Scalable Ontology-Based Information Systems

Ian Horrocks
<ian.horrocks@comlab.ox.ac.uk>
Information Systems Group
Oxford University Computing Laboratory
What is an Ontology?
What is an Ontology?

A model of (some aspect of) the world
What is an Ontology?

A model of (some aspect of) the world

- Introduces **vocabulary** relevant to domain, e.g.:
  - Anatomy
What is an Ontology?

A model of (some aspect of) the world

- Introduces **vocabulary** relevant to domain, e.g.:
  - Anatomy
  - Cellular biology
What is an Ontology?

A model of (some aspect of) the world

• Introduces **vocabulary** relevant to domain, e.g.:
  – Anatomy
  – Cellular biology
  – Aerospace
What is an Ontology?

A model of (some aspect of) the world

- Introduces **vocabulary** relevant to domain, e.g.:
  - Anatomy
  - Cellular biology
  - Aerospace
  - Dogs
What is an Ontology?

A model of (some aspect of) the world

- Introduces **vocabulary** relevant to domain, e.g.:
  - Anatomy
  - Cellular biology
  - Aerospace
  - Dogs
  - Hotdogs
  - …
What is an Ontology?

A model of (some aspect of) the world

- Introduces **vocabulary** relevant to domain
- Specifies **meaning** (semantics) of terms

Heart is a muscular organ that is part of the circulatory system
What is an Ontology?

A model of (some aspect of) the world

- **Introduces** **vocabulary** relevant to domain
- **Specifies** **meaning** (semantics) of terms
  
  Heart is a muscular organ that is part of the circulatory system

- **Formalised** using suitable logic

\[ \forall x. [\text{Heart}(x) \rightarrow \text{MuscularOrgan}(x) \land \exists y. [\text{isPartOf}(x, y) \land \text{CirculatorySystem}(y)]] \]
Web Ontology Language OWL (2)

- **W3C recommendation(s)**
- Motivated by **Semantic Web** activity
  Add meaning to web content by annotating it with terms defined in ontologies
- Supported by **tools and infrastructure**
  - APIs (e.g., OWL API, Thea, OWLink)
  - Development environments
    (e.g., Protégé, Swoop, TopBraid Composer, Neon)
  - Reasoners & Information Systems
    (e.g., Pellet, Racer, HermiT, Quonto, …)
- Based on **Description Logics (SHOIN / SROIQ)**
Description Logics (DLs)

• Fragments of first order logic designed for KR
• Desirable computational properties
  – Decidable (essential)
  – Low complexity (desirable)
• Succinct and variable free syntax

\[
\forall x. \left[ \text{Heart}(x) \rightarrow \text{MuscularOrgan}(x) \land \exists y. \left[ \text{isPartOf}(x, y) \land \right. \right.
\left. \left. \text{CirculatorySystem}(y) \right] \right]
\]

\text{Heart} \sqsubseteq \text{MuscularOrgan} \sqcap \exists \text{isPartOf. CirculatorySystem}
Description Logics (DLs)

DL Knowledge Base (KB) consists of two parts:

– Ontology (aka TBox) axioms define terminology (schema)

\[
\begin{align*}
\text{Heart} & \sqsubseteq \text{MuscularOrgan} \sqsubseteq \\
& \exists \text{isPartOf.} \text{CirculatorySystem} \\
\text{HeartDisease} & \equiv \text{Disease} \sqsubseteq \\
& \exists \text{affects.} \text{Heart} \\
\text{VascularDisease} & \equiv \text{Disease} \sqsubseteq \\
& \exists \text{affects.} (\exists \text{isPartOf.} \text{CirculatorySystem})
\end{align*}
\]

– Ground facts (aka ABox) use the terminology (data)

\[
\begin{align*}
\text{John} : \text{Patient} \sqsubseteq \\
& \exists \text{suffersFrom.} \text{HeartDisease}
\end{align*}
\]
Why Care About Semantics?
Why Care About Semantics?
Why Care About Semantics?

Why should I care about semantics?
Why Care About Semantics?

Why should I care about semantics?
Why Care About Semantics?

Why should I care about semantics?

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

Why should I care about semantics?

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.
Why Care About Semantics?

Why should I care about semantics?

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 and test (ontology-based) information systems we need to precisely define their intended behaviour.
What are Ontologies Good For?

• Coherent **user-centric view** of domain
  – Help identify and resolve disagreements

• Ontology-based **Information Systems**
  – View of data that is independent of logical/physical schema
  – Answers reflect schema & data, e.g.:
    “Patients suffering from Vascular Disease”

Now... *that* should clear up a few things around here
What are Ontologies Good For?

Heart ⊆ MuscularOrgan ⊆
  ∃isPartOf.CirculatorySystem
HeartDisease ≡ Disease ⊆
  ∃affects.Heart
VascularDisease ≡ Disease ⊆
  ∃affects.(∃isPartOf.CirculatorySystem)

John : Patient ⊆
  ∃suffersFrom.HeartDisease
What are Ontologies Good For?

• Coherent **user-centric view** of domain
  – Help identify and resolve disagreements

• Ontology-based **Information Systems**
  – View of data that is independent of logical/physical schema
  – Answers reflect schema & data, e.g.:
    “Patients suffering from Vascular Disease”
  – Query expansion/navigation/refinement
  – Incomplete and semi-structured data
  – Integration of heterogeneous sources

Now... *that* should clear up a few things around here
Information-Based Decisions

Increasingly critical in many areas:

• In Healthcare industry
  – Too much screening harms patients and wastes money
  – Too little screening costs lives
Information-Based Decisions

Increasingly critical in many areas:

• In Oil and Gas industry
  – Better quality information could add €1B/year net value to Statoil production
  – Poorer quality information and analysis costs €6M/weekend!
Information-Based Decisions

Increasingly critical in many areas:

• In IT industry
  – SAP deals with 80,000 queries/month at a cost of approx. €16M
  – SAP estimate 50% of support staff time spent searching for relevant information
Information-Based Decisions

Increasingly critical in many areas:

- In Transport Security
  - Failures can cost hundreds of lives

“We had sufficient information, but failed to integrate and understand it”
Analysis Bottleneck

- Decisions based on **information**
- **Integration and analysis** of data is the bottleneck
Healthcare

• UK NHS £10 billion “Connecting for Health” IT programme

• Key component is Care Records Service (CRS)
  – “Live, interactive patient record service accessible 24/7”
  – Patient data distributed across local centres in 5 regional clusters, and a national DB
  – SNOMED-CT ontology provides common vocabulary for data
    • Clinical data uses terms drawn from ontology
SNOMED-CT

• It’s **BIG** – over **400,000 concepts**
• Language used is **EL profile of OWL 2**
• **Multiple hierarchies** and **rich definitions**
Pulmonary Tuberculosis

kind of Pulmonary disease due to Mycobacteria

kind of tuberculosis

kind of pneumonitis

found in lung structure
What About Scalability?

