DEPARTMENT OF
COMPUTER
SCIENCE

UNIVERSITY OF
OXFORD

$\text{Semantics} \sqcap \text{Scalability} \models \bot\ ?$

Ian Horrocks
Information Systems Group
Department of Computer Science
University of Oxford

The Semantic Web

- Web "invented" by **Tim Berners-Lee** (an Oxford graduate!), then a physicist working at CERN
- His original vision of the Web was much more **ambitious** than the reality of the existing (syntactic) Web:

"… a set of **connected applications** … forming a **consistent logical web of data** … information is given **well-defined meaning**, better enabling computers and people to work in cooperation …"

- This vision of the Web has become known as the **Semantic Web**
- Latest (refined) definition:

  "a web of data that can be processed directly and indirectly by machines"

Semantic Technologies

- Initial focus was on necessary **underpinning**, including:

Semantic Technologies

- Initial focus was on necessary **underpinning**, including:
  - Languages

W3C® RDF

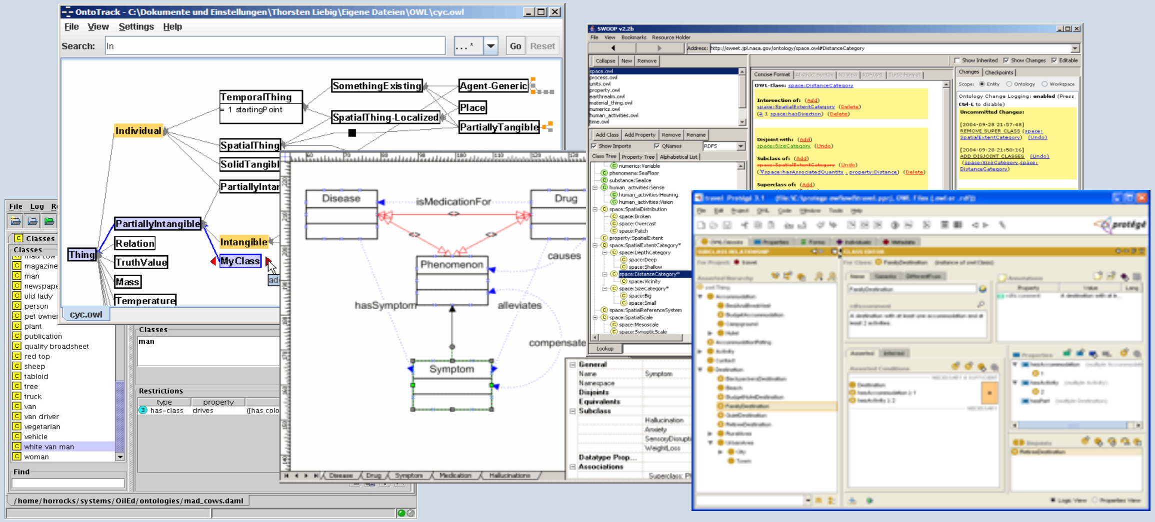
W3C OWL · W3C RDFa · W3C POWDER · W3C SKOS · W3C RDF · W3C SPARQL

Semantic Technologies

- Initial focus was on necessary **underpinning**, including:
  - Languages
  - Storage and querying

Semantic Technologies

- Initial focus was on necessary **underpinning**, including:
  - Languages
  - Storage and querying
  - Development tools



Information Systems Group

DEPARTMENT OF COMPUTER SCIENCE

UNIVERSITY OF OXFORD

EPSRC Engineering and Physical Sciences Research Council

SEVENTH FRAMEWORK PROGRAMME

Semantic Technologies

- Initial focus was on necessary **underpinning**, including:
  - Languages
  - Storage and querying
  - Development tools
- Resulting **robust infrastructure** used in SW applications

UNIVERSITY OF OXFORD DEPARTMENT OF COMPUTER SCIENCE
Information Systems Group
EPSRC Engineering and Physical Sciences Research Council
SEVENTH FRAMEWORK PROGRAMME

Semantic Technologies

- Initial focus was on necessary **underpinning**, including:
  - Languages
  - Storage and querying
  - Development tools
- Resulting **robust infrastructure** used in SW applications
- Also increasingly used in **“Intelligent Information System”** applications

How Does it Work?

❶ **Standardised language for exchanging data**

- W3C standard for data exchange is **RDF**
- RDF is a simple language consisting of <S P O> **triples**
  - for example <eg:Ian eg:worksAt eg:Oxford>
  - all S,P,O are URIs or literals (data values)
- **URIs** provides a flexible **naming scheme**
- Set of triples can be viewed as a **graph**

How Does it Work?

❶ **Standardised language for exchanging data**

How Does it Work?

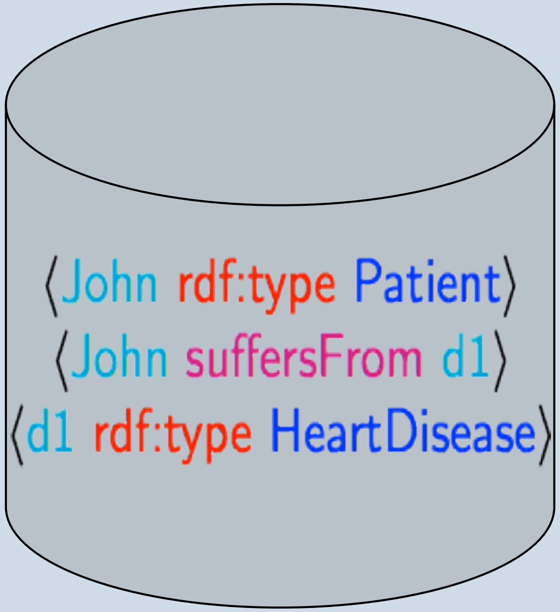
❷ **Standardised language for exchanging vocabularies/schemas**

- W3C standard for vocabulary/schema exchange is **OWL**
- OWL provides for rich conceptual schemas, aka **ONTOLOGIES**

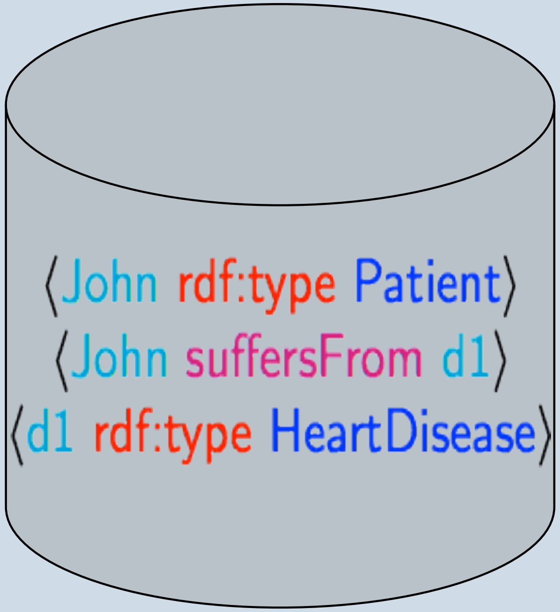
$$
\begin{aligned}
\mathsf{Heart} &\sqsubseteq \mathsf{MuscularOrgan} \sqcap \\
&\quad \exists \mathsf{isPartOf}.\mathsf{CirculatorySystem} \\
\mathsf{HeartDisease} &\equiv \mathsf{Disease} \sqcap \\
&\quad \exists \mathsf{affects}.\mathsf{Heart} \\
\mathsf{VascularDisease} &\equiv \mathsf{Disease} \sqcap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

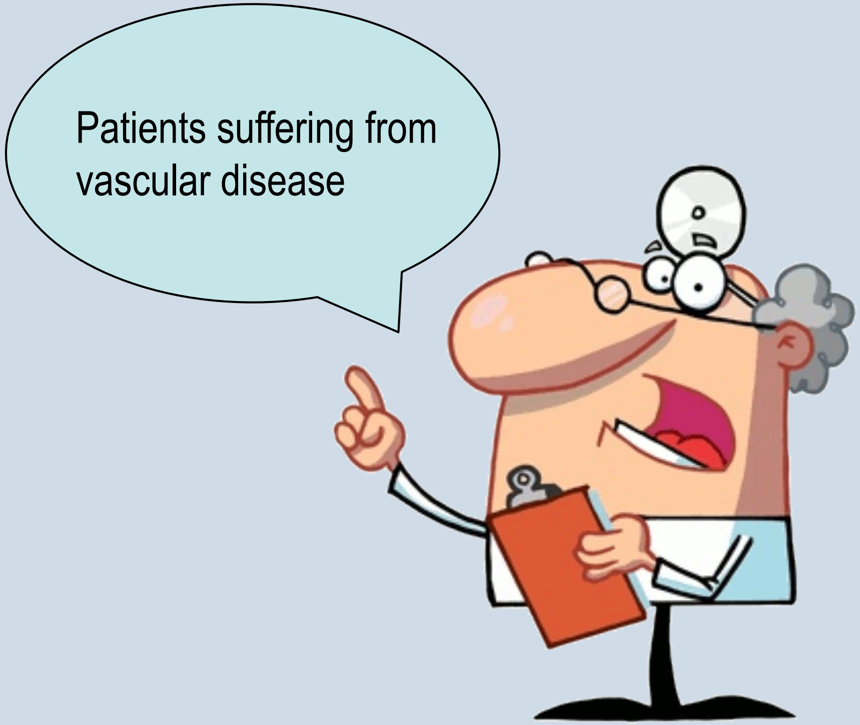
UNIVERSITY OF OXFORD DEPARTMENT OF COMPUTER SCIENCE
Information Systems Group
EPSRC Engineering and Physical Sciences Research Council
SEVENTH FRAMEWORK PROGRAMME

How Does it Work?

