Chase Materialisation — Example

$$\mathcal{O} \begin{cases} \text{Doctor} \equiv \exists \text{treats}.\text{Patient} \\ \text{Consulatant} \sqsubseteq \text{Doctor} \end{cases}$$

$$\text{DB} \begin{cases} \text{treats}(d_1, p_1) \\ \text{Patient}(p_1) \\ \text{Doctor}(d_2) \\ \text{Consultant}(c_1) \end{cases} \qquad \text{DB}' \begin{cases} \text{treats}(d_1, p_1) \\ \text{Patient}(p_1) \\ \text{Doctor}(d_2) \\ \text{Consultant}(c_1) \\ \text{Doctor}(d_1) \\ \text{Doctor}(c_1) \\ \text{treats}(d_2, f(d_2)) \\ \text{Patient}(f(d_2)) \\ \text{treats}(c_1, f(c_1)) \\ \text{Patient}(f(c_1)) \end{cases}$$

Skolems

$$\mathcal{Q}_1 \quad Q(x) \leftarrow \text{Doctor}(y) \qquad \rightsquigarrow \qquad \{d_2, d_1, c_1\}$$

Chase Materialisation — Example

$\mathcal{O}$ $\left\{ \begin{array}{r} \text{Doctor} \equiv \exists\text{treats.Patient} \\ \text{Consulatant} \sqsubseteq \text{Doctor} \end{array} \right.$

DB $\left\{ \begin{array}{l} \text{treats}(d_1, p_1) \\ \text{Patient}(p_1) \\ \text{Doctor}(d_2) \\ \text{Consultant}(c_1) \end{array} \right.$

DB′ $\left\{ \begin{array}{l} \text{treats}(d_1, p_1) \\ \text{Patient}(p_1) \\ \text{Doctor}(d_2) \\ \text{Consultant}(c_1) \\ \text{Doctor}(d_1) \\ \text{Doctor}(c_1) \\ \text{treats}(d_2, f(d_2)) \\ \text{Patient}(f(d_2)) \\ \text{treats}(c_1, f(c_1)) \\ \text{Patient}(f(c_1)) \end{array} \right.$

Skolems

$\mathcal{Q}_1 \quad Q(x) \leftarrow \text{Doctor}(y) \quad \rightsquigarrow \quad \{d_2, d_1, c_1\}$

$\mathcal{Q}_2 \quad Q(x) \leftarrow \text{treats}(x, y) \wedge \text{Patient}(y)$

Chase Materialisation — Example

$$\mathcal{O} \begin{cases} \text{Doctor} \equiv \exists \text{treats}.\text{Patient} \\ \text{Consulatant} \sqsubseteq \text{Doctor} \end{cases}$$

$$\text{DB} \begin{cases} \text{treats}(d_1, p_1) \\ \text{Patient}(p_1) \\ \text{Doctor}(d_2) \\ \text{Consultant}(c_1) \end{cases}$$

$$\text{DB}' \begin{cases} \text{treats}(d_1, p_1) \\ \text{Patient}(p_1) \\ \text{Doctor}(d_2) \\ \text{Consultant}(c_1) \\ \text{Doctor}(d_1) \\ \text{Doctor}(c_1) \\ \text{treats}(d_2, f(d_2)) \\ \text{Patient}(f(d_2)) \\ \text{treats}(c_1, f(c_1)) \\ \text{Patient}(f(c_1)) \end{cases}$$

Skolems

$\mathcal{Q}_1$ $\quad Q(x) \leftarrow \text{Doctor}(y) \quad \rightsquigarrow \quad \{d_2, d_1, c_1\}$

$\mathcal{Q}_2$ $\quad Q(x) \leftarrow \text{treats}(x, y) \wedge \text{Patient}(y) \quad \rightsquigarrow \quad \{d_1, d_2, c_1\}$

Computing Lower and Upper Bounds

- RL reasoning w.r.t. OWL ontology $\mathcal{O}$ gives lower bound answer ***L***

Computing Lower and Upper Bounds

- RL reasoning w.r.t. OWL ontology $\mathcal{O}$ gives lower bound answer ***L***
- Transform $\mathcal{O}$ into strictly stronger OWL RL ontology
  - Transform ontology into Datalog$^{\pm,\vee}$ rules
  - Eliminate $\vee$ by transforming to $\wedge$
  - Eliminate existentials by replacing with Skolem constants
  - Discard rules with empty heads
  - Transform rules into OWL 2 RL ontology $\mathcal{O}'$