• Only **useful in practice** if we can deal with large ontologies and/or large data sets

• Unfortunately, many ontology languages are highly intractable
  – OWL 2 satisfiability is **2NEXPTIME-complete** w.r.t. schema
  – and **NP-Hard** w.r.t. data (upper bound open)

• Problem addressed in practice by
  – Algorithms that work well in **typical cases**
  – Highly **optimised implementations**
  – Use of tractable fragments (aka **profiles**)
Tableau Reasoning Algorithms
Tableau Reasoning Algorithms

Standard technique based on (hyper-) tableau

- Reasoning tasks reducible to (un)satisfiability
  - E.g., $\text{KB} \nvdash \text{HeartDisease} \subseteq \text{VascularDisease}$ iff
  $\text{KB} \cup \{x:(\text{HeartDisease} \land \neg \text{VascularDisease})\}$ is not satisfiable
Tableau Reasoning Algorithms

Standard technique based on (hyper-) tableau

- Reasoning tasks reducible to (un)satisfiability
  
  • E.g., KB ⊨ HeartDisease ⊆ VascularDisease iff
    KB ∪ \{x: (HeartDisease ∧ ¬VascularDisease)\} is not satisfiable
  
- Algorithm tries to construct (an abstraction of) a model:

  \( x: \text{HeartDisease} \land \neg \text{VascularDisease} \)
Tableau Reasoning Algorithms

Standard technique based on (hyper-) tableau

- Reasoning tasks reducible to (un)satisfiability
  - E.g., KB ⊨ HeartDisease ⊆ VascularDisease iff
    KB ∪ {x:(HeartDisease ∨ ¬VascularDisease)} is not satisfiable
  - Algorithm tries to construct (an abstraction of) a model:

  $x : \text{HeartDisease} \land \neg \text{VascularDisease}$

  $x : \text{HeartDisease}$
Tableau Reasoning Algorithms

Standard technique based on (hyper-) tableau

- Reasoning tasks reducible to (un)satisfiability
  
  • E.g., KB ⊨ HeartDisease ⊆ VascularDisease iff
    KB ∪ \{x:(HeartDisease ∨ ¬VascularDisease)\} is not satisfiable

- Algorithm tries to construct (an abstraction of) a model:

  \( x: \text{HeartDisease} \) ∨ \( \neg\text{VascularDisease} \)
  \( x: \text{HeartDisease} \)
  \( x: \text{Disease} \)
  \( x: \exists \text{affects. Heart} \)
Tableau Reasoning Algorithms

Standard technique based on (hyper-) tableau

- Reasoning tasks reducible to (un)satisfiability
  - E.g., $KB \not\models \text{HeartDisease} \sqsubseteq \text{VascularDisease}$ iff $KB \cup \{x: (\text{HeartDisease} \land \neg \text{VascularDisease})\}$ is not satisfiable

- Algorithm tries to construct (an abstraction of) a model:

\[
\begin{align*}
x &: \text{HeartDisease} \land \neg \text{VascularDisease} \\
x &: \text{HeartDisease} \\
x &: \text{Disease} \\
x &: \exists \text{affects. Heart} \\
(x, y) &: \text{affects} \\
y &: \text{Heart}
\end{align*}
\]
Tableau Reasoning Algorithms

Standard technique based on (hyper-) tableau

– Reasoning tasks reducible to (un)satisfiability
  • E.g., \( KB \not\models \text{HeartDisease} \sqsubseteq \text{VascularDisease} \) iff
    \( KB \cup \{ x : (\text{HeartDisease} \sqcap \neg \text{VascularDisease}) \} \) is not satisfiable

– Algorithm tries to construct (an abstraction of) a model:

\[
\begin{align*}
x : & \text{HeartDisease} \sqcap \neg \text{VascularDisease} \\
x : & \text{HeartDisease} \\
x : & \text{Disease} \\
x : & \exists \text{affects.Heart} \\
(x, y) : & \text{affects} \\
y : & \text{Heart} \\
y : & \text{MuscularOrgan} \\
y : & \exists \text{isPartOf.CirculatorySystem}
\end{align*}
\]
Tableau Reasoning Algorithms

Standard technique based on (hyper-) tableau

- Reasoning tasks reducible to (un)satisfiability
  - E.g., KB \models HeartDisease ⊆ VascularDisease iff
    KB ∪ \{x:(HeartDisease ∧ ¬VascularDisease)\} is not satisfiable
  - Algorithm tries to construct (an abstraction of) a model:

\[
\begin{align*}
    x &\colon \text{HeartDisease} \quad \neg \text{VascularDisease} \\
    x &\colon \text{HeartDisease} \\
    x &\colon \text{Disease} \\
    x &\colon \exists \text{affects.Heart} \\
    (x, y) &\colon \text{affects} \\
    y &\colon \text{Heart} \\
    y &\colon \text{MuscularOrgan} \\
    y &\colon \exists \text{isPartOf.CirculatorySystem} \\
    (y, z) &\colon \text{isPartOf} \\
    z &\colon \text{CirculatorySystem}
\end{align*}
\]
Tableau Reasoning Algorithms

Standard technique based on (hyper-) tableau

– Reasoning tasks reducible to (un)satisfiability
  • E.g., KB ⊨ HeartDisease ⊆ VascularDisease iff
    KB ∪ {x:(HeartDisease ∨ ¬VascularDisease)} is not satisfiable
  – Algorithm tries to construct (an abstraction of) a model:

\[ x: \text{HeartDisease} \quad \neg x: \text{VascularDisease} \]
\[ x: \text{HeartDisease} \]
\[ x: \text{Disease} \]
\[ x: \exists y \text{ affects Heart} \]
\[ (x, y): \text{affects} \]
\[ y: \text{Heart} \]
\[ y: \text{MuscularOrgan} \]
\[ y: \exists z \text{ isPartOf CirculatorySystem} \]
\[ (y, z): \text{isPartOf} \]
\[ z: \text{CirculatorySystem} \]
Tableau Reasoning Algorithms

Standard technique based on (hyper-) tableau

- Reasoning tasks reducible to (un)satisfiability
  - E.g., $\text{KB} \not\models \text{HeartDisease} \sqsubseteq \text{VascularDisease}$ iff $\text{KB} \cup \{x:(\text{HeartDisease} \land \neg \text{VascularDisease})\}$ is not satisfiable

- Algorithm tries to construct (an abstraction of) a model:

\[
\begin{align*}
x &: \text{HeartDisease} \land \neg \text{VascularDisease} & x &: \neg \text{VascularDisease} \\
x &: \text{HeartDisease} & x &: \neg \text{Disease} \\
x &: \text{Disease} & \neg \exists \text{affects}.(\exists \text{isPartOf.CirculatorySystem}) \\
x &: \exists \text{affects}.\text{Heart} \\
(x, y) &: \text{affects} \\
y &: \text{Heart} \hspace{1cm} (y, z) &: \text{isPartOf} \\
y &: \text{MuscularOrgan} \\
y &: \exists \text{isPartOf.CirculatorySystem} \\
z &: \text{CirculatorySystem}
\end{align*}
\]
Tableau Reasoning Algorithms

