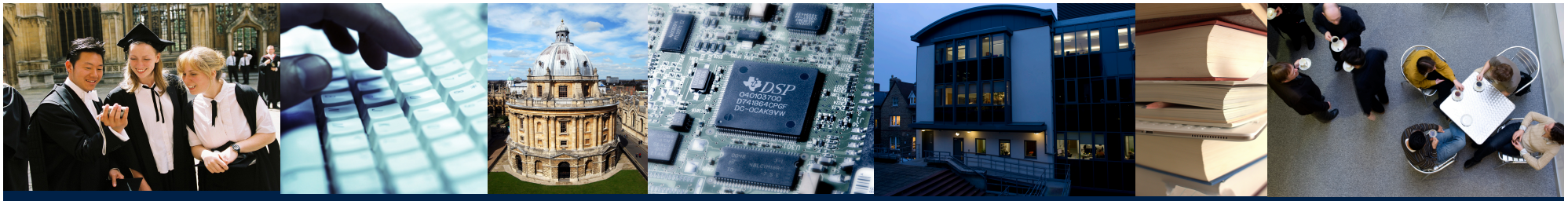
DEPARTMENT OF
COMPUTER
SCIENCE

UNIVERSITY OF
OXFORD



Semantics $\sqcap$ Scalability $\models \perp$?

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

Semantic Technologies

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


Hermit

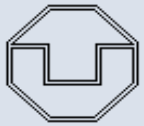
FaCT++

ORACLE

Racer

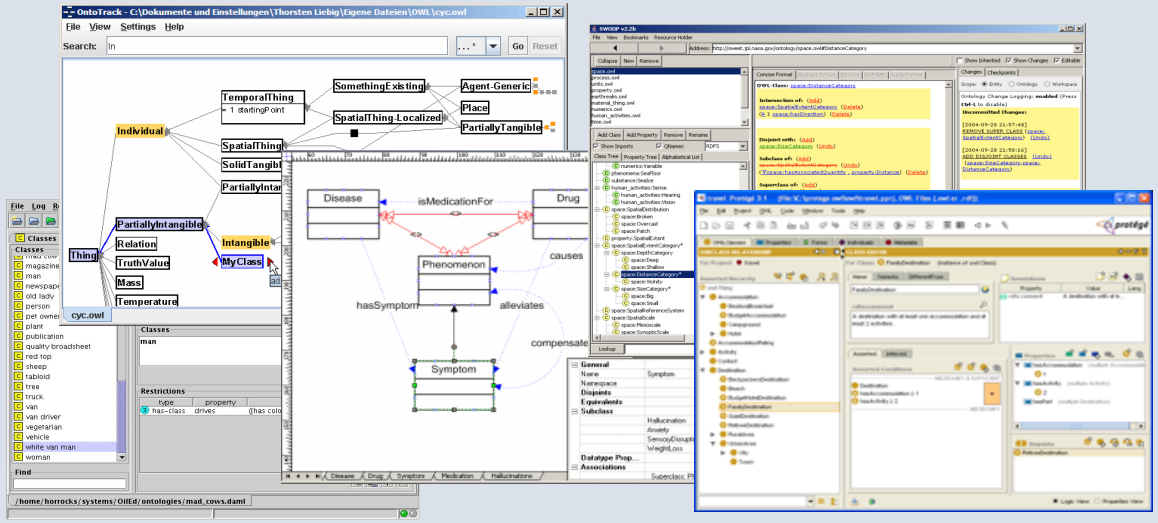
pellet

OWLIM


CEL

Semantic Technologies

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



Semantic Technologies

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

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?

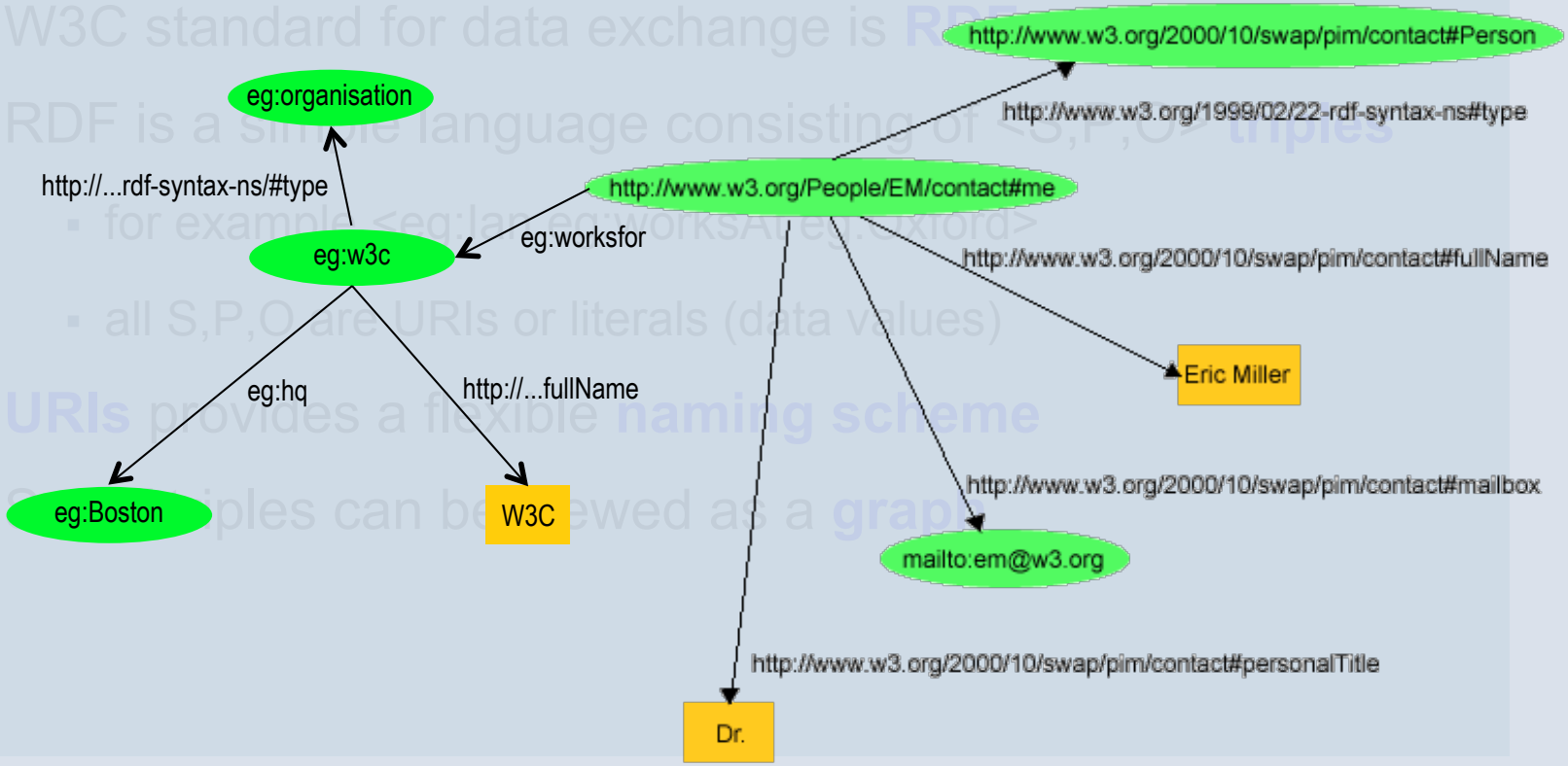
❶ **Standardised language for exchanging data**

- 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

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



How Does it Work?

❶ Standardised language for exchanging data

- W3C s
- RDF i triples
  - for
  - all
- URIs
- Set of

Triple		
S	**P**	**O**
em1234	rdf:type	Person
em1234	name	“Eric Miller”
em1234	title	“Dr”
em1234	mailbox	mailto:em@w3.org
em1234	worksfor	w3c
w3c	rdf:type	organisation
w3c	hq	Boston
w3c	name	“W3C”
...	...	...

How Does it Work?

❶ Standardised language for exchanging data

- W3C standard for data exchange is RDF
- RD
  -
  -
- URI
- Set

PERSON				
ID	NAME	TITLE	MAILBOX	WORKSFOR
em1234	“Eric Miller”	“Dr”	mailto:em@w3.org	w3c
...	...	...	...	...

ORGANISATION		
ID	NAME	HQ
w3c	“W3C”	Boston
...	...	...

…

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**

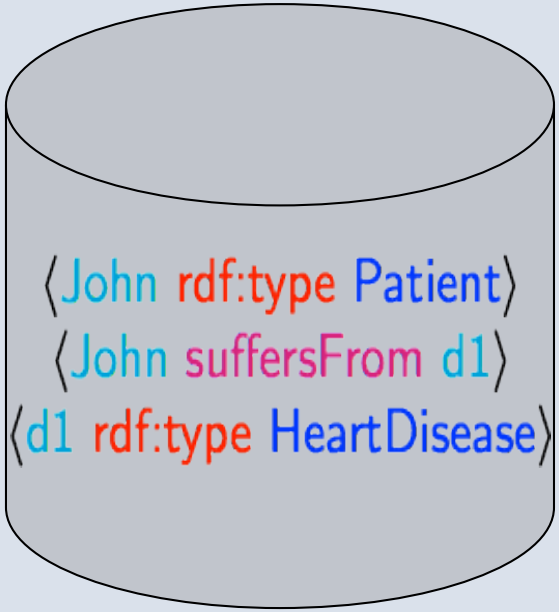
$$
\begin{aligned}
\mathsf{Heart} &\sqsubseteq \mathsf{MuscularOrgan} \sqcap \\
&\quad \exists \mathsf{isPartOf}.\mathsf{CirculatorySystem} \\
\mathsf{HeartDisease} &\equiv \mathsf{Disease} \sqcap \\
&\quad \exists \mathsf{affects}.\mathsf{Heart} \\
\mathsf{VascularDisease} &\equiv \mathsf{Disease} \sqcap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

How Does it Work?

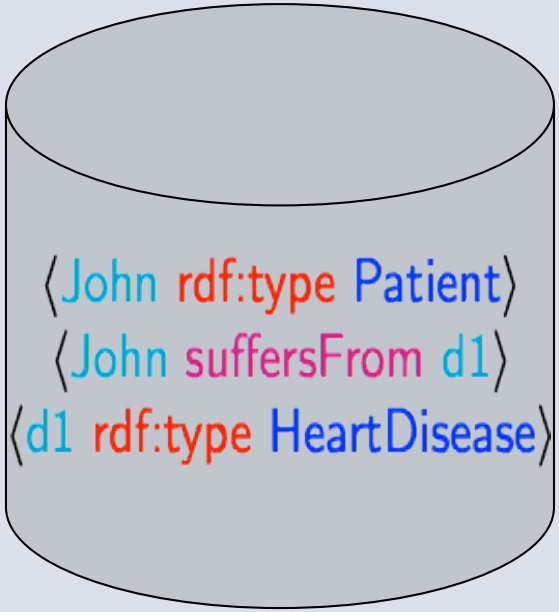
❸ **Standardised language for querying ontologies+data**

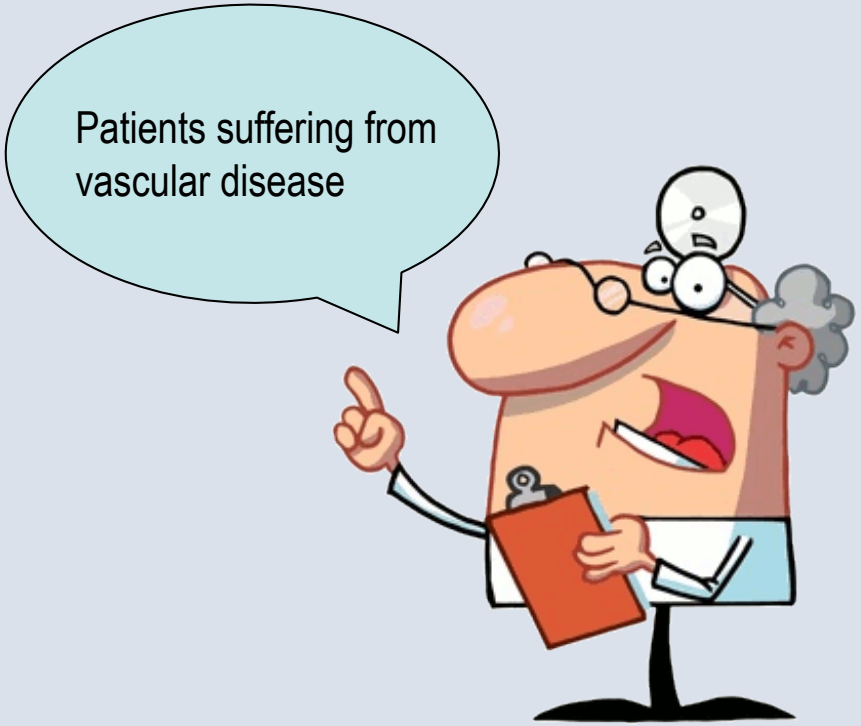
- W3C standard for querying is **SPARQL**
- SPARQL provides a rich query language comparable to SQL
  - ?x worksfor ?y .
    ?y rdf:type organisation .
    ?y hq Boston .
  - Select ?x
    where { ?x worksfor ?y .
    ?y rdf:type organisation .
    ?y hq Boston . }
  - Q(?x) ← worksfor(?x,?y) ∧ organisation(?y) ∧ hq(?y,Boston)

How Does it Work?