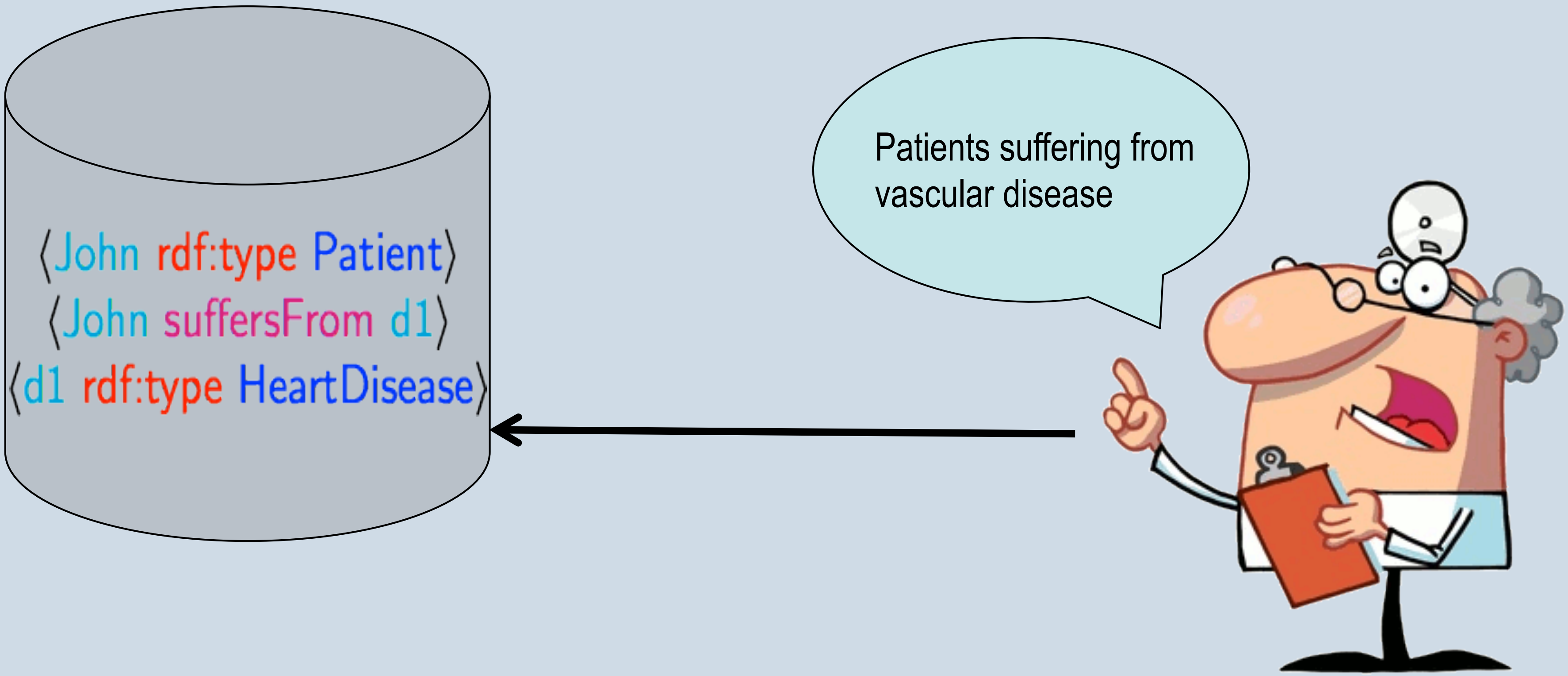


How Does it Work?



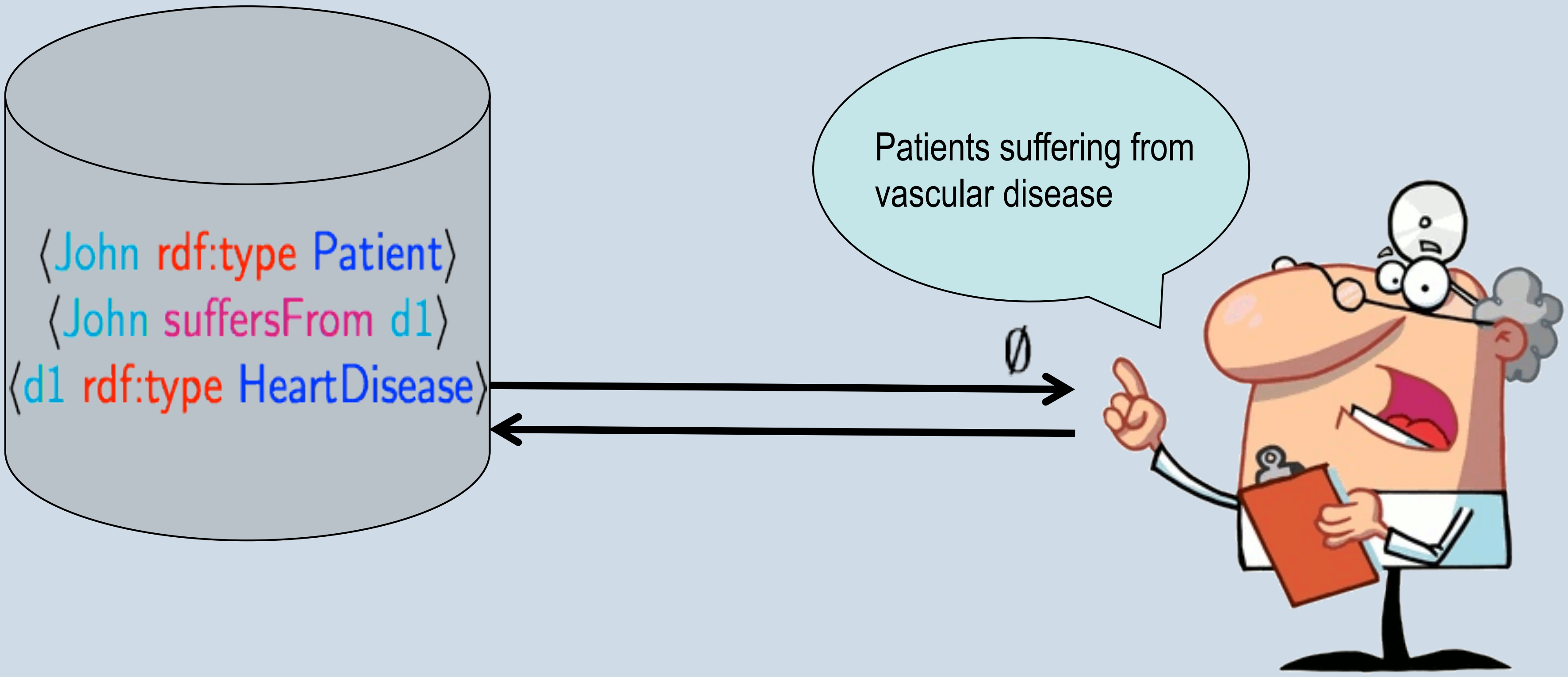


How Does it Work?



UNIVERSITY OF OXFORD DEPARTMENT OF COMPUTER SCIENCE
Information Systems Group
EPSRC Engineering and Physical Sciences Research Council
SEVENTH FRAMEWORK PROGRAMME

How Does it Work?