Standard technique based on (hyper-) tableau

- Reasoning tasks reducible to (un)satisfiability
  - E.g., KB ⊨ HeartDisease ⊆ VascularDisease iff
    KB ∪ \{x:\text{HeartDisease} \land \neg \text{VascularDisease}\} is not satisfiable
  - Algorithm tries to construct (an abstraction of) a model:

\[
\begin{align*}
  x & : \text{HeartDisease} \land \neg \text{VascularDisease} & x & : \neg \text{VascularDisease} \\
  x & : \text{HeartDisease} & x & : \neg \text{Disease} \\
  x & : \text{Disease} & x & : \exists \text{affects.} (\exists \text{isPartOf. CirculatorySystem}) \\
  x & : \exists \text{affects. Heart} & x & : \neg \text{Disease} \\
  (x, y) & : \text{affects} & & \\
  y & : \text{Heart} & & \\
  y & : \text{MuscularOrgan} & & \\
  y & : \exists \text{isPartOf. CirculatorySystem} & & \\
  (y, z) & : \text{isPartOf} & & \\
  z & : \text{CirculatorySystem} & & 
\end{align*}
\]
Tableau Reasoning Algorithms

Standard technique based on (hyper-) tableau

- Reasoning tasks reducible to (un)satisfiability
  - E.g., KB \models \text{HeartDisease} \subseteq \text{VascularDisease} iff
    \[ \text{KB} \cup \{ x : (\text{HeartDisease} \land \neg \text{VascularDisease}) \} \text{ is not satisfiable} \]
  - Algorithm tries to construct (an abstraction of) a model:

\[
\begin{align*}
  x &: \text{HeartDisease} \land \neg \text{VascularDisease} & x &: \neg \text{VascularDisease} \\
  x &: \text{HeartDisease} & x &: \neg \text{Disease} \\
  x &: \text{Disease} & \neg \exists a \text{ffects.}(\exists \text{isPartOf. CirculatorySystem}) \\
  x &: \exists \text{ffects. Heart} & x &: \neg \text{Disease} \\
  (x, y) &: \text{ffects} & \neg \exists a \text{ffects.}(\exists \text{isPartOf. CirculatorySystem}) \\
  y &: \text{Heart} & x &: \neg \text{Disease} \\
  y &: \text{MuscularOrgan} & y &: \exists \text{isPartOf. CirculatorySystem} \\
  (y, z) &: \text{isPartOf} & x &: \neg \text{Disease} \\
  z &: \text{CirculatorySystem}
\end{align*}
\]
Tableau Reasoning Algorithms

Standard technique based on (hyper-) tableau

- Reasoning tasks reducible to (un)satisfiability
  - E.g., KB ⊨ HeartDisease ⊏ VascularDisease iff
    KB ∪ {x: (HeartDisease ∩ ¬VascularDisease)} is not satisfiable
  - Algorithm tries to construct (an abstraction of) a model

\[
\begin{align*}
x &: \text{HeartDisease} \land \neg \text{VascularDisease} & x &: \neg \text{VascularDisease} \\
x &: \text{HeartDisease} & x &: \neg \text{Disease} \lor \\
x &: \text{Disease} & \\
x &: \exists \text{affects.Heart} & \\
(x, y) &: \text{affects} \\
y &: \text{Heart} & \\
y &: \text{MuscularOrgan} & \\
y &: \exists \text{isPartOf.CirculatorySystem} & \\
(y, z) &: \text{isPartOf} \\
z &: \text{CirculatorySystem}
\end{align*}
\]
Tableau Reasoning Algorithms

Standard technique based on (hyper-) tableau

- Reasoning tasks reducible to (un)satisfiability
  
  • E.g., \( KB \models \text{HeartDisease} \sqsubseteq \text{VascularDisease} \) iff
    \( KB \cup \{ x : (\text{HeartDisease} \land \neg \text{VascularDisease}) \} \) is not satisfiable

- Algorithm tries to construct (an abstraction of) a model

\[
\begin{align*}
  x : & \text{HeartDisease} \land \neg \text{VascularDisease} & x : & \neg \text{VascularDisease} \\
  x : & \text{HeartDisease} & x : & \neg \text{Disease} \sqcup \\
  x : & \text{Disease} & & \\
  x : & \exists \text{affects}. \text{Heart} & & \\
  (x, y) : & \text{affects} & & \\
  y : & \text{Heart} & & \\
  y : & \text{MuscularOrgan} & & \\
  y : & \exists \text{isPartOf}. \text{CirculatorySystem} & & \\
  (y, z) : & \text{isPartOf} & & \\
  z : & \text{CirculatorySystem} & & 
\end{align*}
\]
Tableau Reasoning Algorithms

Standard technique based on (hyper-) tableau

- Reasoning tasks reducible to (un)satisfiability
  - E.g., KB ⊨ HeartDisease ⊆ VascularDisease iff
    KB ∪ \{x:(HeartDisease ∧ ¬VascularDisease)\} is not satisfiable
  - Algorithm tries to construct (an abstraction of) a model

\[
\begin{align*}
x &: \text{HeartDisease} \quad &x &: \neg\text{VascularDisease} \\
x &: \text{HeartDisease} \\
x &: \text{Disease} \\
x &: \exists \text{affects.Heart} \\
(x, y) &: \text{affects} \\
y &: \text{Heart} \\
y &: \text{MuscularOrgan} \\
y &: \exists \text{isPartOf.CirculatorySystem} \\
(y, z) &: \text{isPartOf} \\
z &: \text{CirculatorySystem}
\end{align*}
\]
Tableau Reasoning Algorithms

Standard technique based on (hyper-) tableau

- Reasoning tasks reducible to (un)satisfiability
  - E.g., KB ⊨ HeartDisease ⊆ VascularDisease iff
    KB ∪ \{x: (HeartDisease ∧ ¬VascularDisease)\} is not satisfiable
  - Algorithm tries to construct (an abstraction of) a model

\[
\begin{align*}
  x &: HeartDisease \quad \neg VascularDisease & \quad x &: \neg VascularDisease \\
  x &: HeartDisease & \quad x &: \neg Disease \cup \\
  x &: Disease & \quad \neg \exists affects. (\exists isPartOf. CirculatorySystem) \\
  x &: \exists affects. Heart & \quad x &: \neg \exists affects. (\exists isPartOf. CirculatorySystem) \\
  (x, y) &: affects & \quad x &: \forall affects. (\forall isPartOf. \neg CirculatorySystem) \\
  y &: Heart & \quad y &: \forall isPartOf. \neg CirculatorySystem \\
  y &: MuscularOrgan & \\
  y &: \exists isPartOf. CirculatorySystem \\
  (y, z) &: isPartOf & \\
  z &: CirculatorySystem
\end{align*}
\]
Tableau Reasoning Algorithms