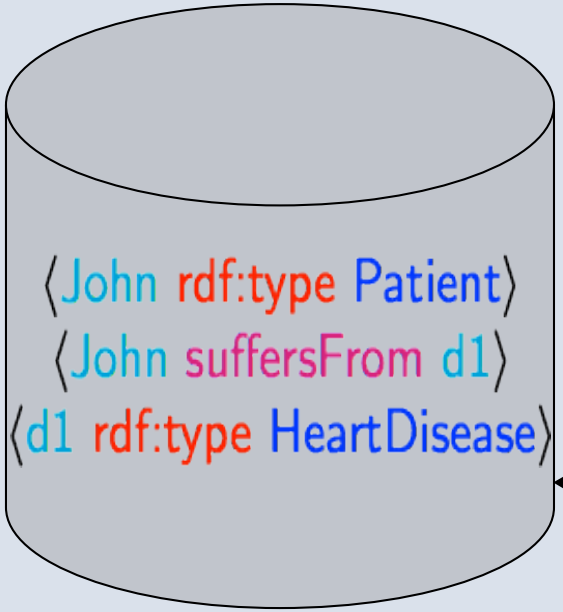


How Does it Work?

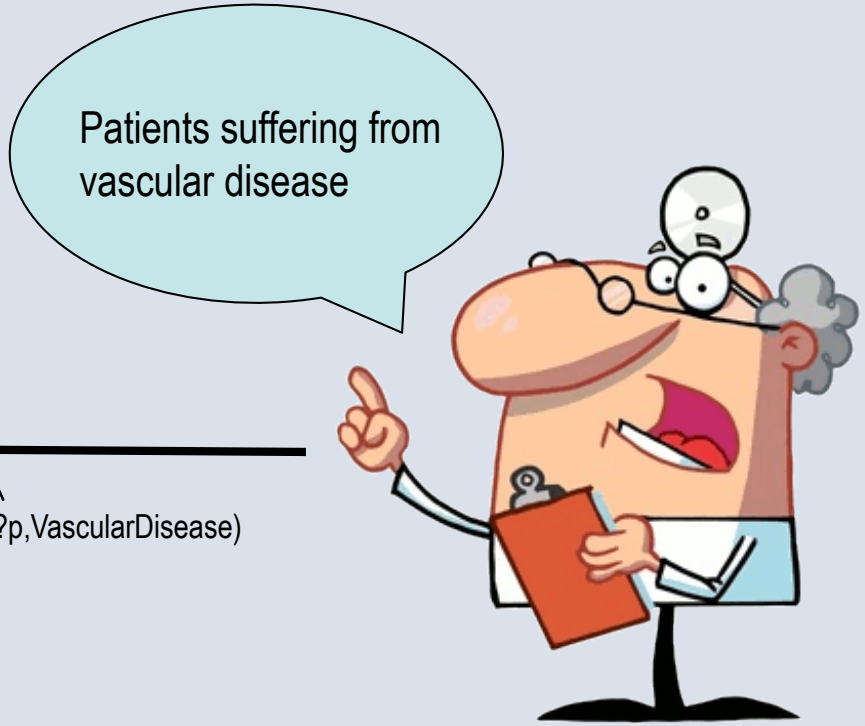




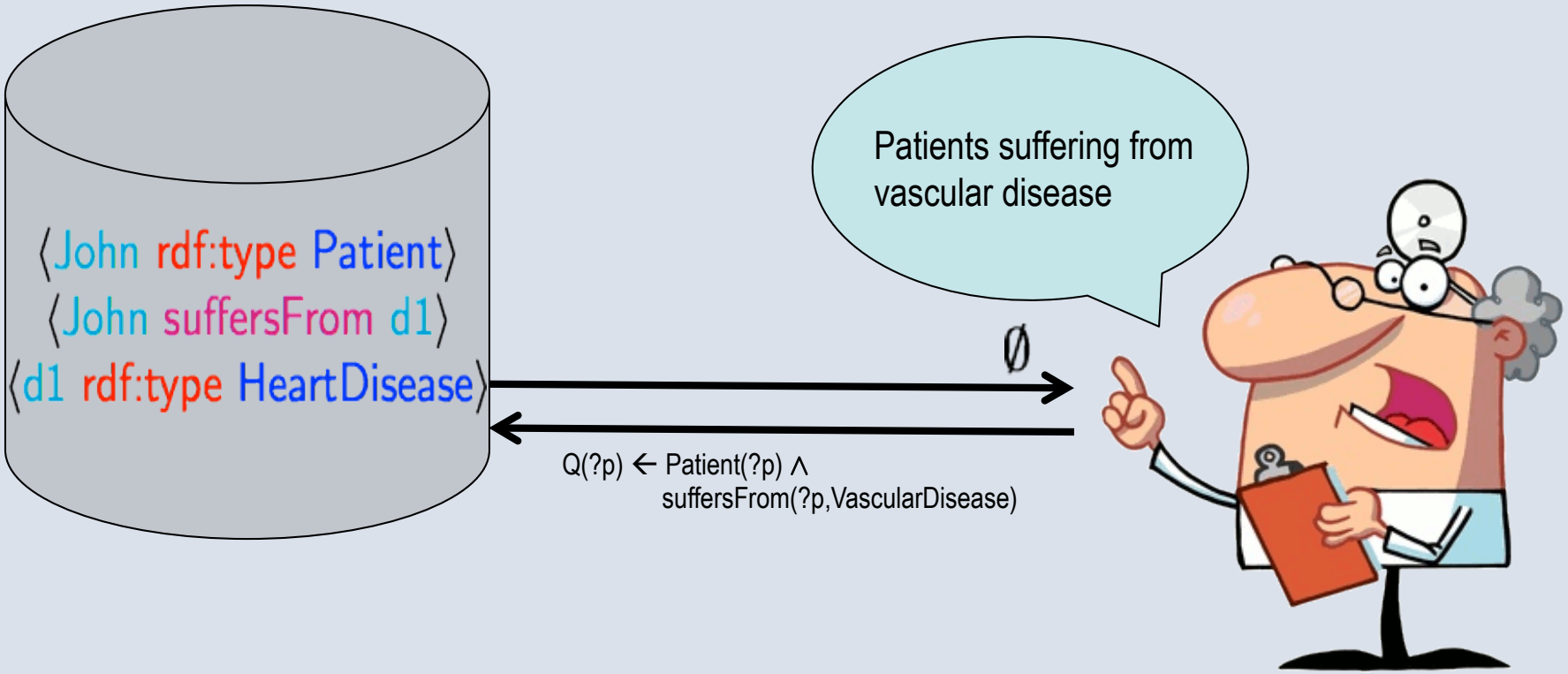
How Does it Work?



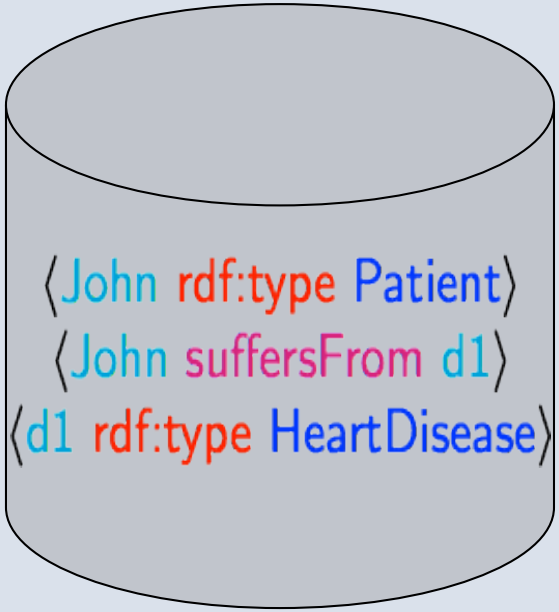
Q(?p) ← Patient(?p) ∧
suffersFrom(?p,VascularDisease)



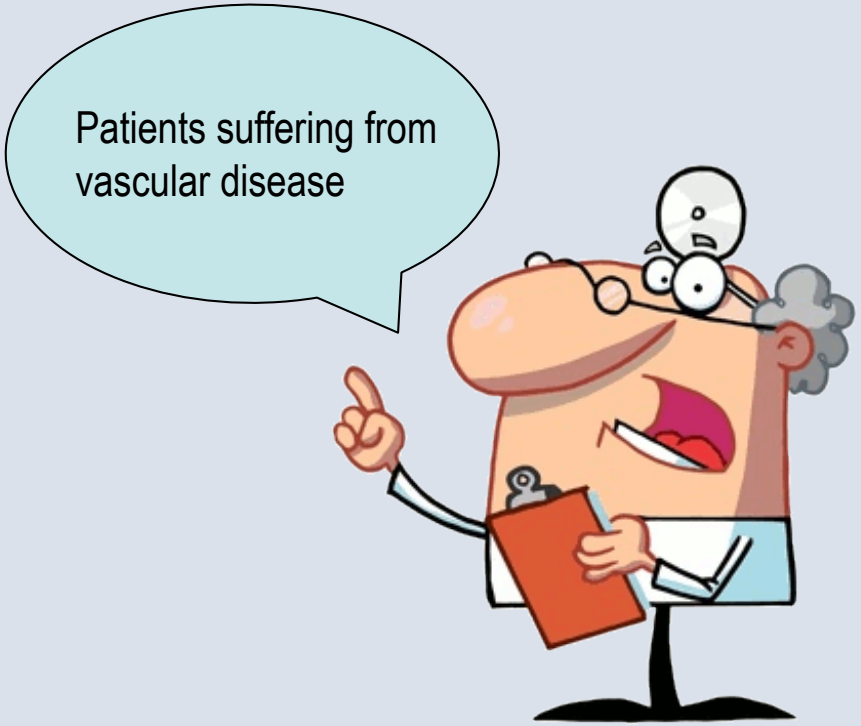
How Does it Work?



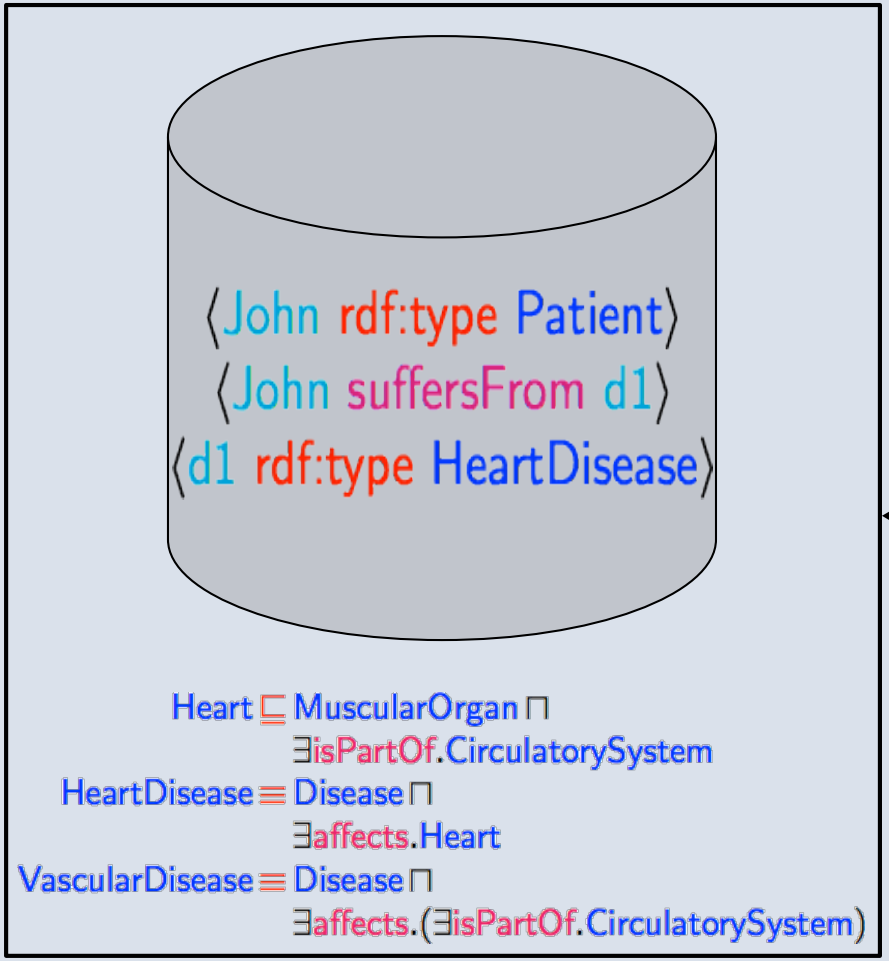
How Does it Work?