Computing Lower and Upper Bounds

- RL reasonting w.r.t. $\mathcal{O}'$ gives (complete but unsound) upper bound answer ***U***

Computing Upper Bound — Example

$$\mathcal{O} \begin{cases} \text{Doctor} \equiv \exists \text{treats}.\text{Patient} \\ \text{Consulatant} \sqsubseteq \text{Doctor} \end{cases}$$

$$\text{DB} \begin{cases} \text{treats}(d_1, p_1) \\ \text{Patient}(p_1) \\ \text{Doctor}(d_2) \\ \text{Consultant}(c_1) \end{cases}$$

Computing Upper Bound — Example

$$\mathcal{O}' \begin{cases} \text{Doctor} \sqsubseteq \exists\text{treats}.\{P\} \\ \{\text{P}\} \sqsubseteq \text{Patient} \\ \exists\text{treats}.\text{Patient} \sqsubseteq \text{Doctor} \\ \text{Consulatant} \sqsubseteq \text{Doctor} \end{cases}$$

$$\text{DB} \begin{cases} \text{treats}(d_1, p_1) \\ \text{Patient}(p_1) \\ \text{Doctor}(d_2) \\ \text{Consultant}(c_1) \end{cases}$$

Computing Upper Bound — Example

$$\mathcal{O}' \left\{ \begin{array}{r} \text{Doctor} \sqsubseteq \exists \text{treats}.\{P\} \\ \{\text{P}\} \sqsubseteq \text{Patient} \\ \exists \text{treats}.\text{Patient} \sqsubseteq \text{Doctor} \\ \text{Consulatant} \sqsubseteq \text{Doctor} \end{array} \right.$$

$$\text{DB} \left\{ \begin{array}{l} \text{treats}(d_1, p_1) \\ \text{Patient}(p_1) \\ \text{Doctor}(d_2) \\ \text{Consultant}(c_1) \end{array} \right.$$

$$\text{DB}' \left\{ \begin{array}{l} \text{treats}(d_1, p_1) \\ \text{Patient}(p_1) \\ \text{Doctor}(d_2) \\ \text{Consultant}(c_1) \\ \text{Patient}(P) \\ \text{Doctor}(d_1) \\ \text{Doctor}(c_1) \\ \text{treats}(d_1, P) \\ \text{treats}(d_2, P) \\ \text{treats}(c_1, P) \end{array} \right.$$

Computing Upper Bound — Example

$$\mathcal{O}' \begin{cases} \text{Doctor} \sqsubseteq \exists \text{treats}.\{P\} \\ \{\text{P}\} \sqsubseteq \text{Patient} \\ \exists \text{treats}.\text{Patient} \sqsubseteq \text{Doctor} \\ \text{Consulatant} \sqsubseteq \text{Doctor} \end{cases}$$

$$\text{DB} \begin{cases} \text{treats}(d_1, p_1) \\ \text{Patient}(p_1) \\ \text{Doctor}(d_2) \\ \text{Consultant}(c_1) \end{cases}$$

$$\text{DB}' \begin{cases} \text{treats}(d_1, p_1) \\ \text{Patient}(p_1) \\ \text{Doctor}(d_2) \\ \text{Consultant}(c_1) \\ \text{Patient}(P) \\ \text{Doctor}(d_1) \\ \text{Doctor}(c_1) \\ \text{treats}(d_1, P) \\ \text{treats}(d_2, P) \\ \text{treats}(c_1, P) \end{cases}$$

$\mathcal{Q}_1 \quad Q(x) \leftarrow \text{Doctor}(y)$

UNIVERSITY OF OXFORD DEPARTMENT OF COMPUTER SCIENCE

DBOnto

SIRIUS

Computing Upper Bound — Example

$$\mathcal{O}' \begin{cases} \text{Doctor} \sqsubseteq \exists \text{treats}.\{P\} \\ \{P\} \sqsubseteq \text{Patient} \\ \exists \text{treats}.\text{Patient} \sqsubseteq \text{Doctor} \\ \text{Consulatant} \sqsubseteq \text{Doctor} \end{cases}$$