Standard technique based on (hyper-) tableau

- Reasoning tasks reducible to (un) satisfiability
  - E.g., \( \text{KB} \models \text{HeartDisease} \sqsubseteq \text{VascularDisease} \) iff 
    \( \text{KB} \cup \{x: (\text{HeartDisease} \land \neg \text{VascularDisease})\} \) is not satisfiable
  - Algorithm tries to construct (an abstraction of) a model

\[
\begin{align*}
x : & \text{HeartDisease} \land \neg \text{VascularDisease} &
\quad x : \neg \text{VascularDisease} \\
x : & \text{HeartDisease} &
\quad x : \neg \text{Disease} \cup \\
x : & \text{Disease} &
\quad \neg \exists \text{affects} \cdot (\exists \text{isPartOf}. \text{CirculatorySystem}) \\
x : & \exists \text{affects}. \text{Heart} &
\quad x : \neg \exists \text{affects} \cdot (\exists \text{isPartOf}. \text{CirculatorySystem}) \\
(x, y) : & \text{affects} &
\quad x : \forall \text{affects} \cdot (\forall \text{isPartOf}. \neg \text{CirculatorySystem}) \\
y : & \text{Heart} &
\quad y : \exists \text{isPartOf}. \text{CirculatorySystem} \\
y : & \text{MuscularOrgan} &
\quad z : \neg \text{CirculatorySystem} \\
y : & \exists \text{isPartOf}. \text{CirculatorySystem} \\
(y, z) : & \text{isPartOf} &
\quad z : \neg \text{CirculatorySystem}
\end{align*}
\]
Tableau Reasoning Algorithms

Standard technique based on (hyper-) **tableau**

- Reasoning tasks reducible to (un)**satisfiability**
  
  • E.g., \( \text{KB} \not\models \text{HeartDisease} \subseteq \text{VascularDisease} \) iff 
  \( \text{KB} \cup \{ \forall x : (\text{HeartDisease} \land \neg \text{VascularDisease}) \} \) is not satisfiable
  
  - Algorithm tries to construct (an abstraction of) a model

\[
\begin{align*}
    x &: \text{HeartDisease} \land \neg \text{VascularDisease} \\
    x &: \text{HeartDisease} \\
    x &: \text{Disease} \\
    x &: \exists \text{affects.Heart} \\
    x, y &: \text{affects} \\
    y &: \text{Heart} \\
    y &: \text{MuscularOrgan} \\
    y &: \exists \text{isPartOf.CirculatorySystem} \\
    y, z &: \text{isPartOf} \\
    z &: \text{CirculatorySystem} \\
    x &: \neg \text{VascularDisease} \\
    x &: \neg \text{Disease} \\
    x &: \neg \exists \text{affects.(}\exists \text{isPartOf.CirculatorySystem)} \\
    x &: \forall \text{affects.(}\forall \text{isPartOf.}\neg \text{CirculatorySystem)} \\
    y &: \forall \text{isPartOf.}\neg \text{CirculatorySystem} \\
    z &: \neg \text{CirculatorySystem}
\end{align*}
\]
Tableau Reasoning Algorithms

Standard technique based on (hyper-) tableau

- Reasoning tasks reducible to (un)satisfiability
  - E.g., $\text{KB} \models \text{HeartDisease} \sqsubseteq \text{VascularDisease}$ iff
    $\text{KB} \cup \{x: (\text{HeartDisease} \land \neg \text{VascularDisease})\}$ is not satisfiable
  - Algorithm tries to construct (an abstraction of) a model

Note similarity to chase!

$x: \text{HeartDisease} \land \neg \text{VascularDisease} \quad x: \neg \text{VascularDisease}$
$x: \text{HeartDisease} \quad x: \neg \text{Disease}$
$x: \exists \text{affects} \cdot \exists \text{Heart} \quad x: \neg \exists \text{affects} \cdot \exists \text{isPartOf. CirculatorySystem}$
$(x, y): \text{affects} \quad \exists \text{affects} \cdot (\exists \text{isPartOf. CirculatorySystem})$
$y: \text{Heart} \quad y: \neg \exists \text{isPartOf. CirculatorySystem}$
$y: \text{MuscularOrgan} \quad y: \text{VisPartOf. CirculatorySystem}$
$(y, z): \text{isPartOf} \quad z: \text{isPartOf. CirculatorySystem}$
$z: \text{CirculatorySystem}$
Highly Optimised Implementations

• Lazy unfolding
• Simplification and rewriting,
  \[ A \cap B \subseteq C \quad \rightarrow \quad A \subseteq C \cup \neg B \]
• HyperTableau (reduces non-determinism)
• Fast semi-decision procedures
• Search optimisations
• Reuse of previous computations
• Heuristics

Not computationally optimal, but effective with many realistic ontologies
Scalability Issues

• Problems with very large and/or cyclical ontologies

• Ontologies may define 10s/100s of thousands of terms

• Can lead to construction of very large models
Scalability Issues

• Problems with large data sets (ABoxes)
  – Main reasoning problem is (conjunctive) query answering, e.g., retrieve all patients suffering from vascular disease:
    \[ Q(x) \leftarrow \text{Patient}(x) \land \text{suffersFrom}(x, y) \land \text{VascularDisease}(y) \]
  – Decidability still open for OWL, although minor restrictions (on cycles in non-distinguished variables) restore decidability
  – Query answering reduced to standard decision problem, e.g., by checking for each individual \( x \) if \( \text{KB} \models Q(x) \)
  – Model construction starts with all ground facts (data)

• Typical applications may use data sets with 10s/100s of millions of individuals (or more)
OWL 2 Profiles

• OWL recommendation now updated to **OWL 2**
• OWL 2 defines several **profiles** – fragments with desirable computational properties
  – **OWL 2 EL** targeted at very large ontologies
  – **OWL 2 QL** targeted at very large data sets
OWL 2 EL

- A (near maximal) fragment of OWL 2 such that
  - Satisfiability checking is in PTime (PTime-Complete)
  - Data complexity of query answering also PTime-Complete

- Based on \textit{EL} family of description logics

- Can exploit saturation based reasoning techniques
  - Computes classification in “one pass”
  - Computationally optimal
  - Can be extended to Horn fragment of OWL DL
Saturation-based Technique (basics)

• Normalise ontology axioms to standard form:

\[
A \subseteq B \quad A \cap B \subseteq C \quad A \subseteq \exists R.B \quad \exists R.B \subseteq C
\]

• Saturate using inference rules:

\[
\frac{A \subseteq B \quad B \subseteq C}{A \subseteq C}
\]

\[
\frac{A \subseteq B \quad A \subseteq C \quad B \cap C \subseteq D}{A \subseteq D}
\]

\[
\frac{A \subseteq \exists R.B \quad B \subseteq C \quad \exists R.C \subseteq D}{A \subseteq D}
\]

• Extension to Horn fragment requires (many) more rules
Saturation-based Technique (basics)

Example:

\[
\begin{align*}
\text{OrganTransplant} & \equiv \text{Transplant} \sqcap \exists \text{site. Organ} \\
\text{HeartTransplant} & \equiv \text{Transplant} \sqcap \exists \text{site. Heart} \\
\text{Heart} & \sqsubseteq \text{Organ}
\end{align*}
\]
Saturation-based Technique (basics)

Example:

\[
\text{Organ Transplant} \equiv \text{Transplant} \cap \exists \text{site. Organ} \\
\text{Heart Transplant} \equiv \text{Transplant} \cap \exists \text{site. Heart} \\
\text{Heart} \subseteq \text{Organ}
\]
Saturation-based Technique (basics)

Example:

\[ \text{OrganTransplant} \equiv \text{Transplant} \land \exists \text{site.Organ} \]
\[ \text{HeartTransplant} \equiv \text{Transplant} \land \exists \text{site.Heart} \]
\[ \text{Heart} \subseteq \text{Organ} \]

\[ \text{OrganTransplant} \subseteq \text{Transplant} \]
\[ \text{OrganTransplant} \subseteq \exists \text{site.Organ} \]
Saturation-based Technique (basics)