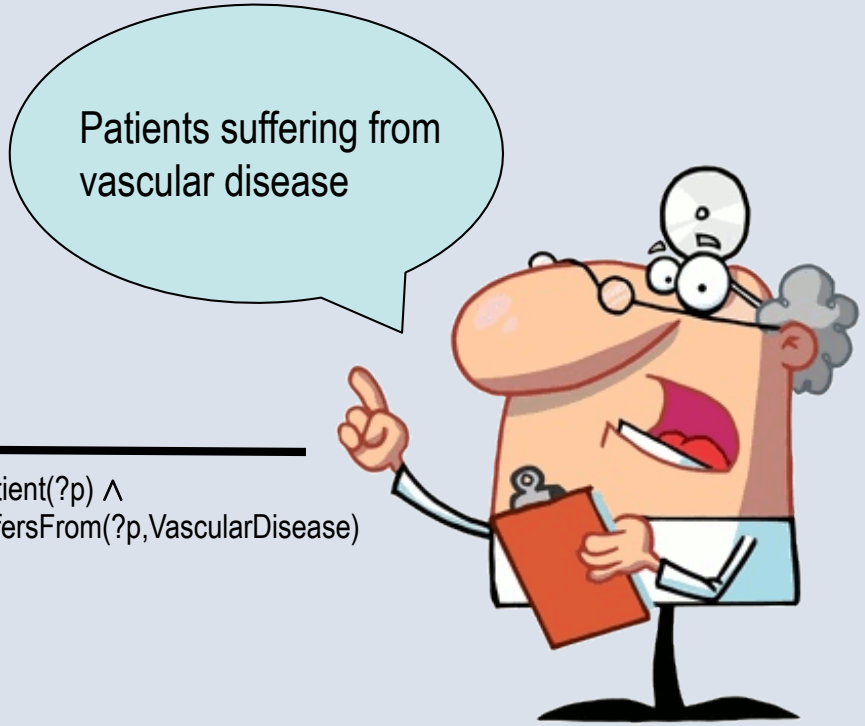
$$\begin{aligned}
\text{Heart} &\sqsubseteq \text{MuscularOrgan} \sqcap \\
&\quad \exists \text{isPartOf}.\text{CirculatorySystem} \\
\text{HeartDisease} &\equiv \text{Disease} \sqcap \\
&\quad \exists \text{affects}.\text{Heart} \\
\text{VascularDisease} &\equiv \text{Disease} \sqcap \\
&\quad \exists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



How Does it Work?

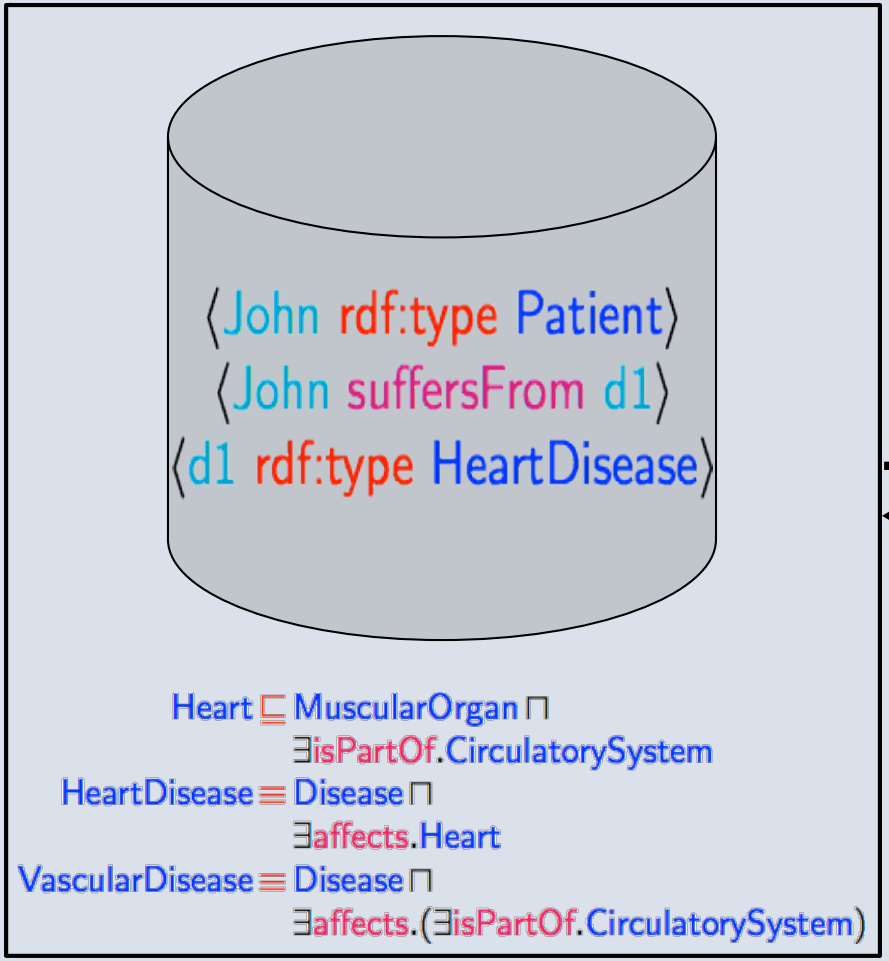


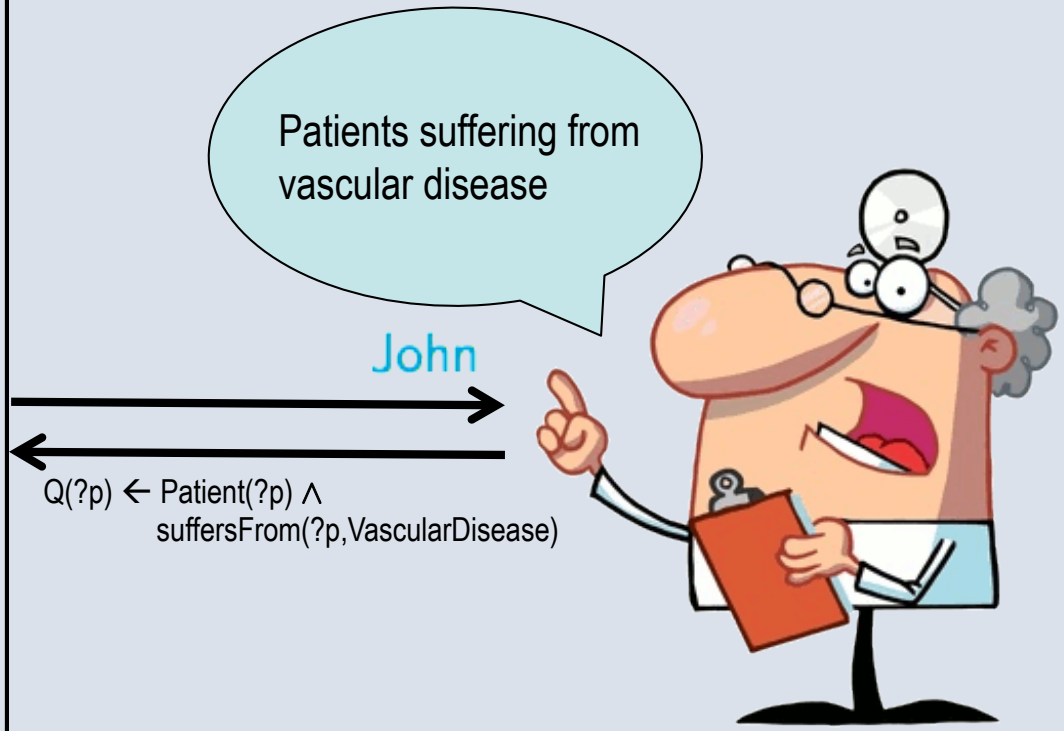
Q(?p) ← Patient(?p) ∧
suffersFrom(?p,VascularDisease)



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

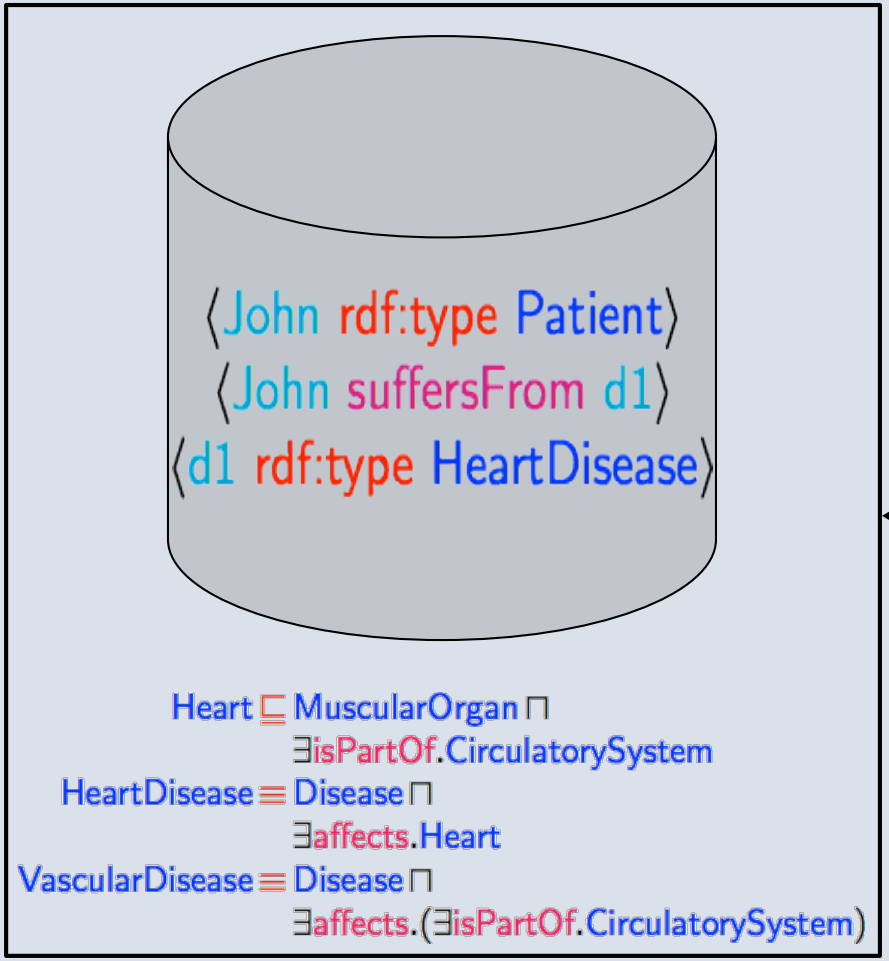
How Does it Work?



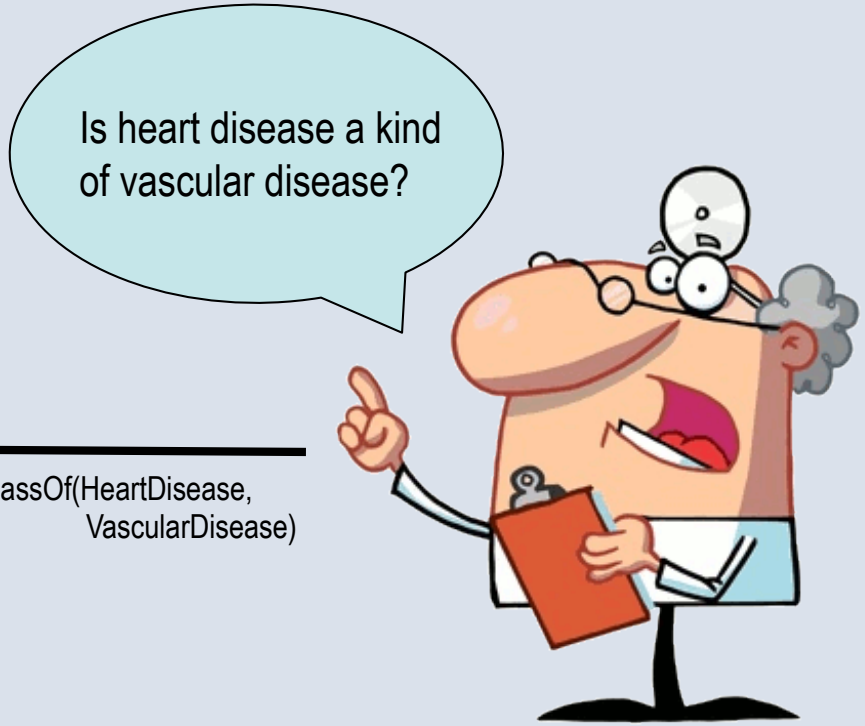


UNIVERSITY OF OXFORD
DEPARTMENT OF COMPUTER SCIENCE
Information Systems Group
EPSRC Engineering and Physical Sciences Research Council
Optique
SEVENTH FRAMEWORK PROGRAMME

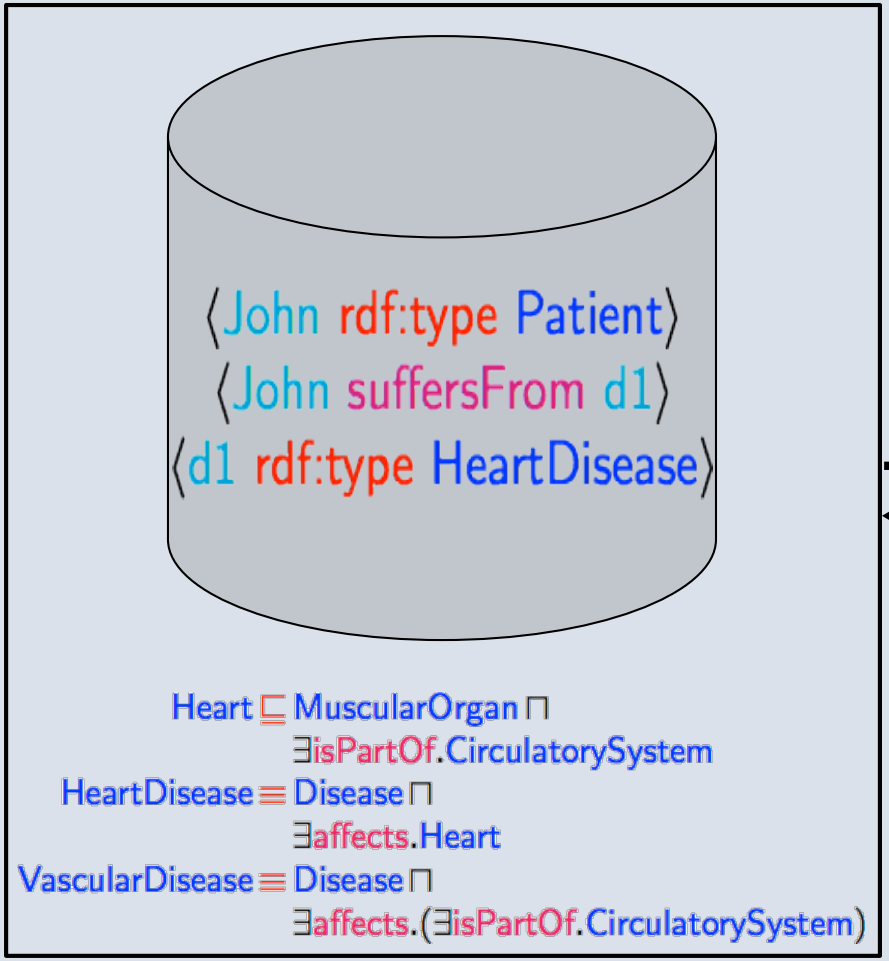
How Does it Work?