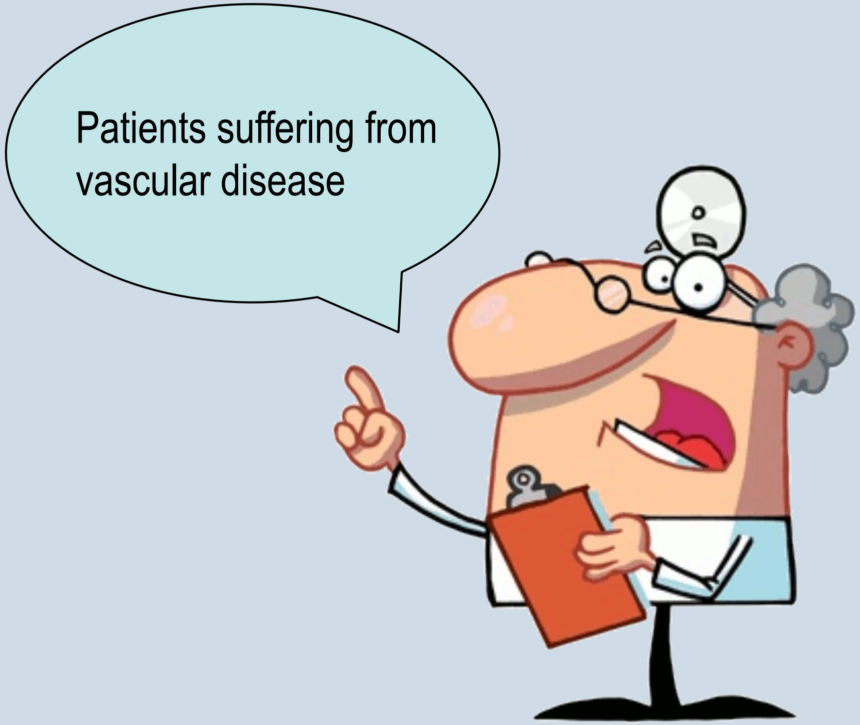


UNIVERSITY OF OXFORD DEPARTMENT OF COMPUTER SCIENCE
Information Systems Group
EPSRC Engineering and Physical Sciences Research Council
SEVENTH FRAMEWORK PROGRAMME

How Does it Work?

⟨John rdf:type Patient⟩
⟨John suffersFrom d1⟩
⟨d1 rdf:type HeartDisease⟩

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



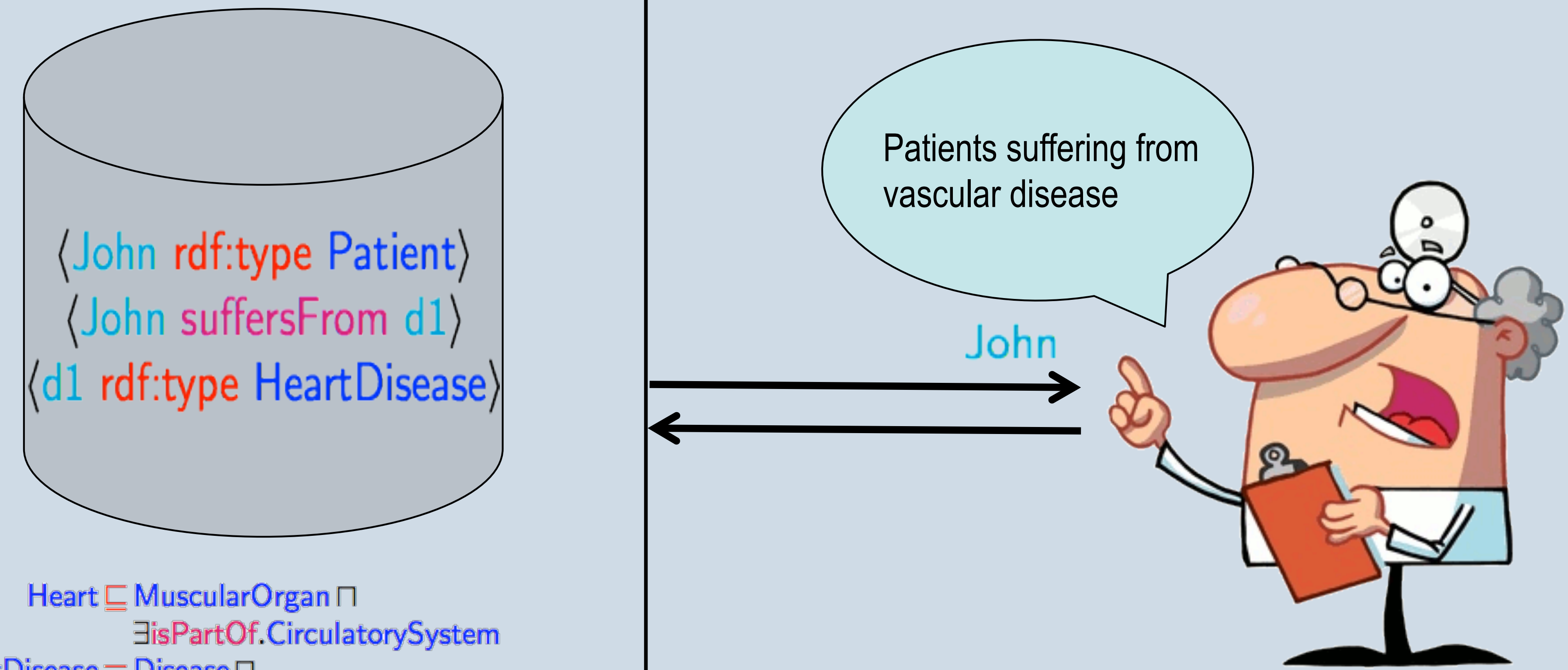
How Does it Work?

⟨John rdf:type Patient⟩
⟨John suffersFrom d1⟩
⟨d1 rdf:type HeartDisease⟩

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

University of Oxford Department of Computer Science
Information Systems Group
EPSRC Engineering and Physical Sciences Research Council
Seventh Framework Programme

How Does it Work?



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

How Does it Work?

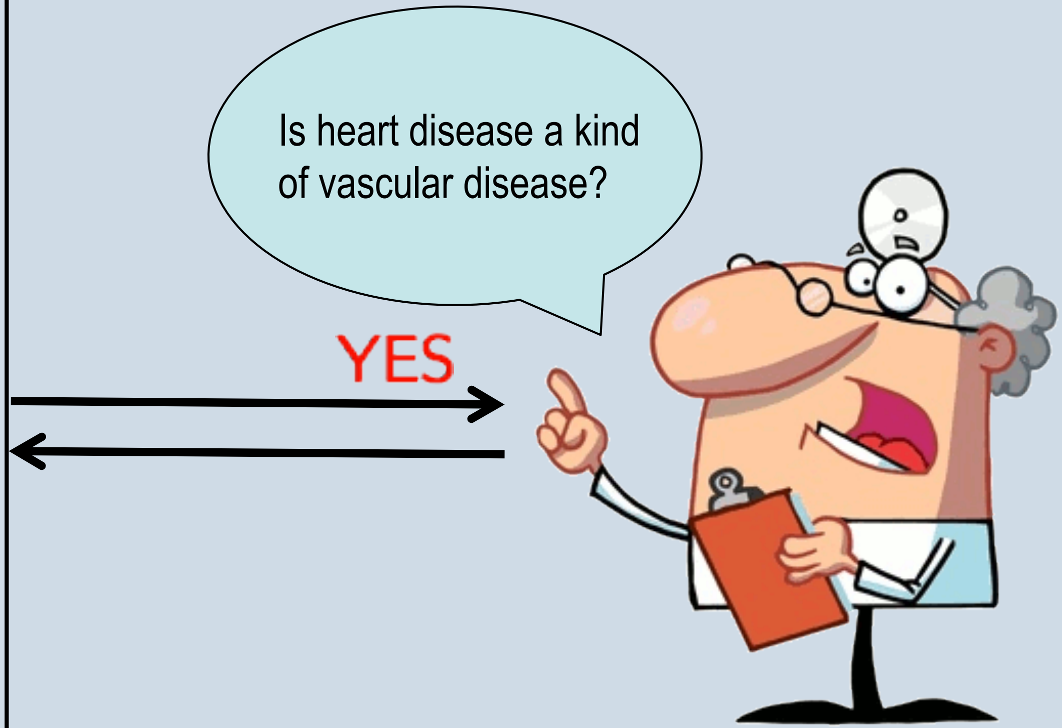
$\langle \text{John rdf:type Patient} \rangle$
$\langle \text{John sufferesFrom d1} \rangle$
$\langle \text{d1 rdf:type HeartDisease} \rangle$

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

How Does it Work?

$\langle \text{John rdf:type Patient} \rangle$
$\langle \text{John sufferesFrom d1} \rangle$
$\langle \text{d1 rdf:type HeartDisease} \rangle$

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



How Does it Work?