Example:

\[
\text{OrganTransplant} \equiv \text{Transplant} \sqcap \exists \text{site. Organ}
\]
\[
\text{HeartTransplant} \equiv \text{Transplant} \sqcap \exists \text{site. Heart}
\]
\[
\text{Heart} \sqsubseteq \text{Organ}
\]

\[
\text{OrganTransplant} \sqsubseteq \text{Transplant}
\]
\[
\text{OrganTransplant} \sqsubseteq \exists \text{site. Organ}
\]
\[
\exists \text{site. Organ} \sqsubseteq \text{SO}
\]
\[
\text{Transplant} \sqcap \text{SO} \sqsubseteq \text{OrganTransplant}
\]
Saturation-based Technique (basics)

Example:

\[
\begin{align*}
\text{OrganTransplant} & \equiv \text{Transplant} \land \exists \text{site}.\text{Organ} \\
\text{HeartTransplant} & \equiv \text{Transplant} \land \exists \text{site}.\text{Heart} \\
\text{Heart} & \subseteq \text{Organ} \\
\text{OrganTransplant} & \subseteq \text{Transplant} \\
\text{OrganTransplant} & \subseteq \exists \text{site}.\text{Organ} \\
\exists \text{site}.\text{Organ} & \subseteq \text{SO} \\
\text{Transplant} \land \text{SO} & \subseteq \text{OrganTransplant}
\end{align*}
\]
Saturation-based Technique (basics)

Example:

\[
\text{OrganTransplant} \equiv \text{Transplant} \sqcap \exists \text{site.} \text{Organ} \\
\text{HeartTransplant} \equiv \text{Transplant} \sqcap \exists \text{site.} \text{Heart} \\
\quad \text{Heart} \sqsubseteq \text{Organ}
\]

\[
\text{OrganTransplant} \sqsubseteq \text{Transplant} \\
\text{OrganTransplant} \sqsubseteq \exists \text{site.} \text{Organ} \\
\quad \exists \text{site.} \text{Organ} \sqsubseteq \text{SO} \\
\text{Transplant} \sqcap \text{SO} \sqsubseteq \text{OrganTransplant} \\
\text{HeartTransplant} \sqsubseteq \text{Transplant} \\
\text{HeartTransplant} \sqsubseteq \exists \text{site.} \text{Heart} \\
\quad \exists \text{site.} \text{Heart} \sqsubseteq \text{SH} \\
\text{Transplant} \sqcap \text{SH} \sqsubseteq \text{HeartTransplant}
\]
Saturation-based Technique (basics)

Example:

\[
\begin{align*}
\text{OrganTransplant} & \equiv \text{Transplant} \lor \exists \text{site.Organ} \\
\text{HeartTransplant} & \equiv \text{Transplant} \lor \exists \text{site.Heart} \\
\text{Heart} & \subseteq \text{Organ} \\
\end{align*}
\]

\[
\begin{align*}
\text{OrganTransplant} & \subseteq \text{Transplant} \\
\text{OrganTransplant} & \subseteq \exists \text{site.Organ} \\
\exists \text{site.Organ} & \subseteq \text{SO} \\
\text{Transplant} \lor \text{SO} & \subseteq \text{OrganTransplant} \\
\text{HeartTransplant} & \subseteq \text{Transplant} \\
\text{HeartTransplant} & \subseteq \exists \text{site.Heart} \\
\exists \text{site.Heart} & \subseteq \text{SH} \\
\text{Transplant} \lor \text{SH} & \subseteq \text{HeartTransplant} \\
\end{align*}
\]
Saturation-based Technique (basics)

Example:

\[ \text{OrganTransplant} \equiv \text{Transplant} \sqcap \exists \text{site.Organ} \]
\[ \text{HeartTransplant} \equiv \text{Transplant} \sqcap \exists \text{site.Heart} \]
\[ \text{Heart} \sqsubseteq \text{Organ} \]

\[ \text{OrganTransplant} \sqsubseteq \text{Transplant} \]
\[ \text{OrganTransplant} \sqsubseteq \exists \text{site.Organ} \]
\[ \exists \text{site.Organ} \sqsubseteq \text{SO} \]
\[ \text{Transplant} \sqcap \text{SO} \sqsubseteq \text{OrganTransplant} \]
\[ \text{HeartTransplant} \sqsubseteq \text{Transplant} \]
\[ \text{HeartTransplant} \sqsubseteq \exists \text{site.Heart} \]
\[ \exists \text{site.Heart} \sqsubseteq \text{SH} \]
\[ \text{Transplant} \sqcap \text{SH} \sqsubseteq \text{HeartTransplant} \]
\[ \text{Heart} \sqsubseteq \text{Organ} \]
Saturation-based Technique (basics)

Example:

\[\text{OrganTransplant} \equiv \text{Transplant} \bigcap \exists \text{site.Organ}\]
\[\text{HeartTransplant} \equiv \text{Transplant} \bigcap \exists \text{site.Heart}\]
\[\text{Heart} \subseteq \text{Organ}\]

\[\text{OrganTransplant} \subseteq \text{Transplant}\]
\[\text{OrganTransplant} \subseteq \exists \text{site.Organ}\]
\[\exists \text{site.Organ} \subseteq \text{SO}\]
\[\text{Transplant} \bigcap \text{SO} \subseteq \text{OrganTransplant}\]
\[\text{HeartTransplant} \subseteq \text{Transplant}\]
\[\text{HeartTransplant} \subseteq \exists \text{site.Heart}\]
\[\exists \text{site.Heart} \subseteq \text{SH}\]
\[\text{Transplant} \bigcap \text{SH} \subseteq \text{HeartTransplant}\]
\[\text{Heart} \subseteq \text{Organ}\]

\[A \subseteq \exists R.B \quad B \subseteq C \quad \exists R.C \subseteq D\]
\[A \subseteq D\]
Saturation-based Technique (basics)

Example:

\[
\begin{align*}
\text{OrganTransplant} & \equiv \text{Transplant} \land \exists \text{site.Organ} \\
\text{HeartTransplant} & \equiv \text{Transplant} \land \exists \text{site.Heart} \\
\text{Heart} & \subseteq \text{Organ}
\end{align*}
\]

\[
\begin{align*}
A \subseteq \exists R.B & \quad B \subseteq C & \quad \exists R.C \subseteq D \\
A \subseteq D
\end{align*}
\]

\[
\begin{align*}
\text{OrganTransplant} & \subseteq \text{Transplant} \\
\exists \text{site.Organ} & \subseteq \text{SO} \\
\text{Transplant} \land \text{SO} & \subseteq \text{OrganTransplant} \\
\text{HeartTransplant} & \subseteq \text{Transplant} \\
\text{HeartTransplant} & \subseteq \exists \text{site.Heart} \\
\exists \text{site.Heart} & \subseteq \text{SH} \\
\text{Transplant} \land \text{SH} & \subseteq \text{HeartTransplant} \\
\text{Heart} & \subseteq \text{Organ}
\end{align*}
\]

\[
\text{HeartTransplant} \subseteq \text{SO}
\]
Saturation-based Technique (basics)

