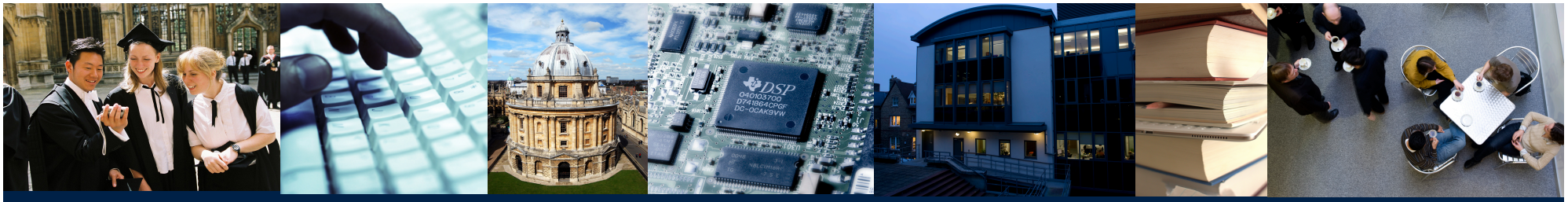
DEPARTMENT OF
COMPUTER
SCIENCE

UNIVERSITY OF
OXFORD



Semantics $\sqcap$ 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"

How Does it Work?

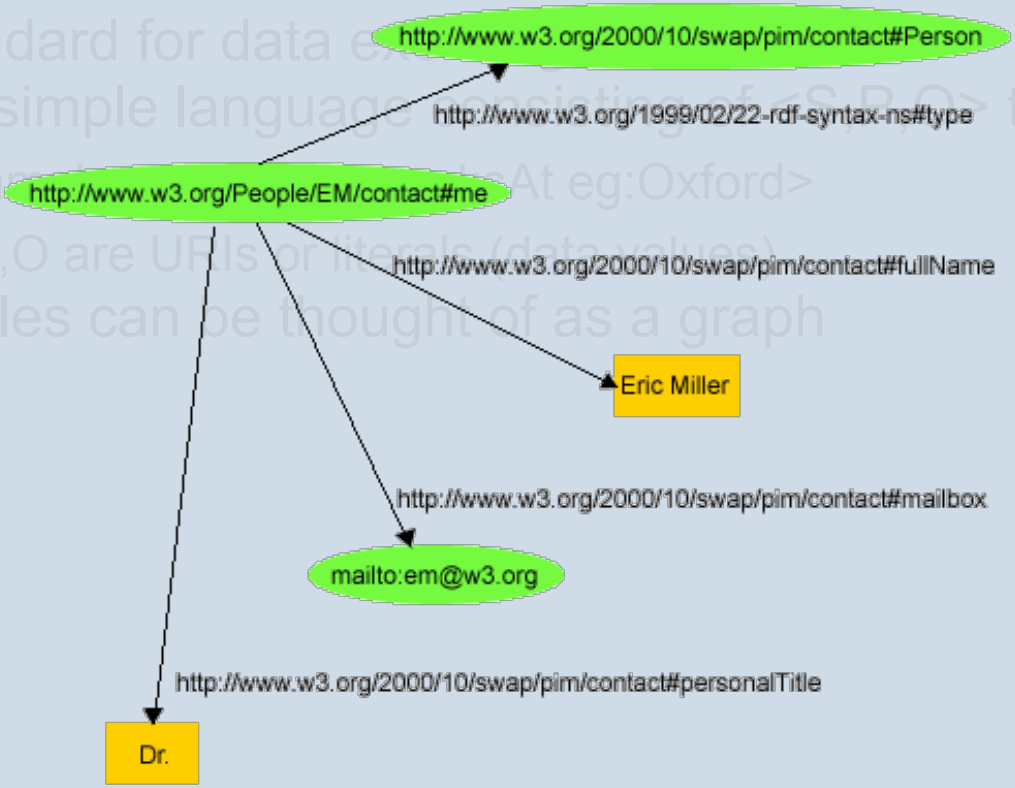
❶ Standardised language for exchanging data

- 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)
- Set of triples can be thought of as a graph

How Does it Work?

❶ Standardised language for exchanging data

- W3C standard for data exchange
- RDF is a simple language consisting of <S,P,O> triples
  - for example <eg:EricMiller eg:hasAt eg:Oxford>
  - all S,P,O are URIs or literals (data values)
- Set of triples can be thought of as a graph



How Does it Work?

❷ **Standardised language for exchanging schemas**

- Use of URIs provides a flexible **naming** scheme

How Does it Work?

❷ Standardised language for exchanging schemas

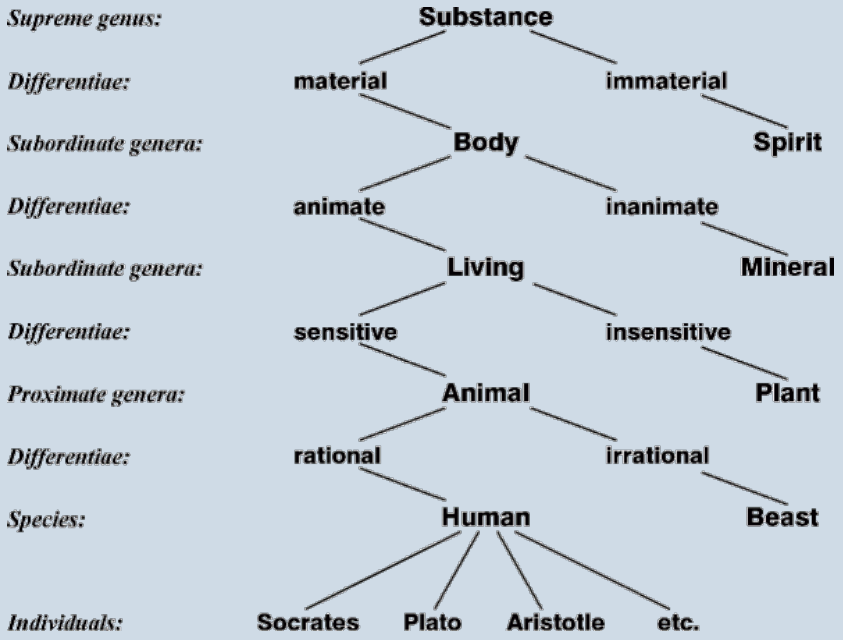
- Use of URIs provides a flexible **naming** scheme
- Also need to exchange **semantics** of data
- RDF(S) provides basic capabilities
  - subclass, subproperty, range and domain
- **OWL** is W3C standard for schema exchange
  - rich language for conceptual schemas, aka **ONTOLOGIES**