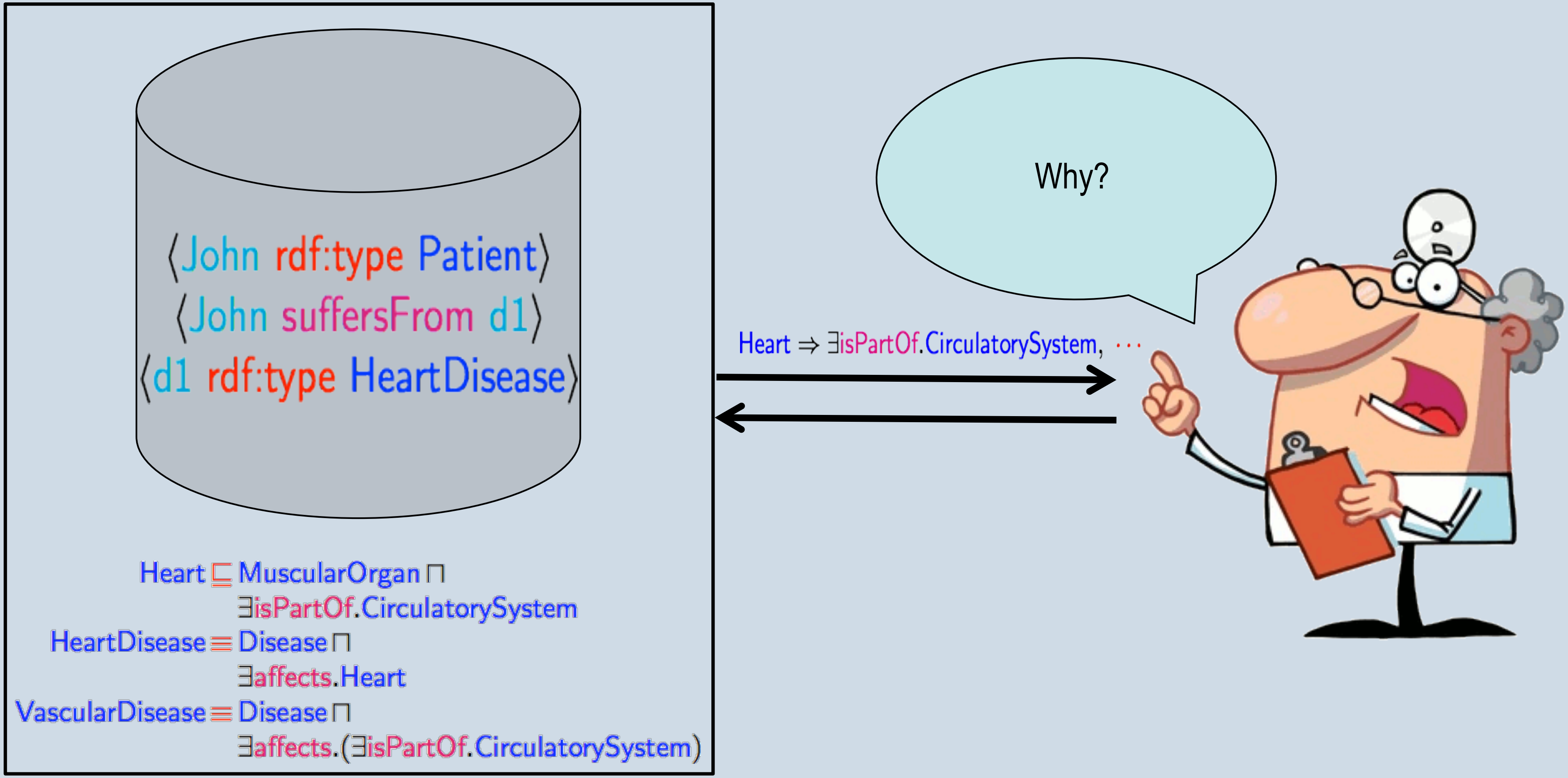
$\langle \text{John rdf:type Patient} \rangle$

$\langle \text{John sufferedFrom d1} \rangle$

$\langle \text{d1 rdf:type HeartDisease} \rangle$

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

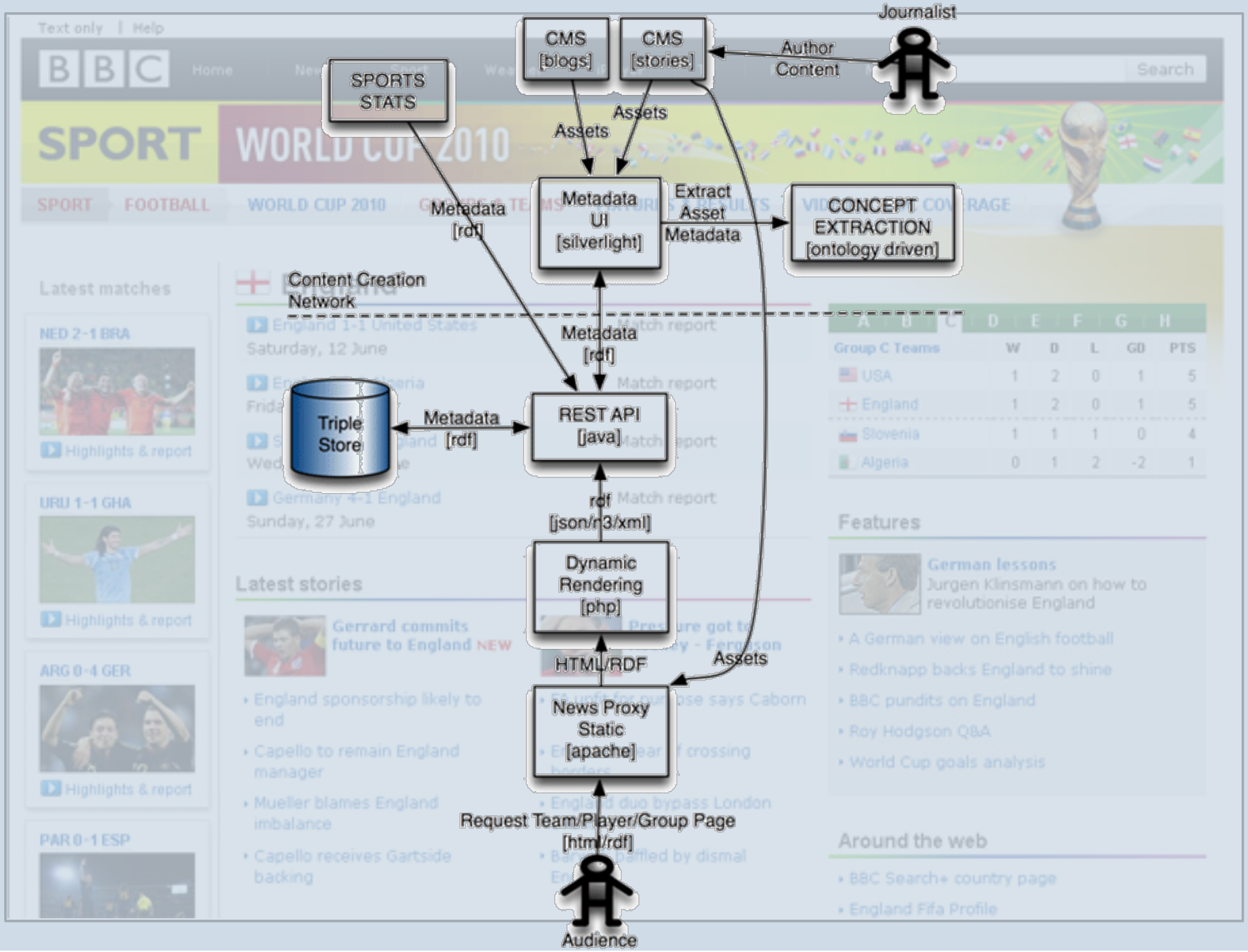
How Does it Work?