Example:

\[
\begin{align*}
\text{OrganTransplant} & \equiv \text{Transplant} \cap \exists \text{site.Organ} \\
\text{HeartTransplant} & \equiv \text{Transplant} \cap \exists \text{site.Heart} \\
& \quad \text{Heart} \subseteq \text{Organ} \\
\end{align*}
\]

\[
\begin{align*}
\text{OrganTransplant} & \sqsubseteq \text{Transplant} \\
\text{OrganTransplant} & \sqsubseteq \exists \text{site.Organ} \\
& \quad \exists \text{site.Organ} \sqsubseteq \text{SO} \\
\text{Transplant} \cap \text{SO} & \sqsubseteq \text{OrganTransplant} \\
\text{HeartTransplant} & \sqsubseteq \text{Transplant} \\
\text{HeartTransplant} & \sqsubseteq \exists \text{site.Heart} \\
& \quad \exists \text{site.Heart} \sqsubseteq \text{SH} \\
\text{Transplant} \cap \text{SH} & \sqsubseteq \text{HeartTransplant} \\
& \quad \text{Heart} \subseteq \text{Organ} \\
\end{align*}
\]
Saturation-based Technique (basics)

Example:

\[
\begin{align*}
\text{OrganTransplant} & \equiv \text{Transplant} \land \exists \text{site.Organ} \\
\text{HeartTransplant} & \equiv \text{Transplant} \land \exists \text{site.Heart} \\
\text{Heart} & \subseteq \text{Organ} \\
\text{OrganTransplant} & \subseteq \text{Transplant} \\
\text{OrganTransplant} & \subseteq \exists \text{site.Organ} \\
\exists \text{site.Organ} & \subseteq \text{SO} \\
\text{Transplant} \land \text{SO} & \subseteq \text{OrganTransplant} \\
\text{HeartTransplant} & \subseteq \text{Transplant} \\
\text{HeartTransplant} & \subseteq \exists \text{site.Heart} \\
\exists \text{site.Heart} & \subseteq \text{SH} \\
\text{Transplant} \land \text{SH} & \subseteq \text{HeartTransplant} \\
\text{Heart} & \subseteq \text{Organ}
\end{align*}
\]

\[
A \subseteq B \quad A \subseteq C \quad B \cap C \subseteq D \\
A \subseteq D
\]

\[
\text{HeartTransplant} \subseteq \text{SO} \\
\text{HeartTransplant} \subseteq \text{OrganTransplant}
\]
## Saturation-based Technique

Performance with large bio-medical ontologies:

<table>
<thead>
<tr>
<th></th>
<th>GO</th>
<th>NCI</th>
<th>Galen v.0</th>
<th>Galen v.7</th>
<th>SNOMED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concepts</td>
<td>20465</td>
<td>27652</td>
<td>2748</td>
<td>23136</td>
<td>389472</td>
</tr>
<tr>
<td>FACT++</td>
<td>15.24</td>
<td>6.05</td>
<td>465.35</td>
<td>—</td>
<td>650.37</td>
</tr>
<tr>
<td>HERMIT</td>
<td>199.52</td>
<td>169.47</td>
<td>45.72</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>PELLET</td>
<td>72.02</td>
<td>26.47</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>CEL</td>
<td>1.84</td>
<td>5.76</td>
<td>—</td>
<td>—</td>
<td>1185.70</td>
</tr>
<tr>
<td>CB</td>
<td>1.17</td>
<td>3.57</td>
<td>0.32</td>
<td>9.58</td>
<td>49.44</td>
</tr>
<tr>
<td>Speed-Up</td>
<td>1.57X</td>
<td>1.61X</td>
<td>143X</td>
<td>∞</td>
<td>13.15X</td>
</tr>
</tbody>
</table>
OWL 2 QL

- A (near maximal) fragment of OWL 2 such that
  - Data complexity of conjunctive query answering in \( AC^0 \)
- Based on DL-Lite family of description logics
- Can exploit query rewriting based reasoning technique
  - Computationally optimal
  - Data storage and query evaluation can be delegated to standard RDBMS
  - Can be extended to more expressive languages (beyond \( AC^0 \)) by delegating query answering to a Datalog engine
Query Rewriting Technique (basics)

- Given ontology $\mathcal{O}$ and query $Q$, use $\mathcal{O}$ to rewrite $Q$ as $Q'$ s.t., for any set of ground facts $\mathcal{A}$:
  - $\text{ans}(Q, \mathcal{O}, \mathcal{A}) = \text{ans}(Q', \emptyset, \mathcal{A})$
Query Rewriting Technique (basics)

- Given ontology $\mathcal{O}$ and query $Q$, use $\mathcal{O}$ to rewrite $Q$ as $Q'$ s.t., for any set of ground facts $A$:
  - $\text{ans}(Q, \mathcal{O}, A) = \text{ans}(Q', \emptyset, A)$

- Use (GAV) mapping $\mathcal{M}$ to map $Q'$ to SQL query
Query Rewriting Technique (basics)

- Given ontology $\mathcal{O}$ and query $Q$, use $\mathcal{O}$ to rewrite $Q$ as $Q'$ s.t., for any set of ground facts $\mathcal{A}$:
  - $\text{ans}(Q, \mathcal{O}, \mathcal{A}) = \text{ans}(Q', \emptyset, \mathcal{A})$
- Use (GAV) mapping $\mathcal{M}$ to map $Q'$ to SQL query
Query Rewriting Technique (basics)

- Given ontology $\mathcal{O}$ and query $Q$, use $\mathcal{O}$ to rewrite $Q$ as $Q'$ s.t., for any set of ground facts $\mathcal{A}$:
  - $\text{ans}(Q, \mathcal{O}, \mathcal{A}) = \text{ans}(Q', \emptyset, \mathcal{A})$

- Use (GAV) mapping $\mathcal{M}$ to map $Q'$ to SQL query

- Resolution based query rewriting
  - **Clausify** ontology axioms
  - **Saturate** (clausified) ontology and query using resolution
  - **Prune** redundant query clauses
Query Rewriting Technique (basics)

• Example:

\[ Q(x) \leftarrow \text{treats}(x, y) \land \text{Patient}(y) \]
Query Rewriting Technique (basics)

• Example:

\[
\begin{align*}
\text{Doctor} & \sqsubseteq \exists \text{treats. Patient} \\
\text{Consultant} & \sqsubseteq \text{Doctor} \\
\text{treats}(x, f(x)) & \leftarrow \text{Doctor}(x) \\
\text{Patient}(f(x)) & \leftarrow \text{Doctor}(x) \\
\text{Doctor}(x) & \leftarrow \text{Consultant}(x) \\
Q(x) & \leftarrow \text{treats}(x, y) \land \text{Patient}(y)
\end{align*}
\]
Query Rewriting Technique (basics)

• Example:

\[
\text{Doctor} \sqsubseteq \exists \text{treats}.\text{Patient} \\
\text{Consultant} \sqsubseteq \text{Doctor}
\]

\[
\begin{align*}
\text{treats}(x, f(x)) & \leftarrow \text{Doctor}(x) \\
\text{Patient}(f(x)) & \leftarrow \text{Doctor}(x) \\
\text{Doctor}(x) & \leftarrow \text{Consultant}(x) \\
\end{align*}
\]

\[
Q(x) \leftarrow \text{treats}(x, y) \land \text{Patient}(y)
\]
Query Rewriting Technique (basics)