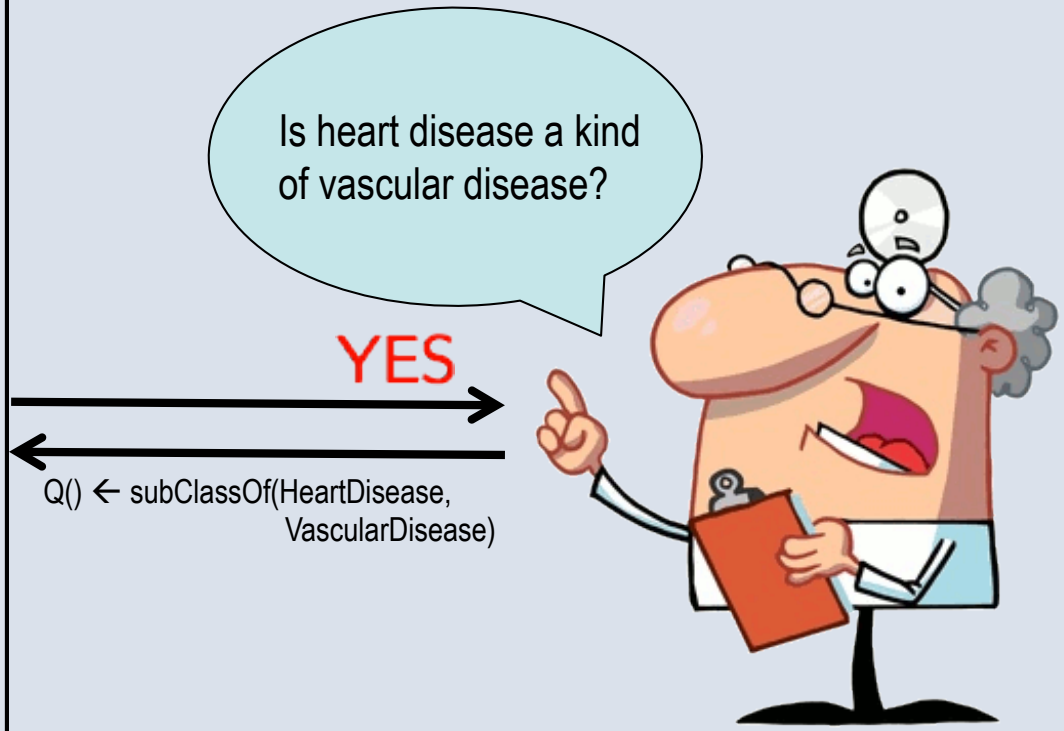


Q() ← subClassOf(HeartDisease, VascularDisease)



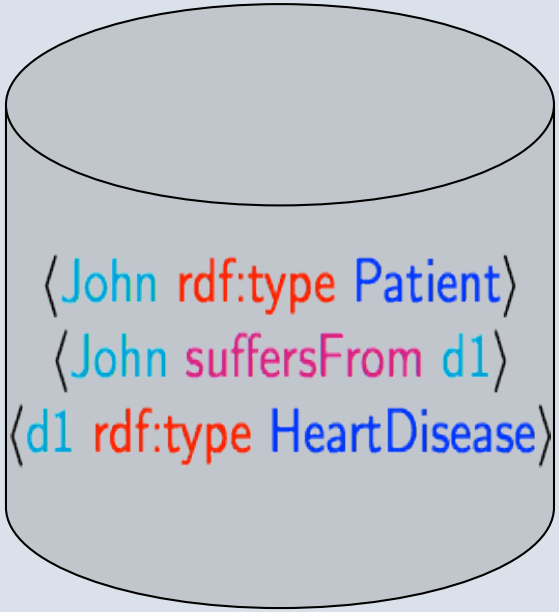
How Does it Work?





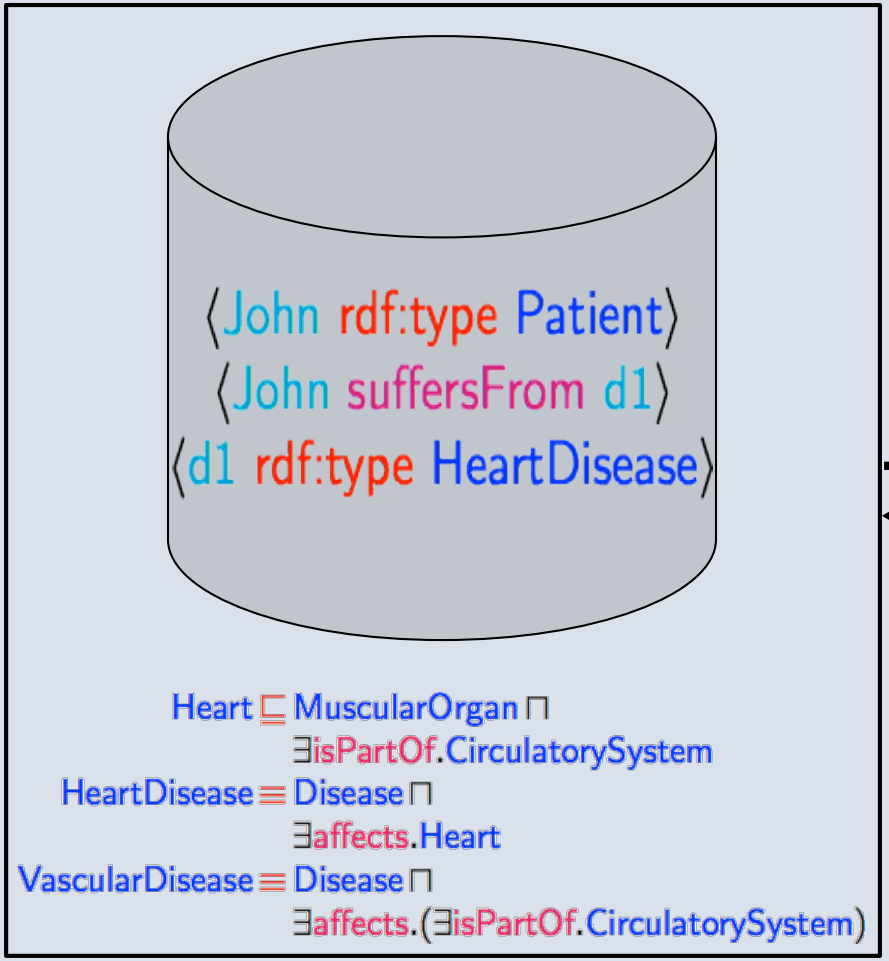
UNIVERSITY OF OXFORD
DEPARTMENT OF COMPUTER SCIENCE
Information Systems Group
EPSRC Engineering and Physical Sciences Research Council
Optique
SEVENTH FRAMEWORK PROGRAMME

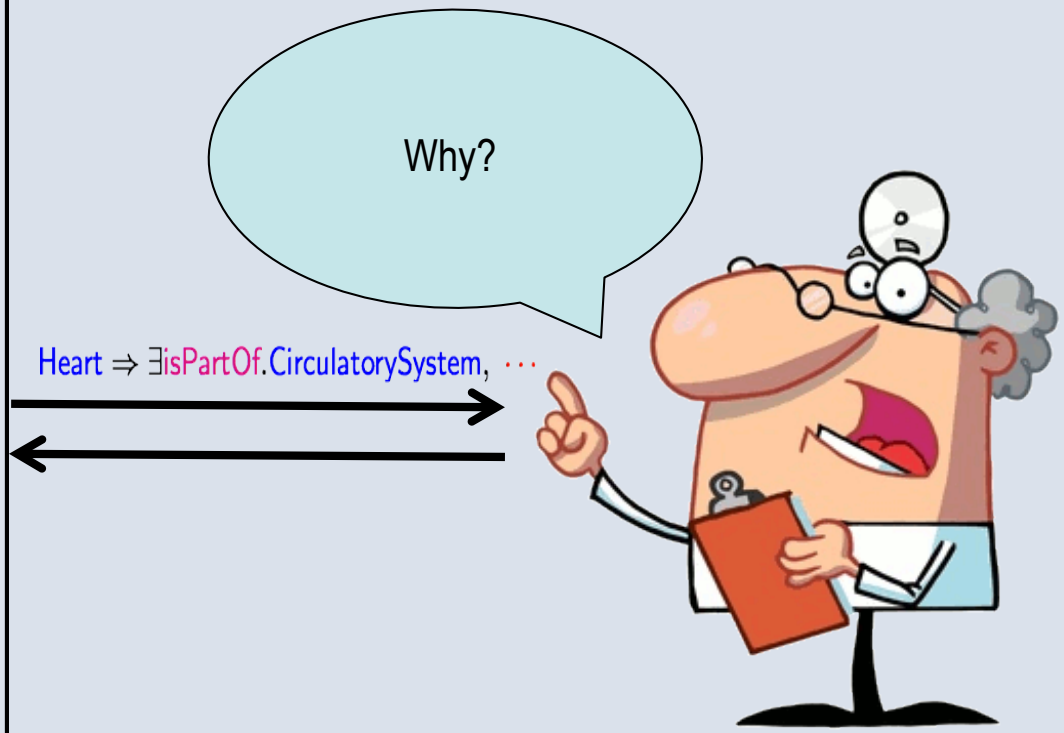
How Does it Work?



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

How Does it Work?

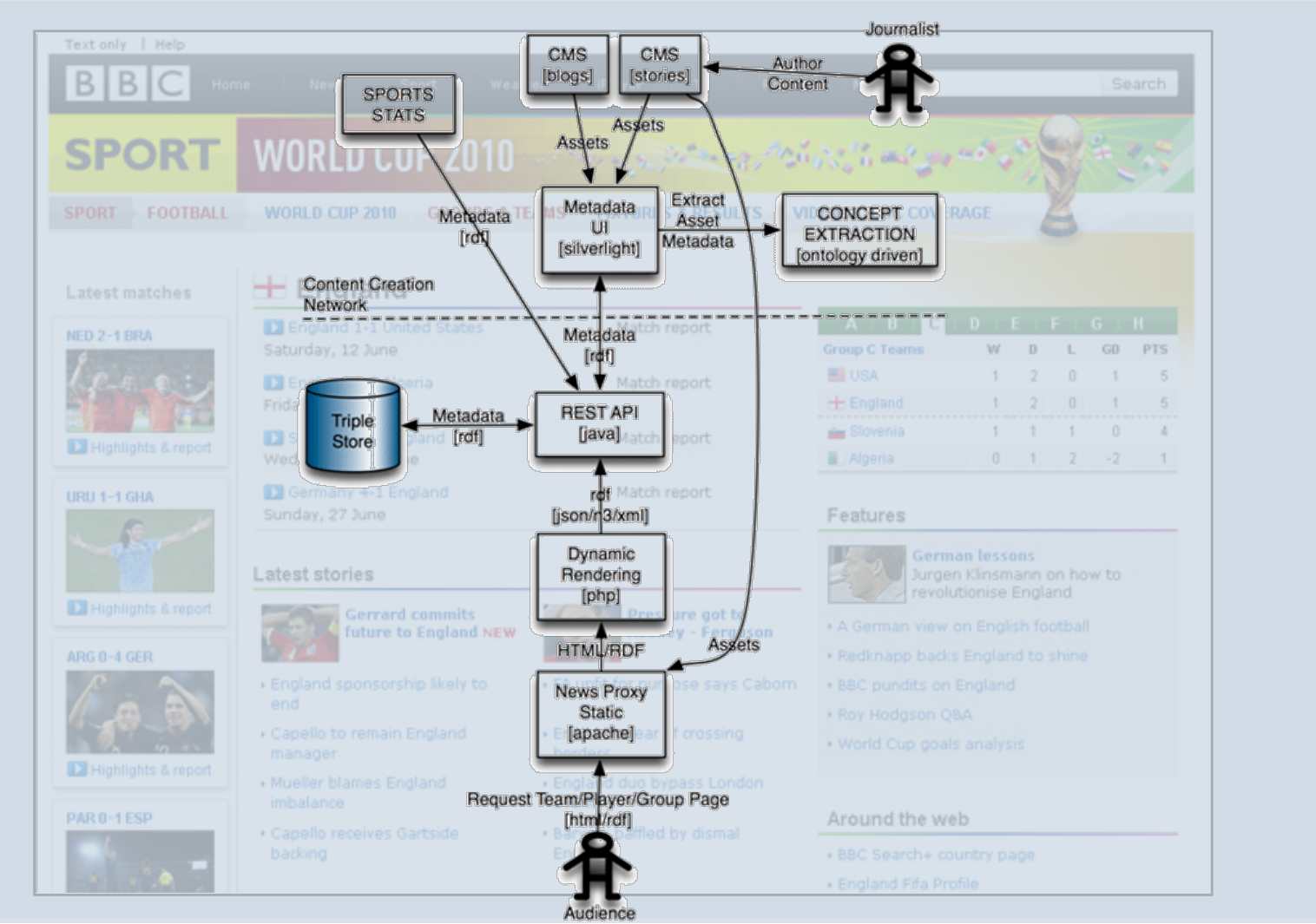




UNIVERSITY OF OXFORD DEPARTMENT OF COMPUTER SCIENCE
Information Systems Group
EPSRC Engineering and Physical Sciences Research Council
Optique
SEVENTH FRAMEWORK PROGRAMME