$$\text{DB} \begin{cases} \text{treats}(d_1, p_1) \\ \text{Patient}(p_1) \\ \text{Doctor}(d_2) \\ \text{Consultant}(c_1) \end{cases}$$

$$\text{DB}' \begin{cases} \text{treats}(d_1, p_1) \\ \text{Patient}(p_1) \\ \text{Doctor}(d_2) \\ \text{Consultant}(c_1) \\ \text{Patient}(P) \\ \text{Doctor}(d_1) \\ \text{Doctor}(c_1) \\ \text{treats}(d_1, P) \\ \text{treats}(d_2, P) \\ \text{treats}(c_1, P) \end{cases}$$

$$\mathcal{Q}_1 \quad Q(x) \leftarrow \text{Doctor}(y) \quad \leadsto \quad \{d_2, d_1, c_1\}$$

Computing Upper Bound — Example

$$\mathcal{O}' \begin{cases} \text{Doctor} \sqsubseteq \exists \text{treats}.\{P\} \\ \{\text{P}\} \sqsubseteq \text{Patient} \\ \exists \text{treats}.\text{Patient} \sqsubseteq \text{Doctor} \\ \text{Consulatant} \sqsubseteq \text{Doctor} \end{cases}$$

$$\text{DB} \begin{cases} \text{treats}(d_1, p_1) \\ \text{Patient}(p_1) \\ \text{Doctor}(d_2) \\ \text{Consultant}(c_1) \end{cases}$$

$$\text{DB}' \begin{cases} \text{treats}(d_1, p_1) \\ \text{Patient}(p_1) \\ \text{Doctor}(d_2) \\ \text{Consultant}(c_1) \\ \text{Patient}(P) \\ \text{Doctor}(d_1) \\ \text{Doctor}(c_1) \\ \text{treats}(d_1, P) \\ \text{treats}(d_2, P) \\ \text{treats}(c_1, P) \end{cases}$$

$$\mathcal{Q}_1 \quad Q(x) \leftarrow \text{Doctor}(y) \quad \rightsquigarrow \quad \{d_2, d_1, c_1\}$$

$$\mathcal{Q}_2 \quad Q(x) \leftarrow \text{treats}(x, y) \wedge \text{Patient}(y)$$

Computing Upper Bound — Example

$\mathcal{O}'$:

$$\text{Doctor} \sqsubseteq \exists\text{treats}.\{P\}$$
$$\{\text{P}\} \sqsubseteq \text{Patient}$$
$$\exists\text{treats}.\text{Patient} \sqsubseteq \text{Doctor}$$
$$\text{Consulatant} \sqsubseteq \text{Doctor}$$

DB:

- $\text{treats}(d_1, p_1)$
- $\text{Patient}(p_1)$
- $\text{Doctor}(d_2)$
- $\text{Consultant}(c_1)$

DB′:

- $\text{treats}(d_1, p_1)$
- $\text{Patient}(p_1)$
- $\text{Doctor}(d_2)$
- $\text{Consultant}(c_1)$
- $\text{Patient}(P)$
- $\text{Doctor}(d_1)$
- $\text{Doctor}(c_1)$
- $\text{treats}(d_1, P)$
- $\text{treats}(d_2, P)$
- $\text{treats}(c_1, P)$

$$\mathcal{Q}_1 \quad Q(x) \leftarrow \text{Doctor}(y) \quad \rightsquigarrow \quad \{d_2, d_1, c_1\}$$

$$\mathcal{Q}_2 \quad Q(x) \leftarrow \text{treats}(x, y) \wedge \text{Patient}(y) \quad \rightsquigarrow \quad \{d_1, d_2, c_1\}$$

Computing Lower and Upper Bounds

- RL reasonting w.r.t. $\mathcal{O}'$ gives (complete but unsound) upper bound answer ***U***

- If ***L*** = ***U***, then both answers are sound and complete
- If ***L*** ≠ ***U***, then ***U*** \ ***L*** identifies a (small) set of "possible" answers
  - Indicates range of uncertainty
  - Can (more efficiently) check possible answers using, e.g., HermiT
  - Can use ***U*** \ ***L*** to identify (small) "relevant" subset of data needed to efficiently compute exact answer

[1] Zhou et al. PAGOdA: Pay-as-you-go Ontology Query Answering Using a Datalog Reasoner. J. of Artificial Intelligence Research, 54:309-367, 2015.