• Example:

\[
\begin{align*}
\text{Doctor} & \sqsubseteq \exists \text{treats. Patient} \\
\text{Consultant} & \sqsubseteq \text{Doctor} \\
\text{treats}(x, f(x)) & \leftarrow \text{Doctor}(x) \\
\text{Patient}(f(x)) & \leftarrow \text{Doctor}(x) \\
\text{Doctor}(x) & \leftarrow \text{Consultant}(x) \\
Q(x) & \leftarrow \text{treats}(x, y) \land \text{Patient}(y) \\
Q(x) & \leftarrow \text{Doctor}(x) \land \text{Patient}(f(x))
\end{align*}
\]
Query Rewriting Technique (basics)

- Example:

\[
\begin{align*}
\text{Doctor} & \sqsubseteq \exists \text{treats. Patient} \\
\text{Consultant} & \sqsubseteq \text{Doctor}
\end{align*}
\]

\[
\begin{align*}
\text{treats}(x, f(x)) & \leftarrow \text{Doctor}(x) \\
\text{Patient}(f(x)) & \leftarrow \text{Doctor}(x) \\
\text{Doctor}(x) & \leftarrow \text{Consultant}(x)
\end{align*}
\]

\[
\begin{align*}
Q(x) & \leftarrow \text{treats}(x, y) \land \text{Patient}(y) \\
Q(x) & \leftarrow \text{Doctor}(x) \land \text{Patient}(f(x))
\end{align*}
\]
Query Rewriting Technique (basics)

• Example:

\[
\text{Doctor} \sqsubseteq \exists \text{treats.Patient} \\
\text{Consultant} \sqsubseteq \text{Doctor}
\]

\[
\text{treats}(x, f(x)) \leftarrow \text{Doctor}(x) \\
\text{Patient}(f(x)) \leftarrow \text{Doctor}(x) \\
\text{Doctor}(x) \leftarrow \text{Consultant}(x)
\]

\[
Q(x) \leftarrow \text{treats}(x, y) \land \text{Patient}(y) \\
Q(x) \leftarrow \text{Doctor}(x) \land \text{Patient}(f(x)) \\
Q(x) \leftarrow \text{treats}(x, f(x)) \land \text{Doctor}(x)
\]
Query Rewriting Technique (basics)

• Example:

\[
\begin{align*}
\text{Doctor} & \subseteq \exists \text{treats.Patient} \\
\text{Consultant} & \subseteq \text{Doctor} \\
\text{treats}(x, f(x)) & \leftarrow \text{Doctor}(x) \\
\text{Patient}(f(x)) & \leftarrow \text{Doctor}(x) \\
\text{Doctor}(x) & \leftarrow \text{Consultant}(x) \\
Q(x) & \leftarrow \text{treats}(x, y) \land \text{Patient}(y) \\
Q(x) & \leftarrow \text{Doctor}(x) \land \text{Patient}(f(x)) \\
Q(x) & \leftarrow \text{treats}(x, f(x)) \land \text{Doctor}(x)
\end{align*}
\]
Query Rewriting Technique (basics)

- Example:

\[
\begin{align*}
\text{Doctor} & \subseteq \exists \text{treats.Patient} \\
\text{Consultant} & \subseteq \text{Doctor}
\end{align*}
\]

\[
\begin{align*}
treats(x, f(x)) & \leftarrow \text{Doctor}(x) \\
\text{Patient}(f(x)) & \leftarrow \text{Doctor}(x) \\
\text{Doctor}(x) & \leftarrow \text{Consultant}(x)
\end{align*}
\]

\[
\begin{align*}
Q(x) & \leftarrow \text{treats}(x, y) \land \text{Patient}(y) \\
Q(x) & \leftarrow \text{Doctor}(x) \land \text{Patient}(f(x)) \\
Q(x) & \leftarrow \text{treats}(x, f(x)) \land \text{Doctor}(x) \\
Q(x) & \leftarrow \text{Doctor}(x)
\end{align*}
\]
Query Rewriting Technique (basics)

• Example:

\[
\begin{align*}
\text{Doctor} & \subseteq \exists \text{treats.Patient} \\
\text{Consultant} & \subseteq \text{Doctor} \\
\text{treats}(x, f(x)) & \leftarrow \text{Doctor}(x) \\
\text{Patient}(f(x)) & \leftarrow \text{Doctor}(x) \\
\text{Doctor}(x) & \leftarrow \text{Consultant}(x) \\
Q(x) & \leftarrow \text{treats}(x, y) \land \text{Patient}(y) \\
Q(x) & \leftarrow \text{Doctor}(x) \land \text{Patient}(f(x)) \\
Q(x) & \leftarrow \text{treats}(x, f(x)) \land \text{Doctor}(x) \\
Q(x) & \leftarrow \text{Doctor}(x)
\end{align*}
\]
Query Rewriting Technique (basics)

• Example:

\[ \text{Doctor} \sqsubseteq \exists \text{treats.Patient} \]
\[ \text{Consultant} \sqsubseteq \text{Doctor} \]

\[ \text{treats}(x, f(x)) \leftarrow \text{Doctor}(x) \]
\[ \text{Patient}(f(x)) \leftarrow \text{Doctor}(x) \]
\[ \text{Doctor}(x) \leftarrow \text{Consultant}(x) \]

\[ Q(x) \leftarrow \text{treats}(x, y) \land \text{Patient}(y) \]
\[ Q(x) \leftarrow \text{Doctor}(x) \land \text{Patient}(f(x)) \]
\[ Q(x) \leftarrow \text{treats}(x, f(x)) \land \text{Doctor}(x) \]
\[ Q(x) \leftarrow \text{Doctor}(x) \]
\[ Q(x) \leftarrow \text{Consultant}(x) \]
Query Rewriting Technique (basics)

• Example:

\[
\begin{align*}
\text{Doctor} & \sqsubseteq \exists \text{treats} \cdot \text{Patient} \\
\text{Consultant} & \sqsubseteq \text{Doctor} \\
\text{treats}(x, f(x)) & \leftarrow \text{Doctor}(x) \\
\text{Patient}(f(x)) & \leftarrow \text{Doctor}(x) \\
\text{Doctor}(x) & \leftarrow \text{Consultant}(x)
\end{align*}
\]

\[
\begin{align*}
Q(x) & \leftarrow \text{treats}(x, y) \land \text{Patient}(y) \\
Q(x) & \leftarrow \text{Doctor}(x) \land \text{Patient}(f(x)) \\
Q(x) & \leftarrow \text{treats}(x, f(x)) \land \text{Doctor}(x) \\
Q(x) & \leftarrow \text{Doctor}(x) \\
Q(x) & \leftarrow \text{Consultant}(x)
\end{align*}
\]
Query Rewriting Technique (basics)