UNIVERSITY OF OXFORD DEPARTMENT OF COMPUTER SCIENCE
Information Systems Group
EPSRC Engineering and Physical Sciences Research Council
SEVENTH FRAMEWORK PROGRAMME

Applications: Semantic Web

Applications: Semantic Web



DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF OXFORD

Information Systems Group

EPSRC
Engineering and Physical Sciences Research Council

SEVENTH FRAMEWORK PROGRAMME

Applications: Semantic Web

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF OXFORD

Information Systems Group

EPSRC
Engineering and Physical Sciences Research Council

SEVENTH FRAMEWORK PROGRAMME

Applications: HCLS

- **SNOMED-CT** (Clinical Terms) ontology
  - provides common vocabulary for recording clinical data
  - used in healthcare systems of more than 15 countries, including Australia, Canada, Denmark, Spain, Sweden and the UK
  - “classified and checked for equivalencies” using ontology reasoners
- **OBO foundry** includes more than 100 biological and biomedical ontologies

  “continuous integration server running Elk and/or HermiT 24/7 checking that multiple independently developed ontologies are mutually consistent”
- **Siemens** “actively building OWL based clinical solutions”

Applications: Energy Supply Industry

- **EDF Energy** offer personalised energy saving advice to every customer
- **OWL ontology** used to model relevant environmental factors
- **HermiT reasoner** used to match customer circumstances with relevant pieces of advice

Information Systems Group

UNIVERSITY OF OXFORD DEPARTMENT OF COMPUTER SCIENCE

EPSRC Engineering and Physical Sciences Research Council

SEVENTH FRAMEWORK PROGRAMME

Applications: Intelligent Mobile Platform

- **Samsung** developing Intelligent Moblile Platform to support context-aware applications
- IMP monitors environment via **sensor data** (GPS, compass, accelerometer, ...)
- **OWL ontology** used to model environment and **infer context** (e.g., coffee with friends)
- Applications exploit context to enable more **intelligent behaviour**

UNIVERSITY OF OXFORD
DEPARTMENT OF COMPUTER SCIENCE

Information Systems Group

EPSRC
Engineering and Physical Sciences Research Council

SEVENTH FRAMEWORK PROGRAMME

Applications: Oil and Gas Industry

- **Statoil** use data to inform production and exploration management

  Large and complex data sets are difficult and time consuming to use

- Semantic technology can **improve access** to relevant data

- Test deployment in EU project Optique

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF OXFORD

Information Systems Group

EPSRC
Engineering and Physical Sciences Research Council

SEVENTH FRAMEWORK PROGRAMME

Theory $\rightsquigarrow$ Practice

-

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF OXFORD

Information Systems Group

EPSRC
Engineering and Physical Sciences Research Council

SEVENTH FRAMEWORK PROGRAMME

Theory $\rightsquigarrow$ Practice

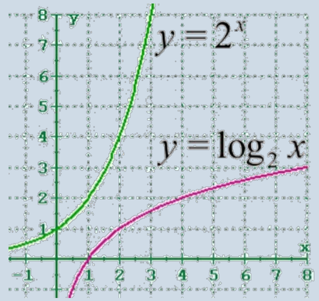
- OWL based on **description logic** $\mathcal{SROIQ}$

Theory $\rightsquigarrow$ Practice

- OWL based on **description logic** $\mathcal{SROIQ}$
- DLs are a family of FOL fragments
  - Clear semantics
  - Well understood computational properties (e.g., decidability, complexity)
  - Simple goal directed reasoning algorithms

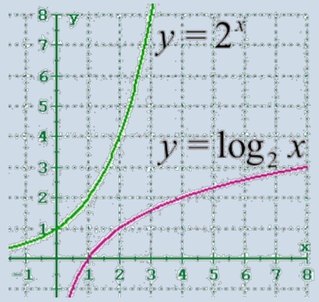
Theory $\leadsto$ Practice

- OWL based on **description logic $\mathcal{SROIQ}$**
- DLs are a family of FOL fragments
  - Clear semantics
  - Well understood computational properties (e.g., decidability, complexity)
  - Simple goal directed reasoning algorithms
- OWL is decidable, but highly **highly intractable**
  - N2ExpTime-comlete combined complexity
  - NP-hard data complexity (-v- logspace for databases)



Theory $\rightsquigarrow$ Practice

- OWL based on **description logic $\mathcal{SROIQ}$**
- DLs are a family of FOL fragments
  - Clear semantics
  - Well understood computational properties (e.g., decidability, complexity)
  - Simple goal directed reasoning algorithms
- OWL is decidable, but highly **highly intractable**
  - N2ExpTime-comlete combined complexity
  - NP-hard data complexity (-v- logspace for databases)



How can we provide robustly scalable query answering?

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF OXFORD
Information Systems Group
EPSRC
Engineering and Physical Sciences Research Council
SEVENTH FRAMEWORK PROGRAMME

Various Approaches — Different Tradeoffs

❶ 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, but not robust and unlikely to scale to large data sets

Various Approaches — Different Tradeoffs

❷ 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 — Different Tradeoffs

❷ 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 ontology reasoners:

- E.g., CEL, ELK, ...
- Based on “consequence based” (deduction) theorem provers
- Target HCLS applications where many ontologies are (mainly) in the EL profile

Schema Reasoning — Solved Problem?

	SNOMED CT	GALEN	FMA	GO
Logic	$\mathcal{EL}$	$\mathcal{EL}$	$\mathcal{EL}$	$\mathcal{EL}$
#classes	315,489	23,136	78,977	19,468
#properties	58	950	7	1
#axioms	430,844	36,547	121,712	28,897
$\#\sqsubseteq$	$> 10^{11}$	$> 10^{8}$	$> 10^{9}$	$> 10^{8}$
ELK (1 worker)	13.15	1.33	0.44	0.20
ELK (4 workers)	5.02	0.77	0.39	0.19
	Plant Anat.	**SWEET-P**	**NCI-2**	**DOLCE-P**
Logic	$\mathcal{SHIF}$	$\mathcal{SHOIN}$	$\mathcal{ALCH}$	$\mathcal{SHOIN}$
#classes	19,145	1,728	70,576	118
#properties	82	145	189	264
#axioms	35,770	2,419	100,304	265
$\#\sqsubseteq$	$> 10^{8}$	$> 10^{6}$	$> 10^{9}$	$> 10^{4}$
HermiT	11.2	11.2	—	105.1
Pellet	87.2	—	172.0	105.1
FaCT++	22.9	0.2	60.7	—

Schema Reasoning — Solved Problem?

- Full expressive power may be needed to model, e.g.:
  - *non-viral pneumonia* (negation)
  - *infectious pneumonia* is caused by a *virus* or a *bacterium* (disjunction)
  - *double pneumonia* occurs in two *lungs* (cardinalities)
  - *groin* has a part that is part of the *abdomen*, and has a part that is part of the *leg* (inverse properties)
- Single non-EL axiom may incur massive performance penalty

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF OXFORD
Information Systems Group
EPSRC
Engineering and Physical Sciences Research Council
SEVENTH FRAMEWORK PROGRAMME

MORe Modular Reasoner

- Integrates powerful (slower) and weaker (faster) reasoners
- Exploits module extraction techniques to identify subset of ontology that can be completely classified using fast reasoner.
- Slower reasoner performs as few computations as possible
- Bulk of computation delegated to faster reasoner
- Current prototype integrates **HermiT** and **ELK** [1]

[1] Armas Romero, Cuenca Grau, and Horrocks. Modular Combination of Reasoners for Ontology Classification. In Proc. of ISWC 2012 (to appear).

MORe Modular Reasoner

Ontology	$\lvert \mathcal{O} \setminus \mathcal{O}_{\mathcal{L}} \rvert$	$\lvert \Sigma^{\mathcal{L}} \rvert$	$\lvert \mathcal{M}_{[\mathcal{O},\overline{\Sigma^{\mathcal{L}}}]} \rvert$	Classif. time (seconds)			
				HermiT	MORe		
					total	HermiT	ELK
GO	0	100%	0%	7.1	2.2 (↓69.0%)	0	0.1
Gazeteer	0	100%	0%	838.1	28.2 (↓96.6%)	0	15.6
NCI	65	94.9%	15.4%	84.1	28.6 (↓66.0%)	15.8	3.3
Protein	12	98.1%	6.6%	11.4	2.9 (↓74.6%)	0.4	0.9
Biomodels	22,079	45.2%	66.4%	741.4	575.6 (↓22.4%)	540.1	2.6
cellCycle	1	$> 99.9\%$	$< 0.1\%$	–	13.9 (–)	<0.1	4.9
NCI+CHEBI	65	95.6%	10.3%	116.6	34.0 (↓70.8%)	16.3	4.1
NCI+GO	65	96.7%	10.4%	110.0	37.6 (↓65.8%)	17.6	3.2
NCI+Mouse	65	96.0%	13.3%	93.7	31.0 (↓66.9%)	16.6	2.6

OWL 2 EL — Data Retrieval Queries?

- PTime potentially problematical for very large datasets

OWL 2 EL — Data Retrieval Queries?

- PTime potentially problematical for very large datasets
- Various approaches:
  - Materialise taxonomy and use DBMS (incomplete reasoning)
  - “Combined approach” using materialisation + OBDA [2]
  - Datalog engine with (some form of) query rewriting [3]
  - Highly optimised ABox reasoners [4]

[2] Kontchakov, Lutz, Toman, Wolter, Zakharyaschev: The Combined Approach to Ontology-Based Data Access. IJCAI 2011.