Applications: Semantic Web

Applications: Semantic Web



Applications: Semantic Web

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: 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

Theory $\rightsquigarrow$ Practice

Theory $\rightsquigarrow$ Practice

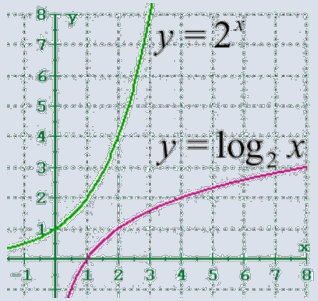
- OWL based on **description logic** $\mathcal{SROIQ}$

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 $\rightsquigarrow$ Practice

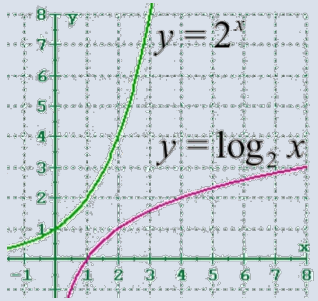
- OWL based on **description logic** $\mathcal{SROIQ}$
- DLs are a family of FOL fragments
  - Clear semantics
  - Well understood computational properties (e.g., decidability, complexity)
  - Simple goal directed reasoning algorithms
- OWL is decidable, but highly **highly intractable**
  - N2ExpTime-comlete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



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

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?

Various Approaches — Different Tradeoffs

❶ 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**" ($\approx \mathsf{DL} \cap \mathsf{LP}$)
  - PTime-complete for combined and data complexity

Various Approaches — Different Tradeoffs

❶ Use **EL** "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?

	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?

	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)
- Restrictions may result in Byzantine/incorrect modelling
  - e.g., in SNOMED, groin modelled as part of abdomen and of leg
- Single non-EL axiom may incur massive performance penalty

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).

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

MORe Modular Reasoner

Ontology	Classif. time (seconds)			
	HermiT	MORe		
		total	HermiT	ELK
GO	7.1	2.2 (↓69.0%)	0	0.1
Gazeteer	838.1	28.2 (↓96.6%)	0	15.6
NCI	84.1	28.6 (↓66.0%)	15.8	3.3
Protein	11.4	2.9 (↓74.6%)	0.4	0.9
Biomodels	741.4	575.6 (↓22.4%)	540.1	2.6
cellCycle	–	13.9 (–)	<0.1	4.9
NCI+MGED	1018.0	769.2 (↓24.4%)	605.5	2.4
NCI+CHEBI	116.6	34.0 (↓70.8%)	16.3	4.1
NCI+GO	110.0	37.6 (↓65.8%)	17.6	3.2
NCI+Mouse	93.7	31.0 (↓66.9%)	16.6	2.6

MORe Modular Reasoner

Currently improving our technique:

- Combining module extraction techniques with over approximation of ontology in a less expressive DL (EL/RL)
- Classification of over approximated ontology is fast, and is complete (but unsound) w.r.t. subsumption
- Significantly reduces number of subsumption checks delegated to powerful (but slower) reasoner

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 partial data 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

❶ Use **QL** "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

Various Approaches — Different Tradeoffs

❶ Use **QL** "profile" and specialised reasoner:

✓ Tractable query answering

✓ Reliable answers (for inputs in the profile)

✗ Restricted expressivity of the ontology language

✗ Reasoners reject inputs outside profile

OWL 2 QL ontology reasoners:

- E.g., Presto, 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)

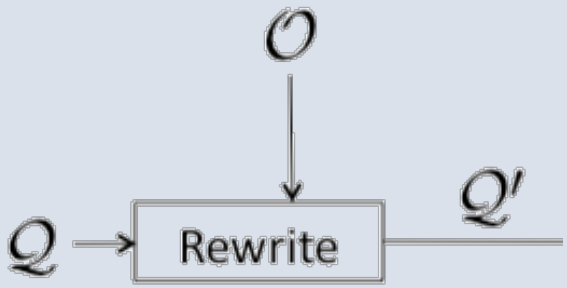
Query Rewriting — How Does It Work?

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

Query Rewriting — How Does It Work?

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

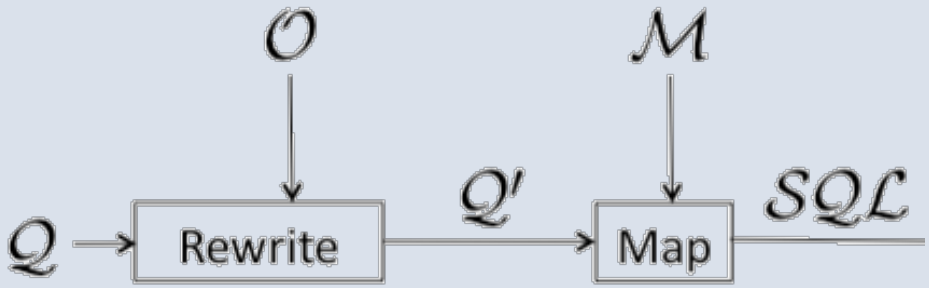
- **Rewrite** $\mathcal{Q} \rightarrow \mathcal{Q}'$ s.t. answering $\mathcal{Q}'$ without $\mathcal{O}$ equivalent to answering $\mathcal{Q}$ w.r.t. $\mathcal{O}$ *for any dataset*



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

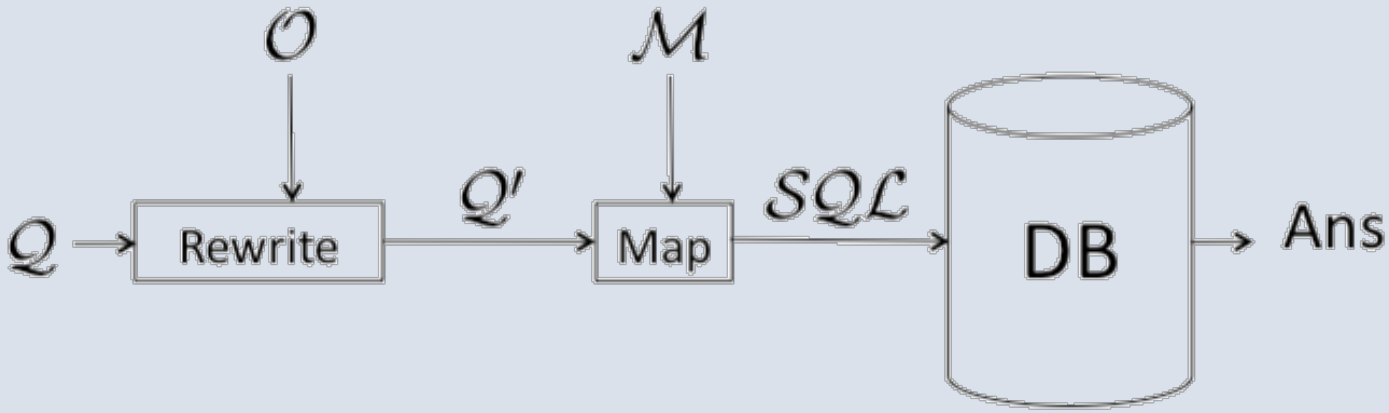


UNIVERSITY OF OXFORD DEPARTMENT OF COMPUTER SCIENCE
Information Systems Group
EPSRC Engineering and Physical Sciences Research Council
Optique
SEVENTH FRAMEWORK PROGRAMME

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*
- **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

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

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

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

Query Rewriting — Example

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

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

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

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

Query Rewriting — Example

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

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

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

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

Query Rewriting — Example

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

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

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

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

Query Rewriting — Example

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

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

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

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

$$\mathcal{SQL}\left\{\begin{array}{l}\text{SELECT Name FROM Doctor UNION}\\ \text{SELECT DName FROM Treats, Patient WHERE PName=Name}\end{array}\right.$$

Query Rewriting — Issues

1 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]

2 Mappings

- May be difficult to develop and maintain
- Relatively 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

❷ 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

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?

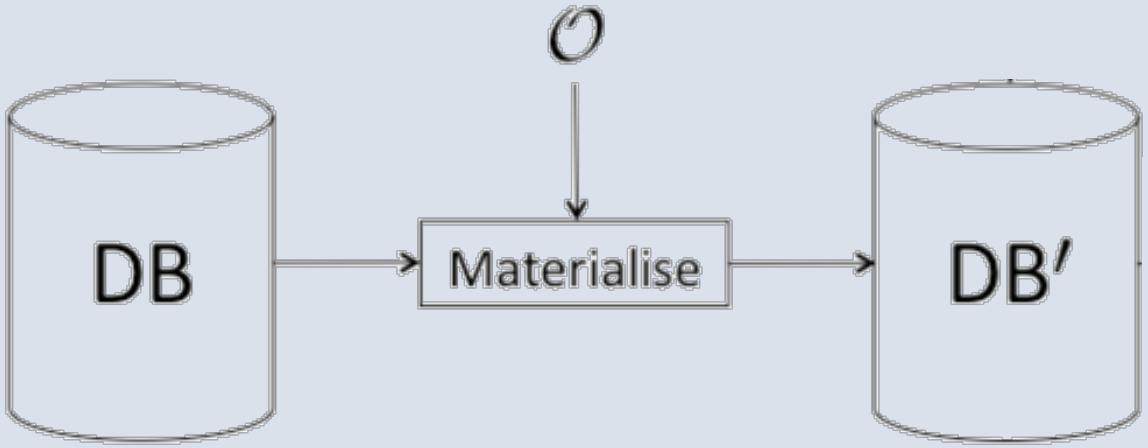
Given (RDF) data DB, ontology $\mathcal{O}$ and query $\mathcal{Q}$:

Materialisation — How Does It Work?

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

- **Materialise** (RDF) data DB → 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



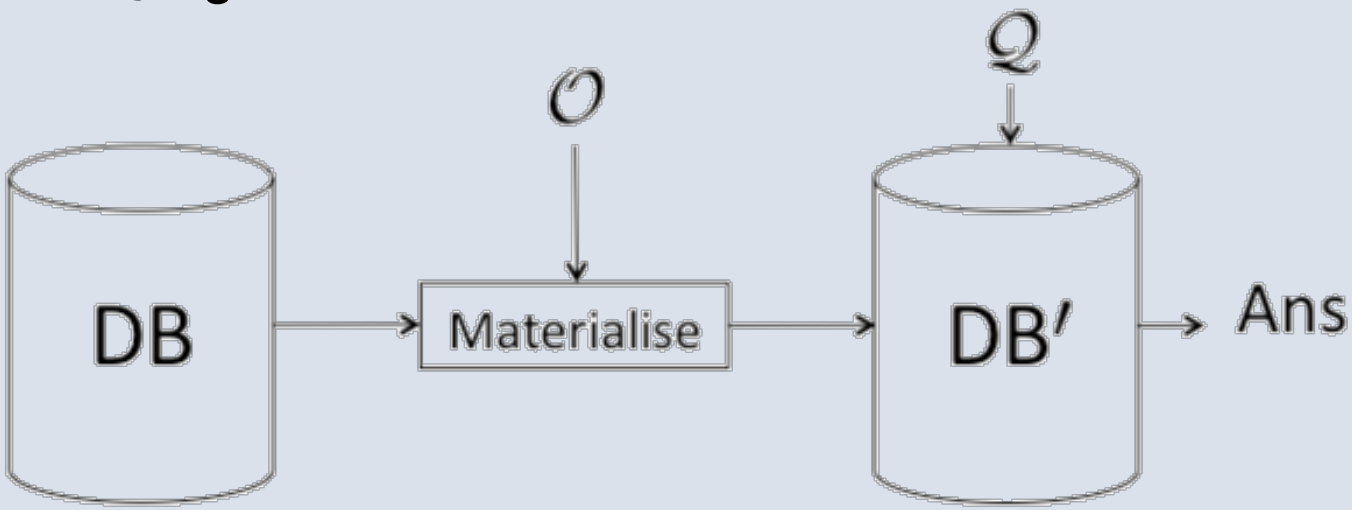
Materialisation — How Does It Work?