- Example:

\[
\text{Doctor} \subseteq \exists \text{treats}. \text{Patient} \\
\text{Consultant} \subseteq \text{Doctor}
\]

\[
\begin{align*}
\text{treats}(x, f(x)) & \leftarrow \text{Doctor}(x) \\
\text{Patient}(f(x)) & \leftarrow \text{Doctor}(x) \\
\text{Doctor}(x) & \leftarrow \text{Consultant}(x)
\end{align*}
\]

\[
\begin{align*}
Q(x) & \leftarrow \text{treats}(x, y) \land \text{Patient}(y) \\
Q(x) & \leftarrow \text{Doctor}(x) \land \text{Patient}(f(x)) \\
Q(x) & \leftarrow \text{treats}(x, f(x)) \land \text{Doctor}(x) \\
Q(x) & \leftarrow \text{Doctor}(x) \\
Q(x) & \leftarrow \text{Consultant}(x)
\end{align*}
\]
Query Rewriting Technique (basics)

• Example:

\[
\begin{align*}
\text{Doctor} & \sqsubseteq \exists \text{treats}.\text{Patient} \\
\text{Consultant} & \sqsubseteq \text{Doctor}
\end{align*}
\]

\[
\begin{align*}
t\text{reats}(x, f(x)) & \leftarrow \text{Doctor}(x) \\
\text{Patient}(f(x)) & \leftarrow \text{Doctor}(x) \\
\text{Doctor}(x) & \leftarrow \text{Consultant}(x)
\end{align*}
\]

\[
\begin{align*}
Q(x) & \leftarrow \text{treats}(x, y) \land \text{Patient}(y) \\
Q(x) & \leftarrow \text{Doctor}(x) \land \text{Patient}(f(x)) \\
Q(x) & \leftarrow \text{treats}(x, f(x)) \land \text{Doctor}(x) \\
Q(x) & \leftarrow \text{Doctor}(x) \\
Q(x) & \leftarrow \text{Consultant}(x)
\end{align*}
\]

• For DL-Lite, result is a union of conjunctive queries

\[
Q(x) \leftarrow (\text{treats}(x, y) \land \text{Patient}(y)) \lor \text{Doctor}(x) \lor \text{Consultant}(x)
\]
Query Rewriting Technique (basics)

- Data can be stored/left in **RDBMS**
- Relationship between ontology and DB defined by **mappings**, e.g.:
  
  - `Doctor` ↔ `SELECT Name FROM Doctor`
  - `Patient` ↔ `SELECT Name FROM Patient`
  - `treats` ↔ `SELECT DName, PName FROM Treats`
Query Rewriting Technique (basics)

- Data can be stored/leave in **RDBMS**
- Relationship between ontology and DB defined by **mappings**, e.g.:
  
  Doctor $\mapsto$ SELECT Name FROM Doctor  
  Patient $\mapsto$ SELECT Name FROM Patient  
  treats $\mapsto$ SELECT DName, PName FROM Treats

- UCQ translated into **SQL query**:

  $$Q(x) \leftarrow (\text{treats}(x, y) \land \text{Patient}(y)) \lor \text{Doctor}(x) \lor \text{Consultant}(x)$$

  $\Downarrow$

  SELECT Name FROM Doctor UNION  
  SELECT DName FROM Treats, Patient WHERE PName = Name
Problems & Research Challenges

• Combining best features of DLs & DBs
  – In particular, integrating OWA and CWA

• Hard to find a coherent semantic framework
  – Problems mainly due to existential quantifiers: should existentially implied objects be considered different?
    • Does a person owning a phone and an ipod own 2 things?
    • Does a person owning a phone and an iphone own 2 things?
    • Does a person owning a phone and a phone own 2 things?

• Interesting ideas emerging in DL & DB communities, e.g.:
Problems & Research Challenges

- Open questions w.r.t. query rewriting
Problems & Research Challenges

• Open questions w.r.t. query rewriting
  – Currently only for very weak ontology languages
Problems & Research Challenges

• Open questions w.r.t. query rewriting
  – Currently only for very weak ontology languages
  – Even for these languages, queries can get very large (order \(|\mathcal{O}| \cdot |Q|^{|\mathcal{O}|}\), and existing RDBMSs may behave poorly
    • Not clear if this will be a problem in practice, see, e.g., Savo et al. MASTRO at Work: Experiences on Ontology-based Data Access. DL 2010.
Problems & Research Challenges

- Open questions w.r.t. query rewriting
  - Currently only for very weak ontology languages
  - Even for these languages, queries can get very large (order $|O| \cdot |Q|^{O}$), and existing RDBMSs may behave poorly
    - Not clear if this will be a problem in practice, see, e.g., Savo et al. MASTRO at Work: Experiences on Ontology-based Data Access. DL 2010.
    - Larger fragments require (at least) Datalog engines and/or extension to technique (e.g., partial materialisation)
      - Promising new work in this area, see, e.g., Lutz et al. Conjunctive Query Answering in the Description Logic EL Using a Relational Database System. IJCAI 2009.
Problems & Research Challenges

• Infrastructure
Problems & Research Challenges

• Infrastructure
  – Standardised query language
    • SPARQL standard for RDF
    • Currently being extended for OWL, see http://www.w3.org/2009/sparql/wiki/Main_Page
Problems & Research Challenges

• Infrastructure
  – Standardised query language
    • SPARQL standard for RDF
    • Currently being extended for OWL, see http://www.w3.org/2009/sparql/wiki/Main_Page
  – Privacy and information hiding
    • May want to keep parts of data/schema private
    • Difficulties compounded when information can be inferred, see, e.g., Cuenca Grau et al. Privacy-preserving query answering in logic-based information systems. ECAI 2008.
Problems & Research Challenges

• Infrastructure
  - Standardised query language
    • SPARQL standard for RDF
    • Currently being extended for OWL, see http://www.w3.org/2009/sparql/wiki/Main_Page
  - Privacy and information hiding
    • May want to keep parts of data/schema private
    • Difficulties compounded when information can be inferred, see, e.g., Cuenca Grau et al. Privacy-preserving query answering in logic-based information systems. ECAI 2008.
  - ...


The 9th International Semantic Web Conference
Shanghai International Convention Center, Shanghai, China
Nov 7th - 11th, 2010
http://iswc2010.semanticweb.org

ISWC 2010
Semantics for a Better Web!

General Chair
Ian Horrocks

Program Chairs
Peter F. Patel-Schneider
Yue Pan

Local Chair
Yong Yu

Workshop & Tutorial Chairs
Philippe Cudré-Mauroux
Bijan Parsia

Poster & Demo Chairs
Huajun Chen
Axel Polleres

Industry & Semantic Web in Use Chairs
Pascal Hitzler
Peter Mika

Doctoral Consortium Chair
Jeff Pan

Semantic Web Challenge Chairs
Chris Bizer
Diana Maynard

Publicity Chair
Sebastian Rudolph

Metadata Chair
Jie Bao

Proceedings Chair
Birte Glimm

Sponsor Chairs
Kendall Clark
Anand Ranganathan

Local Organization
Dingyi Han
Gui-Rong Xue
Haifeng Wang
Lei Zhang

Running Venue
Great Time
Fascinating City
Excellent Food

IBM Research
Thanks To

• Boris Motik
• Yevgeny Kazakov
• Héctor Pérez-Urbina
• Rob Shearer
• Bernardo Cuenca Grau
• Birte Glimm
Thank you for listening
Thank you for listening

Any questions?
Select Bibliography