[3] Stefanoni, Motik, Horrocks: Small Datalog Query Rewritings for EL. DL 2012

[4] Kazakov, Kroetzsch, Simancik: Practical Reasoning with Nominals in the EL Family of Description Logics. KR 2012

Various Approaches — Different Tradeoffs

❷ Use a suitable "profile" and specialised reasoner:

✓ LogSpace query answering (in size of data)

✓ Reliable answers (for inputs in the profile)

✗ Restricted expressivity of the ontology language

✗ Reasoners reject inputs outside profile

Various Approaches — Different Tradeoffs

❷ Use a suitable "profile" and specialised reasoner:

✓ LogSpace query answering (in size of data)

✓ Reliable answers (for inputs in the profile)

✗ Restricted expressivity of the ontology language

✗ Reasoners reject inputs outside profile

OWL 2 QL ontology reasoners:

- E.g., QuOnto, **Requiem**, ...
- Based on query rewriting technique — ontology used to rewrite (expand) query
- Targets applications where data stored in RDBMS — aka **Ontology Based Data Access** (OBDA)

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF OXFORD
Information Systems Group
EPSRC
Engineering and Physical Sciences Research Council
SEVENTH FRAMEWORK PROGRAMME

Query Rewriting — How Does It Work?

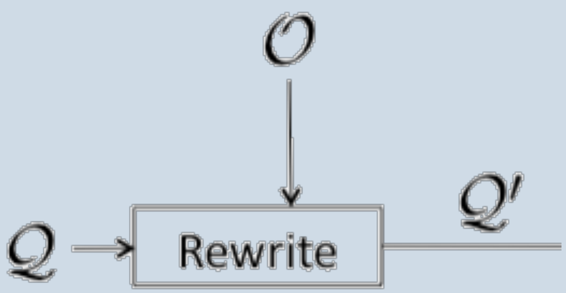
Given ontology $\mathcal{O}$ query $\mathcal{Q}$ and mappings $\mathcal{M}$:

UNIVERSITY OF OXFORD DEPARTMENT OF COMPUTER SCIENCE
Information Systems Group
EPSRC Engineering and Physical Sciences Research Council
SEVENTH FRAMEWORK PROGRAMME

Query Rewriting — How Does It Work?

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

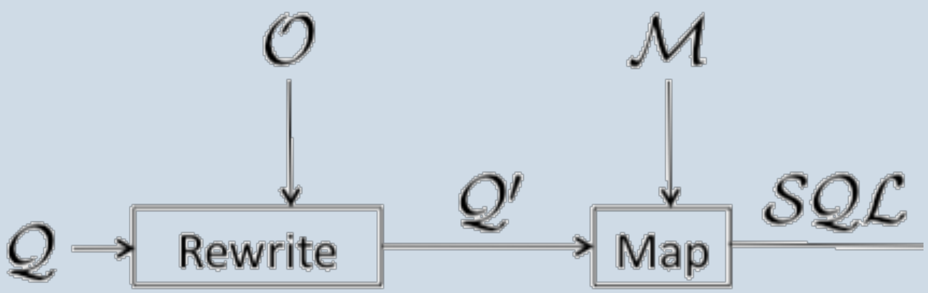
- **Rewrite** $\mathcal{Q} \rightarrow \mathcal{Q}'$ s.t. answering $\mathcal{Q}'$ without $\mathcal{O}$ equivalent to answering $\mathcal{Q}$ w.r.t. $\mathcal{O}$ *for any dataset*



Query Rewriting — How Does It Work?

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

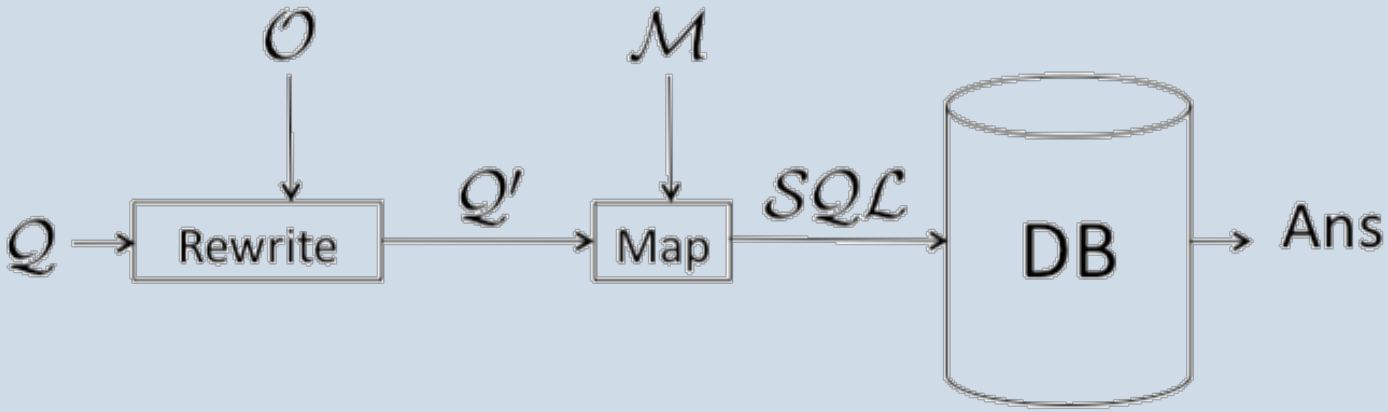
- **Rewrite** $\mathcal{Q} \rightarrow \mathcal{Q}'$ s.t. answering $\mathcal{Q}'$ without $\mathcal{O}$ equivalent to answering $\mathcal{Q}$ w.r.t. $\mathcal{O}$ *for any dataset*
- **Map** ontology queries → DB queries (typically SQL) using mappings $\mathcal{M}$ to rewrite $\mathcal{Q}$' into a DB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



Query Rewriting — How Does It Work?

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

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



Query Rewriting — Example

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

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

$$\mathcal{M}\begin{cases} \mathsf{Doctor} \mapsto \mathsf{SELECT\ Name\ FROM\ Doctor} \\ \mathsf{Patient} \mapsto \mathsf{SELECT\ Name\ FROM\ Patient} \\ \mathsf{treats} \mapsto \mathsf{SELECT\ DName,\ PName\ FROM\ Treats} \end{cases}$$

UNIVERSITY OF OXFORD DEPARTMENT OF COMPUTER SCIENCE
Information Systems Group
EPSRC Engineering and Physical Sciences Research Council
SEVENTH FRAMEWORK PROGRAMME

Query Rewriting — Example

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

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

$$\mathcal{Q}'\left\{\begin{array}{l}Q(x) \leftarrow \mathsf{treats}(x,y) \wedge \mathsf{Patient}(y)\\ Q(x) \leftarrow \mathsf{Doctor}(x) \wedge \mathsf{Patient}(f(x))\\ Q(x) \leftarrow \mathsf{treats}(x,f(x)) \wedge \mathsf{Doctor}(x)\\ Q(x) \leftarrow \mathsf{Doctor}(x)\\ Q(x) \leftarrow \mathsf{Consultant}(x)\end{array}\right.$$

$$\mathcal{M}\left\{\begin{array}{rcl}\mathsf{Doctor} & \mapsto & \text{SELECT Name FROM Doctor}\\ \mathsf{Patient} & \mapsto & \text{SELECT Name FROM Patient}\\ \mathsf{treats} & \mapsto & \text{SELECT DName, PName FROM Treats}\end{array}\right.$$

Query Rewriting — Example

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

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

$$Q' \begin{cases} Q(x) \leftarrow \mathsf{treats}(x, y) \wedge \mathsf{Patient}(y) \\ \cancel{Q(x) \leftarrow \mathsf{Doctor}(x) \wedge \mathsf{Patient}(f(x))} \\ \cancel{Q(x) \leftarrow \mathsf{treats}(x, f(x)) \wedge \mathsf{Doctor}(x)} \\ Q(x) \leftarrow \mathsf{Doctor}(x) \\ Q(x) \leftarrow \mathsf{Consultant}(x) \end{cases}$$

$$\mathcal{M} \left\{ \begin{array}{rcl} \mathsf{Doctor} & \mapsto & \mathsf{SELECT\ Name\ FROM\ Doctor} \\ \mathsf{Patient} & \mapsto & \mathsf{SELECT\ Name\ FROM\ Patient} \\ \mathsf{treats} & \mapsto & \mathsf{SELECT\ DName,\ PName\ FROM\ Treats} \end{array} \right.$$