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

- **Materialise** (RDF) data DB → DB′ s.t. evaluating $\mathcal{Q}$ w.r.t. DB′ equivalent to answering $\mathcal{Q}$ w.r.t. DB and $\mathcal{O}$

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

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



Materialisation — Example

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

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

Materialisation — Example

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

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

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

Materialisation — Example

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

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

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

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

Materialisation — Example

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

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

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

Materialisation — Example

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

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

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

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

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 incomplete for ontologies **outside the profile**:
  - Incompleteness not evenly distributed
  - Incompleteness difficult to measure via empirical testing
  - Can lead to unsoundness, e.g., via aggregation
- Possible solutions offered by recent work:
  - **Measuring and repairing incompleteness**
  - **Chase materialisation**
  - **Computing upper and lower bounds**

Measuring and Repairing Incompleteness

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

Measuring and Repairing Incompleteness

- Use ontology $\mathcal{O}$ (and query $\mathcal{Q}$) to generate a test suite
- A **test suite** for $\mathcal{O}$ is a pair $\mathbf{S} = \langle \mathbf{S}_\perp, \mathbf{S}_Q \rangle$
  - $\mathbf{S}_\perp$ 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

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.

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

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

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

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

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

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

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

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

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

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

Skolems

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

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

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

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

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

Computing Lower and Upper Bounds

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

Computing Lower and Upper Bounds

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

Computing Lower and Upper Bounds

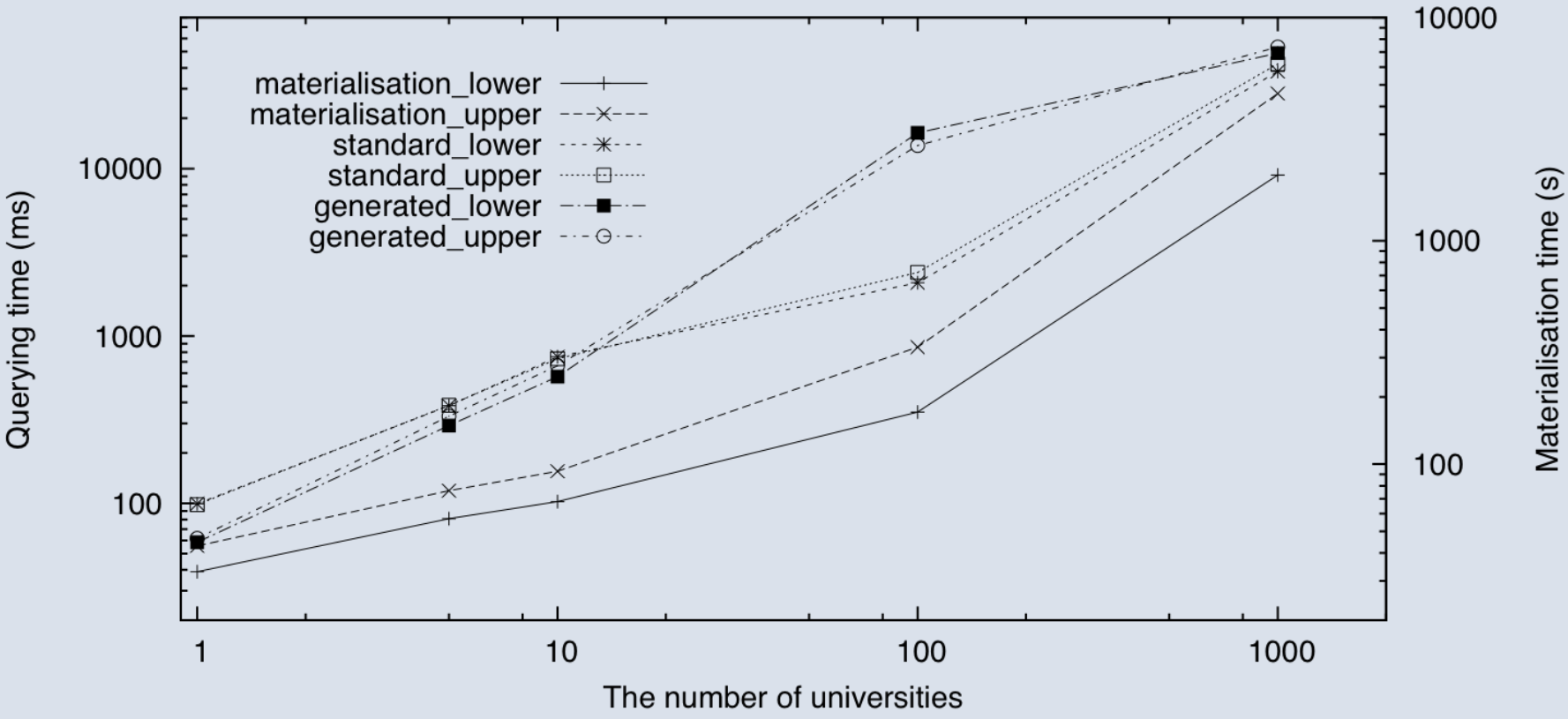
- RL reasonting w.r.t. $\mathcal{O}'$ gives (complete but unsound) upper bound answer $\boldsymbol{U}$

Computing Lower and Upper Bounds

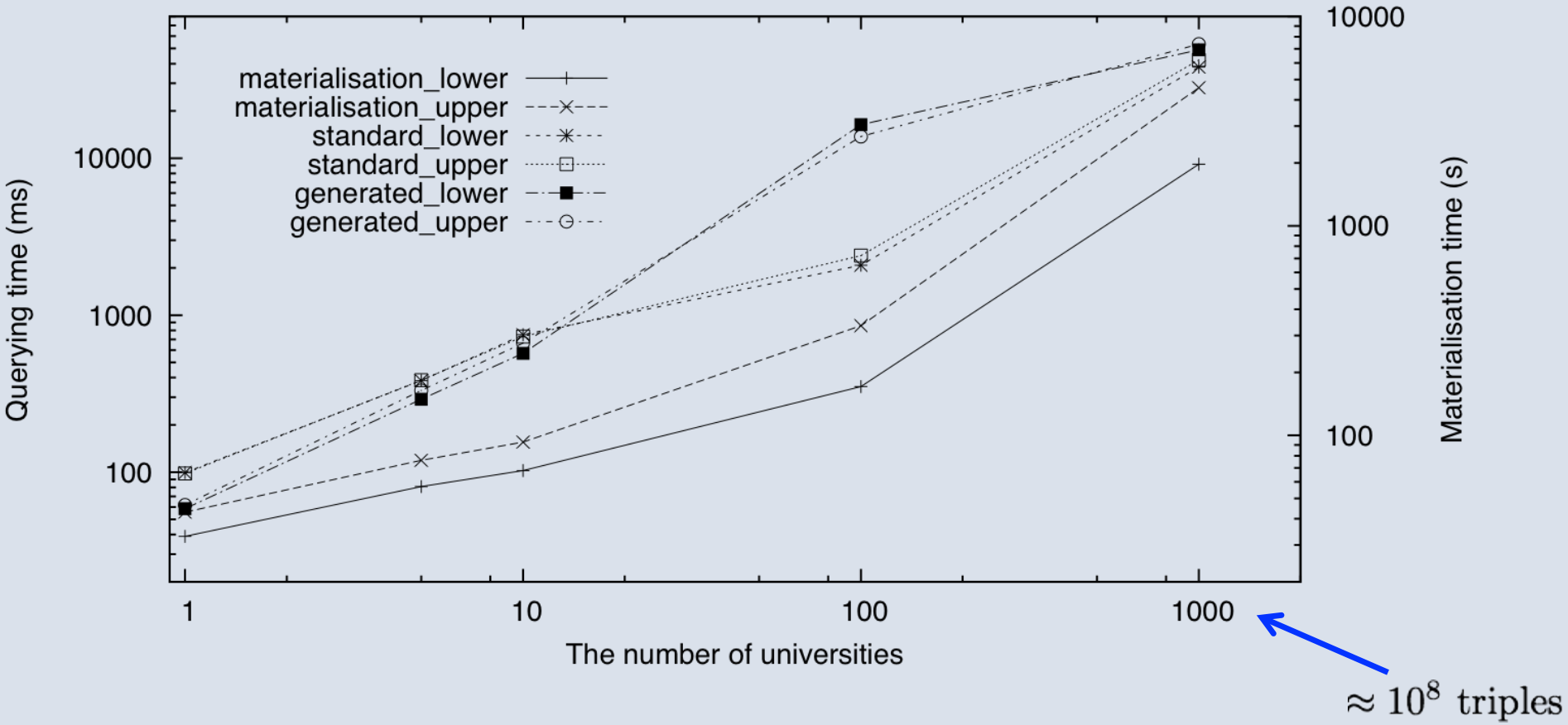
- RL reasonting w.r.t. $\mathcal{O}'$ gives (complete but unsound) upper bound answer ***U***
- If ***L*** = ***U***, then both answers are sound and complete
- If ***L*** ≠ ***U***, then ***U*** \ ***L*** identifies a (small) set of "possible" answers
  - Delineates range of uncertainty
  - Can (more 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. Making the Most of your Triple Store: Query Answering in OWL 2 Using an RL Reasoner. In Proc. of WWW 2013.

Performance on LUBM

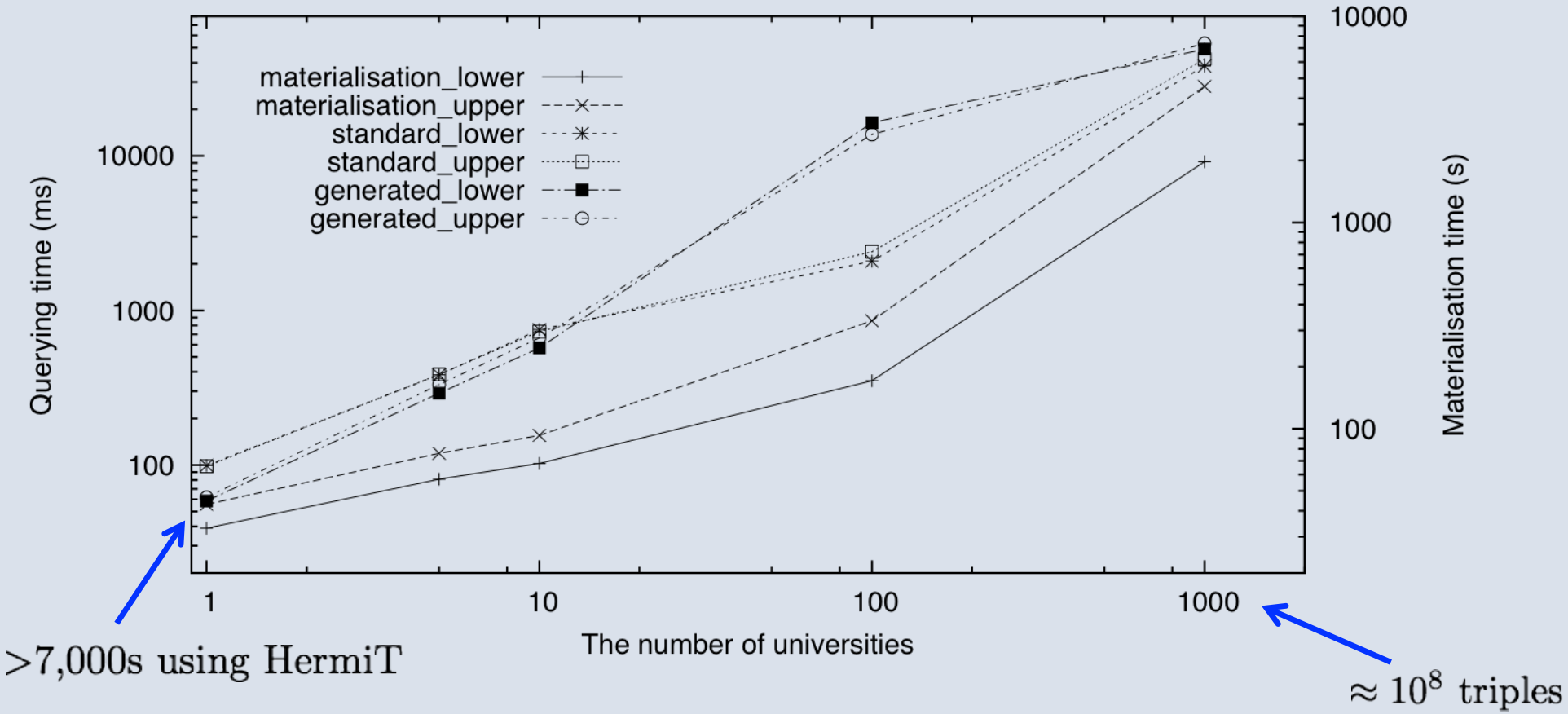


Performance on LUBM



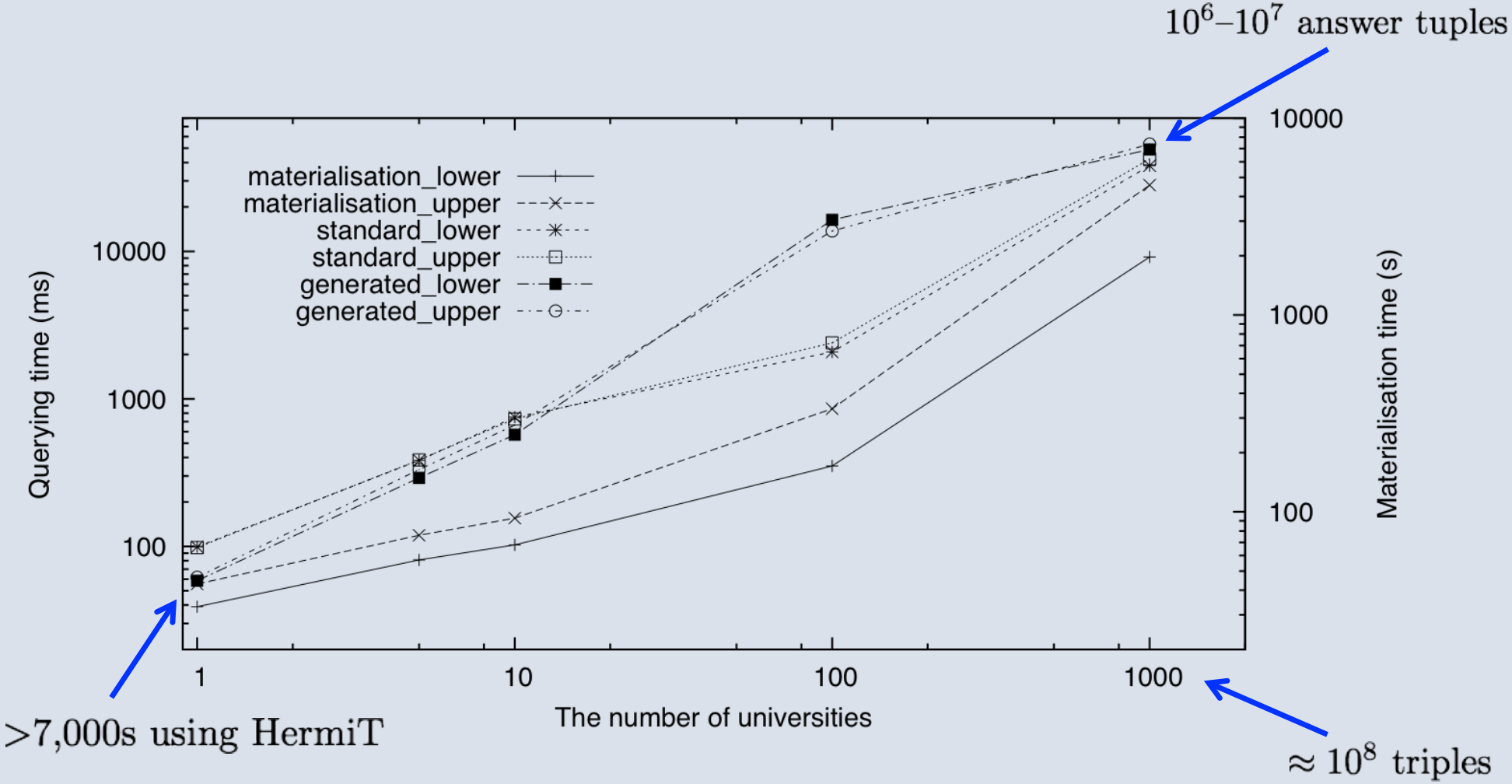
UNIVERSITY OF OXFORD
DEPARTMENT OF COMPUTER SCIENCE
Information Systems Group
EPSRC
Engineering and Physical Sciences Research Council
Optique
SEVENTH FRAMEWORK PROGRAMME

Performance on LUBM

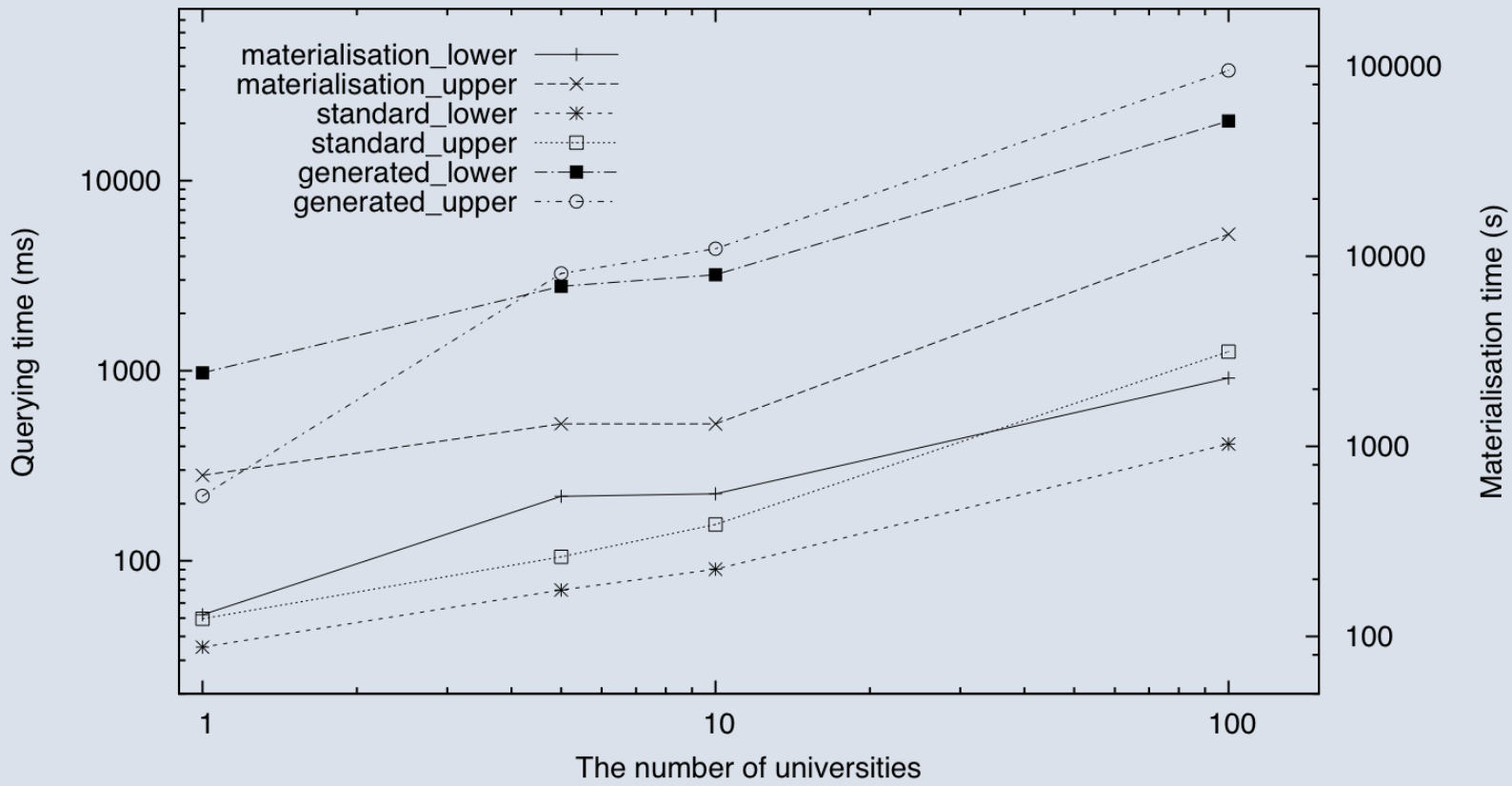


UNIVERSITY OF OXFORD
DEPARTMENT OF COMPUTER SCIENCE
Information Systems Group
EPSRC
Engineering and Physical Sciences Research Council
Optique
SEVENTH FRAMEWORK PROGRAMME

Performance on LUBM

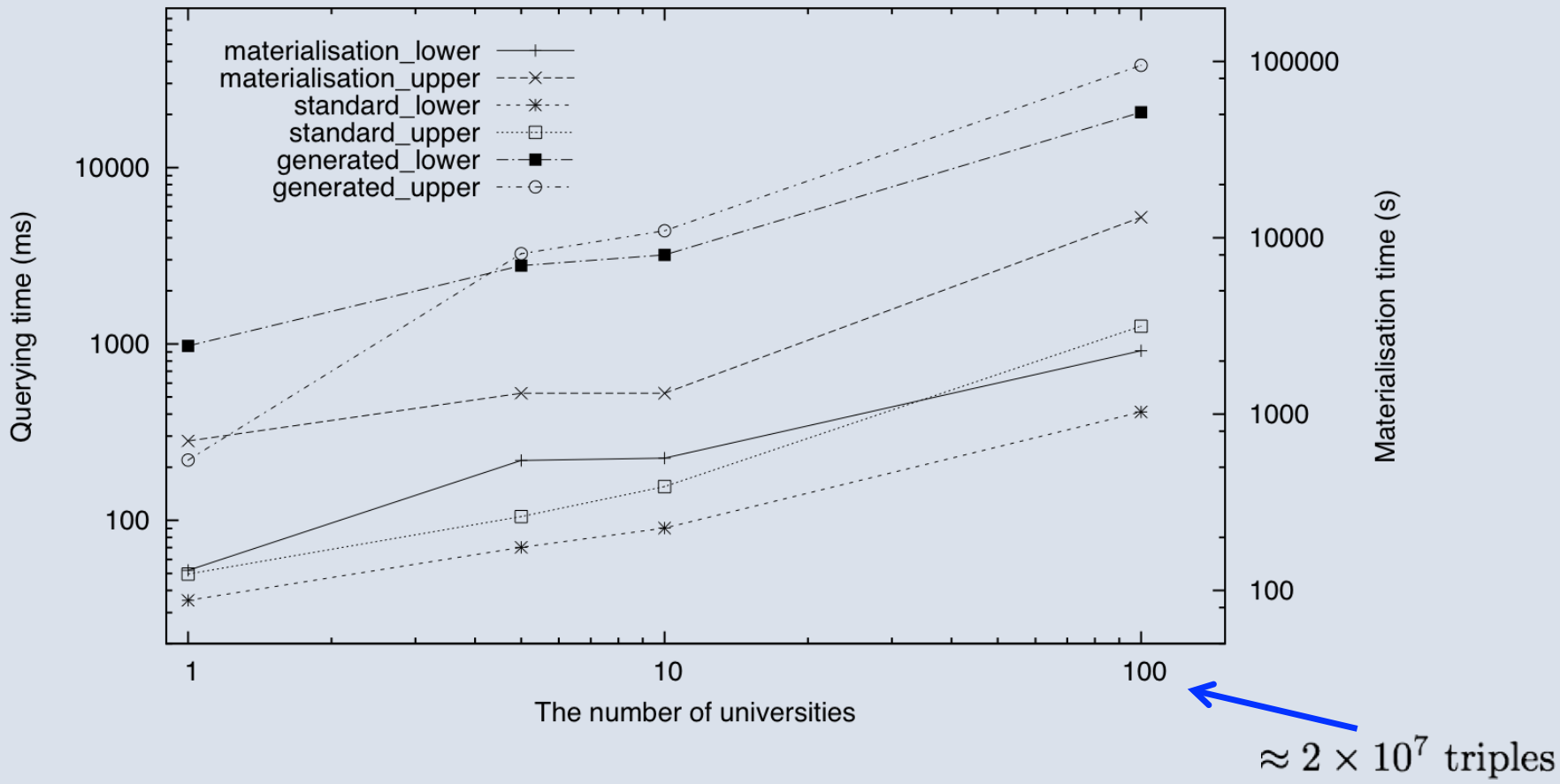


Performance on UOBM



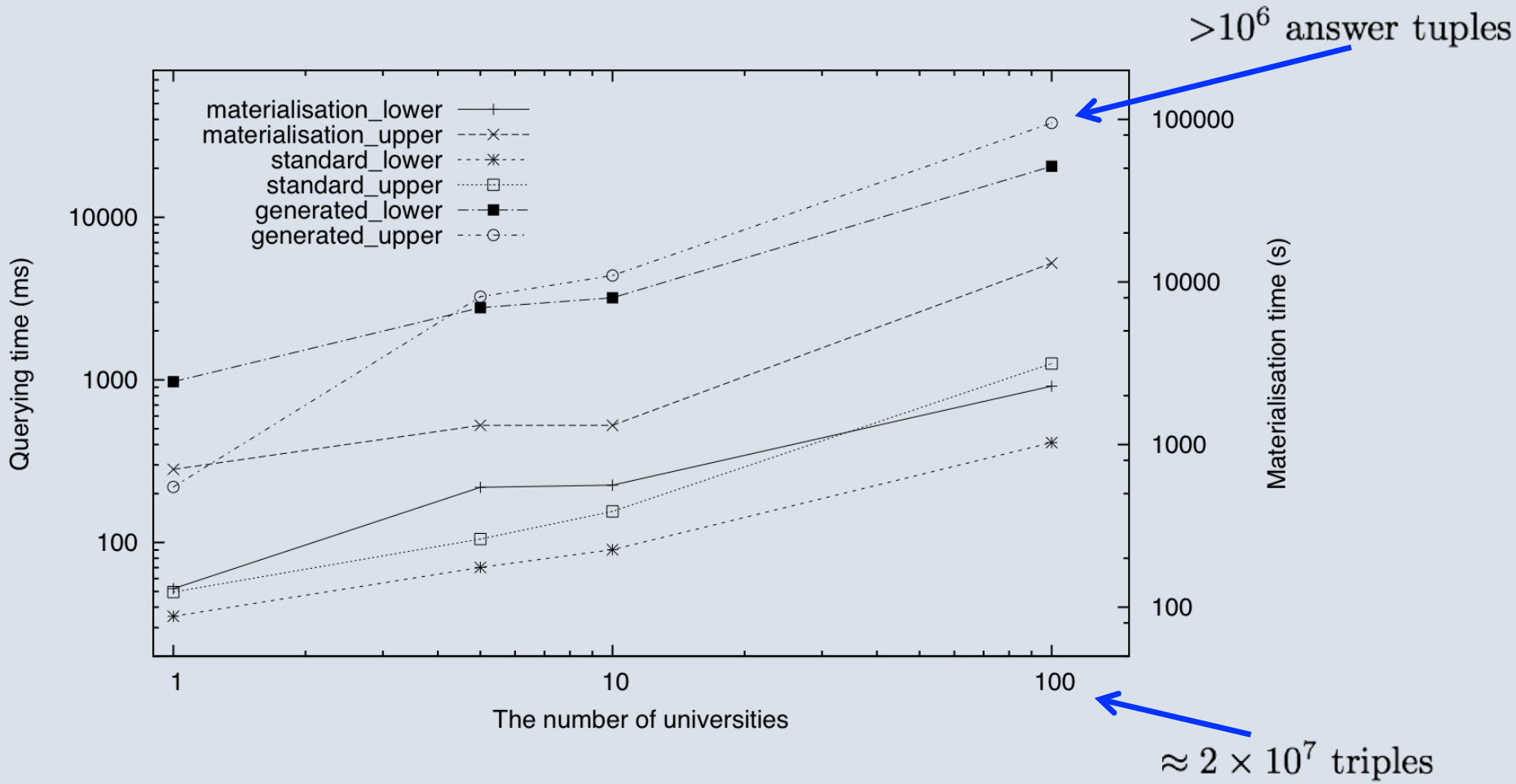
UNIVERSITY OF OXFORD
DEPARTMENT OF COMPUTER SCIENCE
Information Systems Group
EPSRC
Engineering and Physical Sciences Research Council
Optique
SEVENTH FRAMEWORK PROGRAMME