Query Rewriting — Example

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

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

$$\mathcal{Q}' \begin{cases} Q(x) \leftarrow \mathsf{treats}(x, y) \wedge \mathsf{Patient}(y) \\ \cancel{Q(x) \leftarrow \mathsf{Doctor}(x) \wedge \mathsf{Patient}(f(x))} \\ \cancel{Q(x) \leftarrow \mathsf{treats}(x, f(x)) \wedge \mathsf{Doctor}(x)} \\ Q(x) \leftarrow \mathsf{Doctor}(x) \\ \cancel{Q(x) \leftarrow \mathsf{Consultant}(x)} \end{cases}$$

$$\mathcal{M} \begin{cases} \mathsf{Doctor} \mapsto \text{SELECT Name FROM Doctor} \\ \mathsf{Patient} \mapsto \text{SELECT Name FROM Patient} \\ \mathsf{treats} \mapsto \text{SELECT DName, PName FROM Treats} \end{cases}$$

Query Rewriting — Example

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

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

$$\mathcal{Q}'\begin{cases}Q(x) \leftarrow \mathsf{treats}(x,y) \wedge \mathsf{Patient}(y)\\ \cancel{Q(x) \leftarrow \mathsf{Doctor}(x) \wedge \mathsf{Patient}(f(x))}\\ \cancel{Q(x) \leftarrow \mathsf{treats}(x,f(x)) \wedge \mathsf{Doctor}(x)}\\ Q(x) \leftarrow \mathsf{Doctor}(x)\\ \cancel{Q(x) \leftarrow \mathsf{Consultant}(x)}\end{cases}$$

$$\mathcal{M}\begin{cases}\mathsf{Doctor} & \mapsto & \mathsf{SELECT\ Name\ FROM\ Doctor}\\ \mathsf{Patient} & \mapsto & \mathsf{SELECT\ Name\ FROM\ Patient}\\ \mathsf{treats} & \mapsto & \mathsf{SELECT\ DName,\ PName\ FROM\ Treats}\end{cases}$$

$$\mathcal{SQL}\begin{cases}\mathsf{SELECT\ Name\ FROM\ Doctor\ UNION}\\ \mathsf{SELECT\ DName\ FROM\ Treats,\ Patient\ WHERE\ PName{=}Name}\end{cases}$$

Query Rewriting — Issues

❶ **Rewriting**

- May be large (worst case exponential in size of ontology)
- Queries may be hard for existing DBMSs
- Ongoing work on OBDA optimisation techniques, e.g., [5]

❷ **Mappings**

- May be difficult to develop and maintain
- Little work in this area to date

[5] Rodriguez-Muro, Calvanese: High Performance Query Answering over DL-Lite Ontologies. KR 2012

Various Approaches — Different Tradeoffs

❸ 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:

- E.g., Oracle's Semantic Datastore, Sesame, Jena, OWLim, ...
- Based on RDF triple stores and chase-like materialisation
- Widely used in practice to reason with large datasets
- Complete (only) for RL ontologies and ground atomic queries

Materialisation — How Does It Work?

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

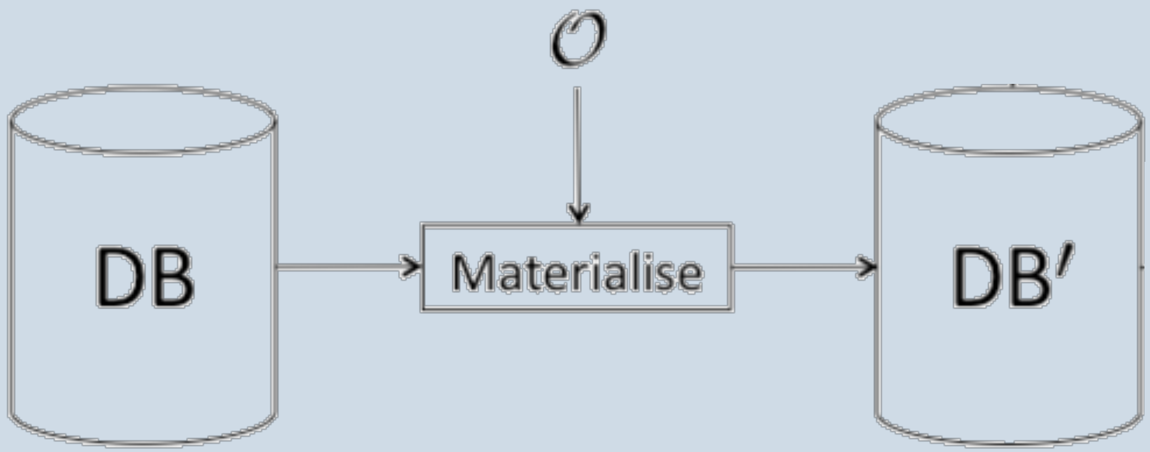
DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF OXFORD
Information Systems Group
EPSRC
Engineering and Physical Sciences Research Council
SEVENTH FRAMEWORK PROGRAMME

Materialisation — How Does It Work?

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

- **Materialise** (RDF) data DB $\longrightarrow$ 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



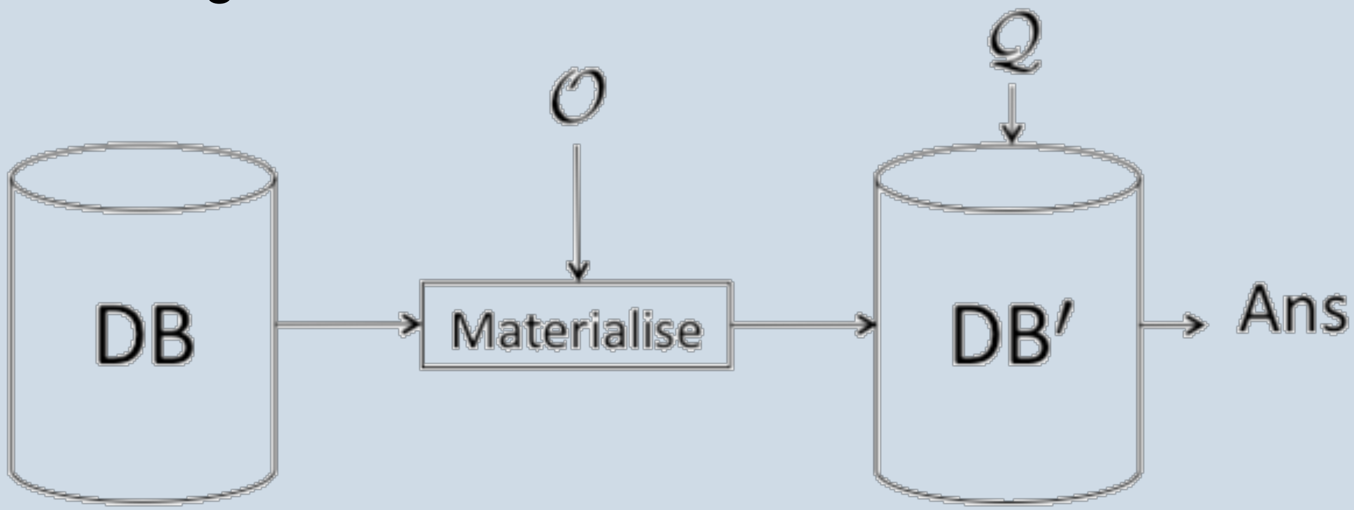
Materialisation — How Does It Work?

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$'$



DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF OXFORD
Information Systems Group
EPSRC
Engineering and Physical Sciences Research Council
SEVENTH FRAMEWORK PROGRAMME

Materialisation — Example

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

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

Materialisation — Example

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

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

Materialisation — Example

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

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

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

Materialisation — Example

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

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

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

Materialisation — Example

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

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

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

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

Materialisation — Example

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

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

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

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

Dealing With Frequently Changing Data

Adding data is relatively easy

- Monotonicity of FOL means that extending existing materialisation is sound
- Can still be quite costly if naively implemented

Changing/retracting data is much harder

- Naive solution requires all materialised facts to be discarded
- Re-materialisation very costly for large data sets
- But incremental reasoning is possible using view maintenance based techniques [6]

[6] Motik, Horrocks, and Kim. Delta-reasoner: a semantic web reasoner for an intelligent mobile platform. In Proc. of WWW 2012.

Dealing with Incompleteness

- Materialisation based reasoning complete for **OWL 2 RL** profile (and ground atomic queries)
- But for ontologies **outside the profile**:
  - 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**

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF OXFORD
Information Systems Group
EPSRC
Engineering and Physical Sciences Research Council
SEVENTH FRAMEWORK PROGRAMME

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 $\mathbf{S} = \langle \mathbf{S}_\perp, \mathbf{S}_Q \rangle$
  - $\mathbf{S}_\perp$ 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 $\mathcal{A} \in \mathbf{S}_\perp$
  - $\mathcal{R}$ 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.