Performance on UOBM



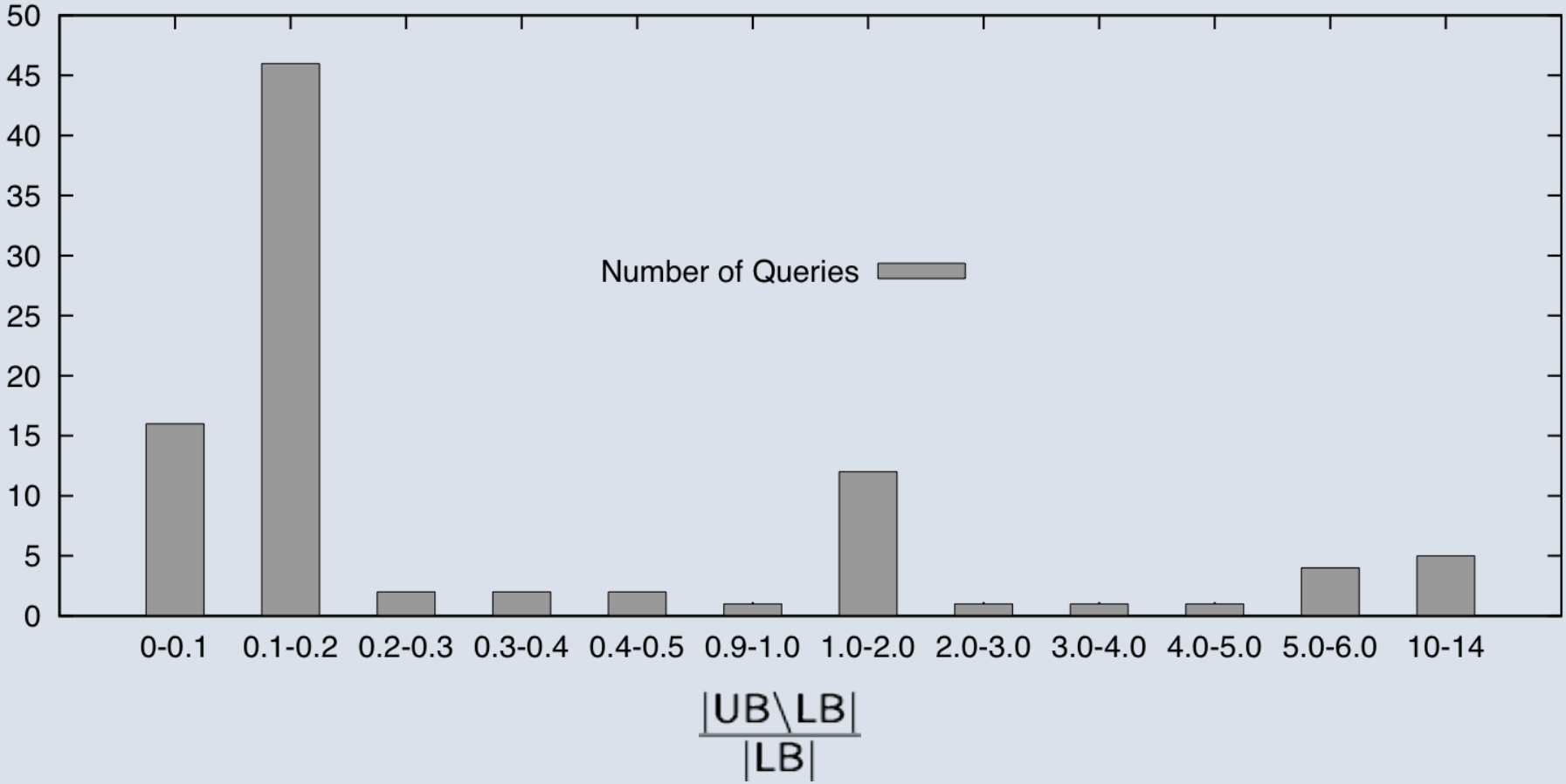
UNIVERSITY OF OXFORD
DEPARTMENT OF COMPUTER SCIENCE
Information Systems Group
EPSRC Engineering and Physical Sciences Research Council
Optique
SEVENTH FRAMEWORK PROGRAMME

Performance on UOBM



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

Gap Between LB and UB



Scalability: data storage

- Choice of architectures:
  - distributed memory – highly scalable, but reasoning is problematical
  - single machine + secondary storage – highly scalable, but access slow
  - single machine + in memory storage – fast, but limited scalability

- In memory, fully indexed, efficient but compressed data structures:
  - loading speed: 0.5 million triples per second
  - £4,000 desktop with 128 GB can store more than 2×10^9 triples
    - LUBM1000: 34b/t (48b/t including IRIs)
    - Claros: 40b/t (64b/t including IRIs)

Scalability: parallelization

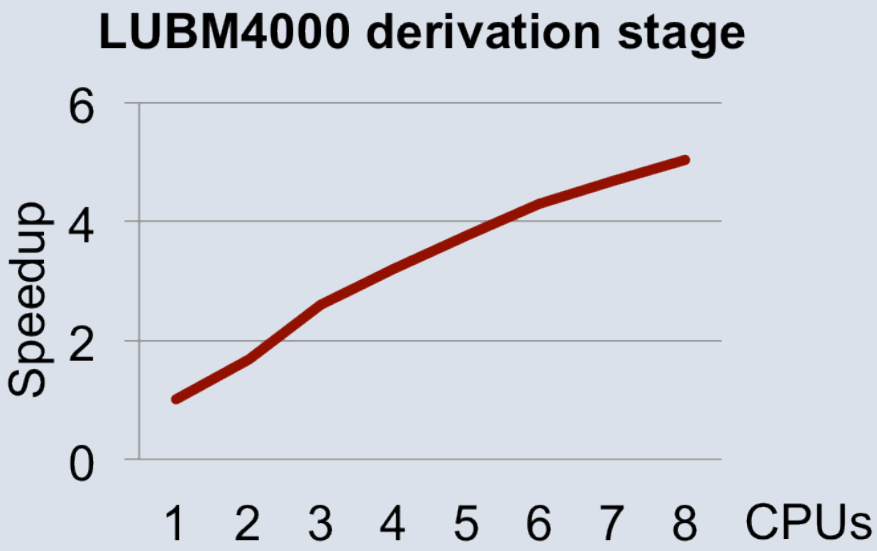
- Parallel data storage tasks:
  - import, materialization, querying
- Map reduce approaches not well suited to reasoning
  - high communication overhead
  - redundant computation
  - query answering over (distributed) materialized data is problematical
- Multiple processors with shared memory architecture
  - no communication overhead
  - linear Datalog fragment theoretically parallelizable
  - data-driven parallel evaluation of general Datalog possible (parallelization depends on the data)

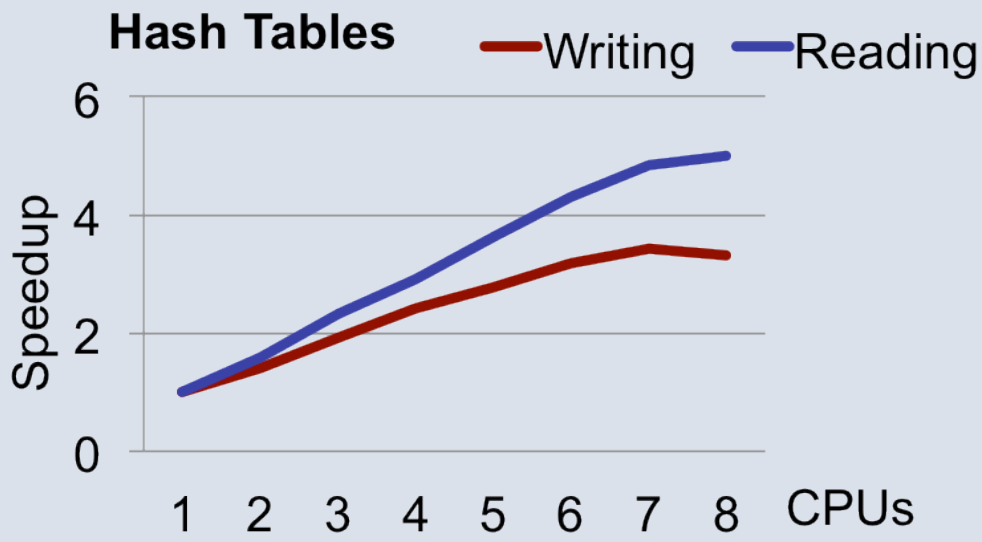
Scalability: parallel materialization

- Materialization: derivation stage + committing stage

Scalability: parallel materialization

- Materialization: derivation stage + committing stage
- Derivation stage: A prototype achieves 90% parallelization
- Committing stage (work in progress)
  - bottleneck: locking of hash tables
  - solution: fine-grained locking (striped hash tables)





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.

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

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

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

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**:

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 DBs: stores, main-memory, graph, RDF, etc.
- Semantic complexity, query optimisation 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

UNIVERSITY OF OXFORD DEPARTMENT OF COMPUTER SCIENCE

Information Systems Group

EPSRC Engineering and Physical Sciences Research Council

Optique

SEVENTH FRAMEWORK PROGRAMME

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

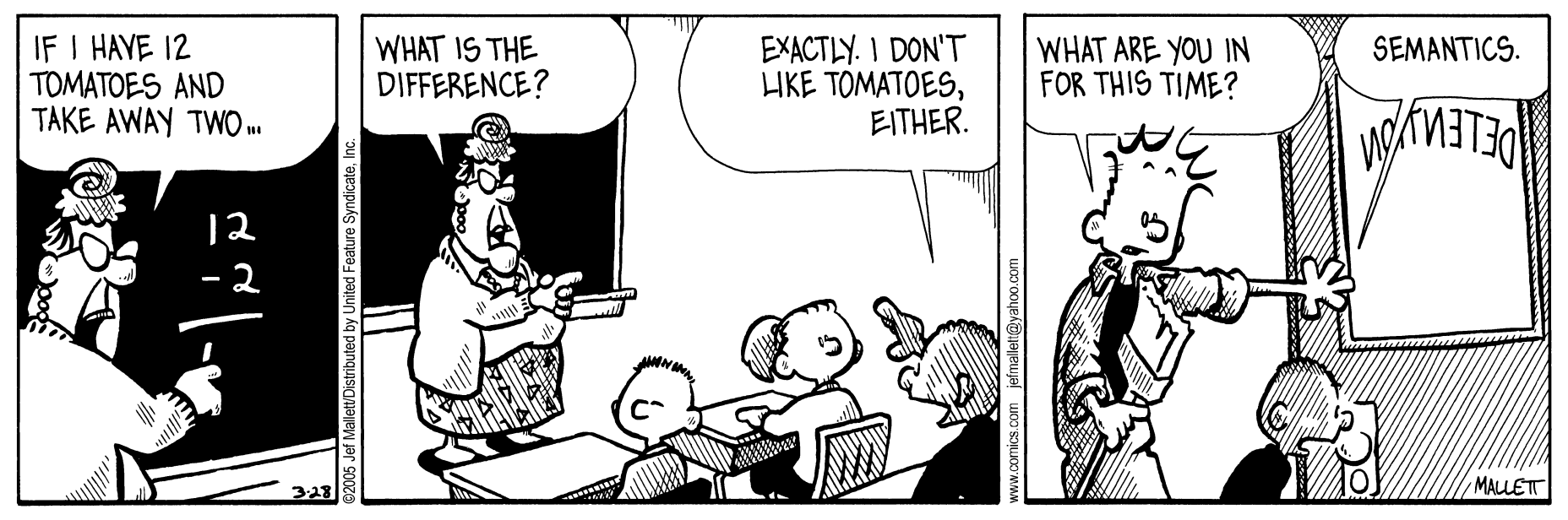
[6] Motik, Horrocks, and Kim. Delta-reasoner: a semantic web reasoner for an intelligent mobile platform. In Proc. of WWW 2012.

[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. Making the Most of your Triple Store: Query Answering in OWL 2 Using an RL Reasoner. In Proc. of WWW 2013.

Thank you for listening



FRAZZ: © Jeff Mallett/Dist. by United Feature Syndicate, Inc.

Any questions?

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