DEPARTMENT OF COMPUTER SCIENCE UNIVERSITY OF OXFORD · Information Systems Group · EPSRC Engineering and Physical Sciences Research Council · SEVENTH FRAMEWORK PROGRAMME

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

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

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

UNIVERSITY OF OXFORD DEPARTMENT OF COMPUTER SCIENCE
Information Systems Group
EPSRC Engineering and Physical Sciences Research Council
SEVENTH FRAMEWORK PROGRAMME

Chase Materialisation — Example

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

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

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

Skolems

Chase Materialisation — Example

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

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

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

Skolems

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

DEPARTMENT OF COMPUTER SCIENCE, UNIVERSITY OF OXFORD

Information Systems Group

EPSRC Engineering and Physical Sciences Research Council

SEVENTH FRAMEWORK PROGRAMME

Chase Materialisation — Example

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

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

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

Skolems

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

Chase Materialisation — Example

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

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

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

Skolems

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

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

UNIVERSITY OF OXFORD DEPARTMENT OF COMPUTER SCIENCE
Information Systems Group
EPSRC Engineering and Physical Sciences Research Council
SEVENTH FRAMEWORK PROGRAMME

Chase Materialisation — Example

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

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

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

Skolems

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

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

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 $\mathcal{O}'$

UNIVERSITY OF OXFORD DEPARTMENT OF COMPUTER SCIENCE
Information Systems Group
EPSRC Engineering and Physical Sciences Research Council
SEVENTH FRAMEWORK PROGRAMME

Computing Lower and Upper Bounds

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

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF OXFORD
Information Systems Group
EPSRC
Engineering and Physical Sciences Research Council
SEVENTH FRAMEWORK PROGRAMME

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
  - Future work: use ***U*** \ ***L*** to identify (small) "relevant" subset of data needed to efficiently compute exact answer

[9] Zhou, Cuenca Grau, and Horrocks. Efficient Upper Bound Computation of Query Answers in Expressive Description Logics. In Proc. of DL 2012, volume 846 of CEUR.

Discussion

Numerous **exciting developments** & research areas

- Rewriting: optimisations, extensions (datalog engines), etc.
- Materialisation: chase, repair, truth maintenance, upper bounds etc.
- Combined techniques (materialisation+rewriting), Datalog
- Specialised RDF stores, Column stores, massive parallelism, etc.
- Parameterised complexity, new query evaluation techniques, etc.

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF OXFORD
Information Systems Group
EPSRC Engineering and Physical Sciences Research Council
SEVENTH FRAMEWORK PROGRAMME

Discussion

Numerous **exciting developments** & research areas

- Rewriting: optimisations, extensions (datalog engines), etc.
- Materialisation: chase, repair, truth maintenance, upper bounds etc.
- Combined techniques (materialisation+rewriting), Datalog
- Specialised RDF stores, Column stores, massive parallelism, etc.
- Parameterised complexity, new query evaluation techniques, etc.

Consider **progress on schema reasoning**:

Year	$\mathcal{O}$-size	Complete	Time (s)
1995	3,000	No	10^5
1998	3,000	Yes	300
2005	30,000	Yes	30
2010	400,000	Yes	5

Discussion

Numerous **exciting developments** & research areas

- Rewriting: optimisations, extensions (datalog engines), etc.
- Materialisation: chase, repair, truth maintenance, upper bounds etc.
- Combined techniques (materialisation+rewriting), Datalog
- Specialised RDF stores, Column stores, massive parallelism, etc.
- Parameterised complexity, new query evaluation techniques, etc.

Consider **progress on schema reasoning**:

[illegible]	[illegible]	[illegible]	[illegible]
1995	3,000	No	10^5
1998	[illegible]	[illegible]	[illegible]
2005	30,000	Yes	30
2010	400,000	Yes	5

Looking forward to similar progress on query answering!

Discussion

Numerous **exciting developments** & research areas

- Rewriting: optimisations, extensions (datalog engines), etc.
- Materialisation: chase, repair, truth maintenance, upper bounds etc.
- Hybrid techniques (materialisation+rewriting), Datalog
- Specialised RDF stores, Column stores, massive parallelism, etc.
- Parameterised complexity, new query evaluation techniques, etc.

$$\text{Semantics} \sqcap \text{Scalability} \not\models \bot \; !$$

Consider **progress on schema reasoning**:

Year	$\mathcal{O}$-size	Complete	Time (s)
1995	3,000	No	10^5
1998	3,000	Yes	300
2005	30,000	Yes	30
2010	400,000	Yes	5

Acknowledgements

EPSRC Engineering and Physical Sciences Research Council

SIXTH FRAMEWORK PROGRAMME

SEVENTH FRAMEWORK PROGRAMME

DARPA

References

[1] Armas Romero, Cuenca Grau, and Horrocks. Modular Combination of Reasoners for Ontology Classification. In Proc. of ISWC 2012 (to appear).

[2] Kontchakov, Lutz, Toman, Wolter, Zakharyaschev: The Combined Approach to Ontology-Based Data Access. IJCAI 2011.

[3] Stefanoni, Motik, Horrocks: Small Datalog Query Rewritings for EL. DL 2012

[4] Kazakov, Kroetzsch, Simancik: Practical Reasoning with Nominals in the EL Family of Description Logics. KR 2012

[5] Rodriguez-Muro, Calvanese: High Performance Query Answering over DL-Lite Ontologies. KR 2012

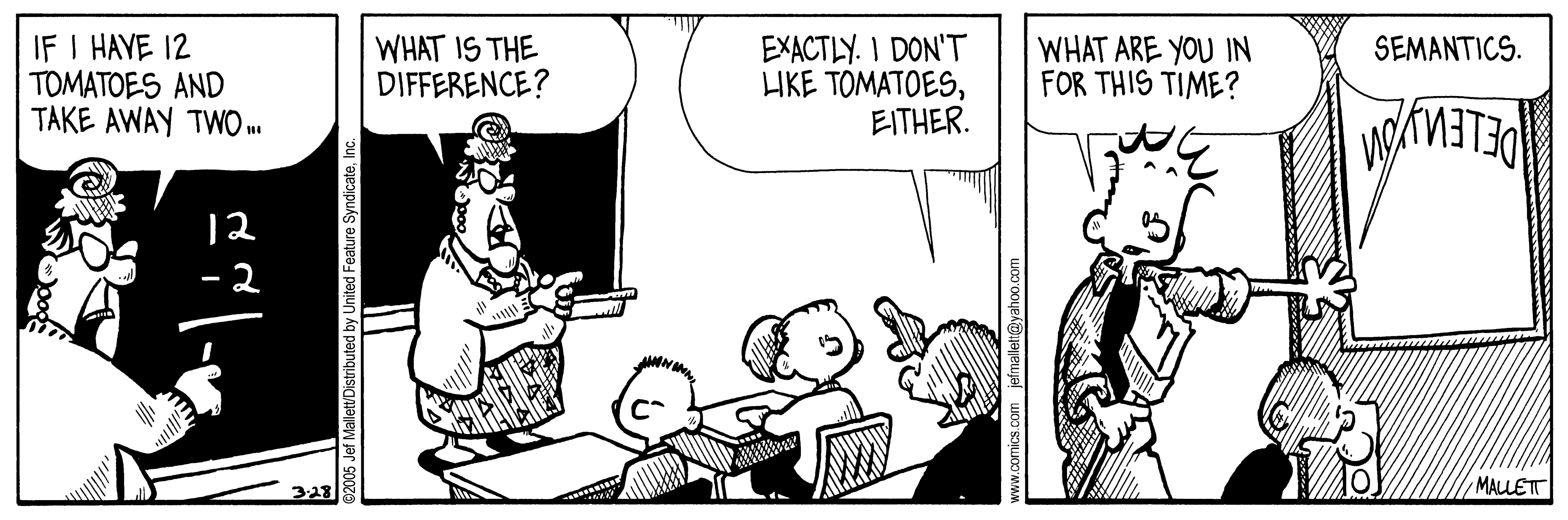
[6] Motik, Horrocks, and Kim. Delta-reasoner: a semantic web reasoner for an intelligent mobile platform. In Proc. of WWW 2012.

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

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

[9] Zhou, Cuenca Grau, and Horrocks. Efficient Upper Bound Computation of Query Answers in Expressive Description Logics. In Proc. of DL 2012

Thank you for listening



Any questions?

UNIVERSITY OF OXFORD DEPARTMENT OF COMPUTER SCIENCE

Information Systems Group

EPSRC Engineering and Physical Sciences Research Council

SEVENTH FRAMEWORK PROGRAMME