What is an Ontology?

What is an Ontology?

A fundamental branch of **metaphysics**

- Studies “being” or “existence” and their **basic categories**
- Aims to find out what **entities** and **types of entities** exist



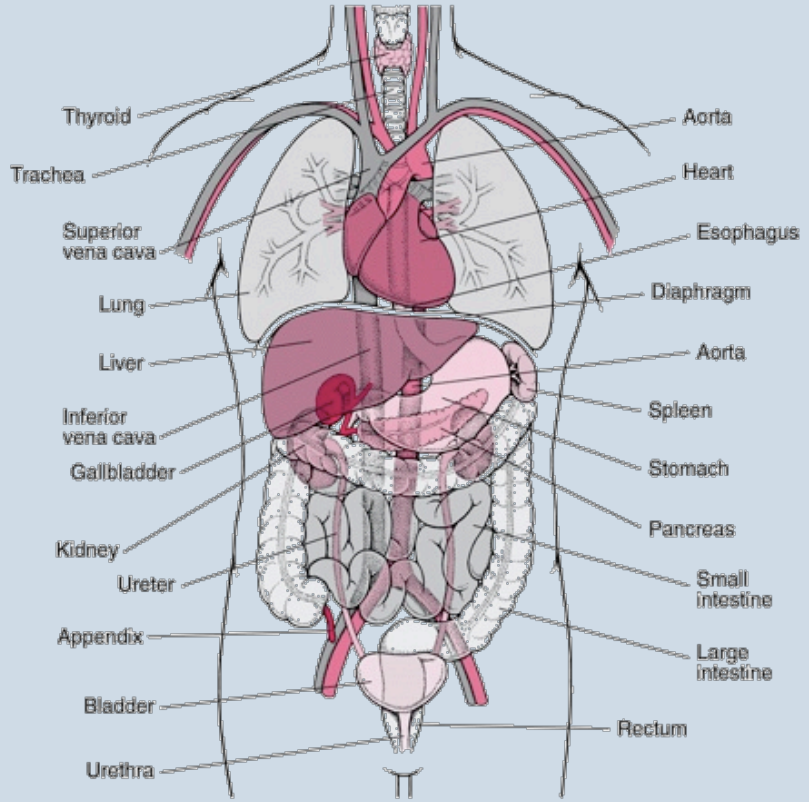
What is an Ontology?

A model of (some aspect of) the world

What is an Ontology?

A model of (some aspect of) the world

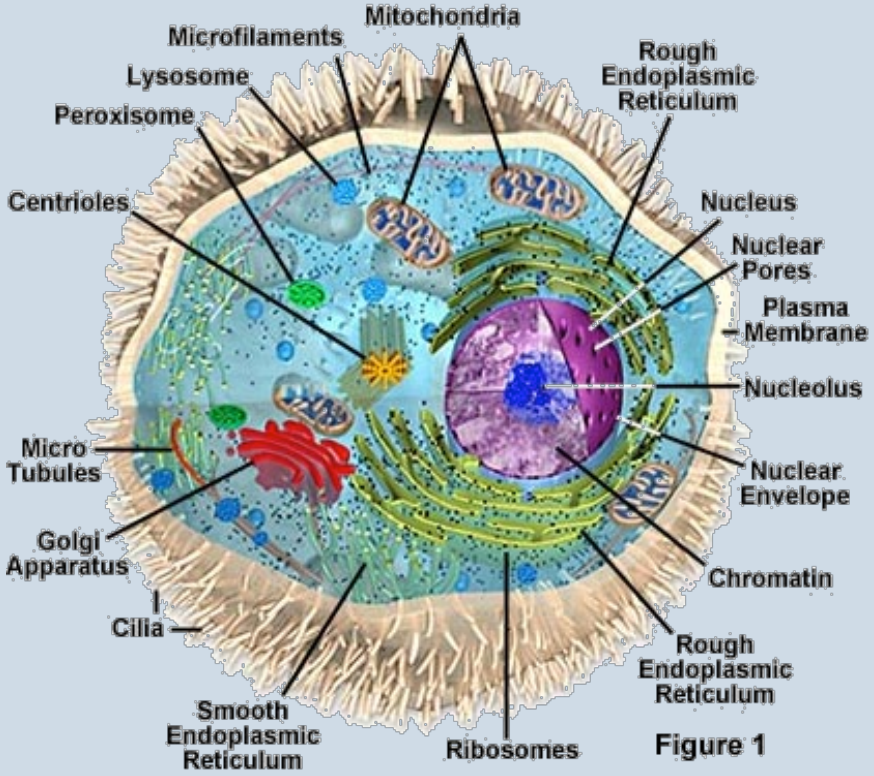
- Introduces **vocabulary** relevant to domain, e.g.:
  - Anatomy



What is an Ontology?

A model of (some aspect of) the world

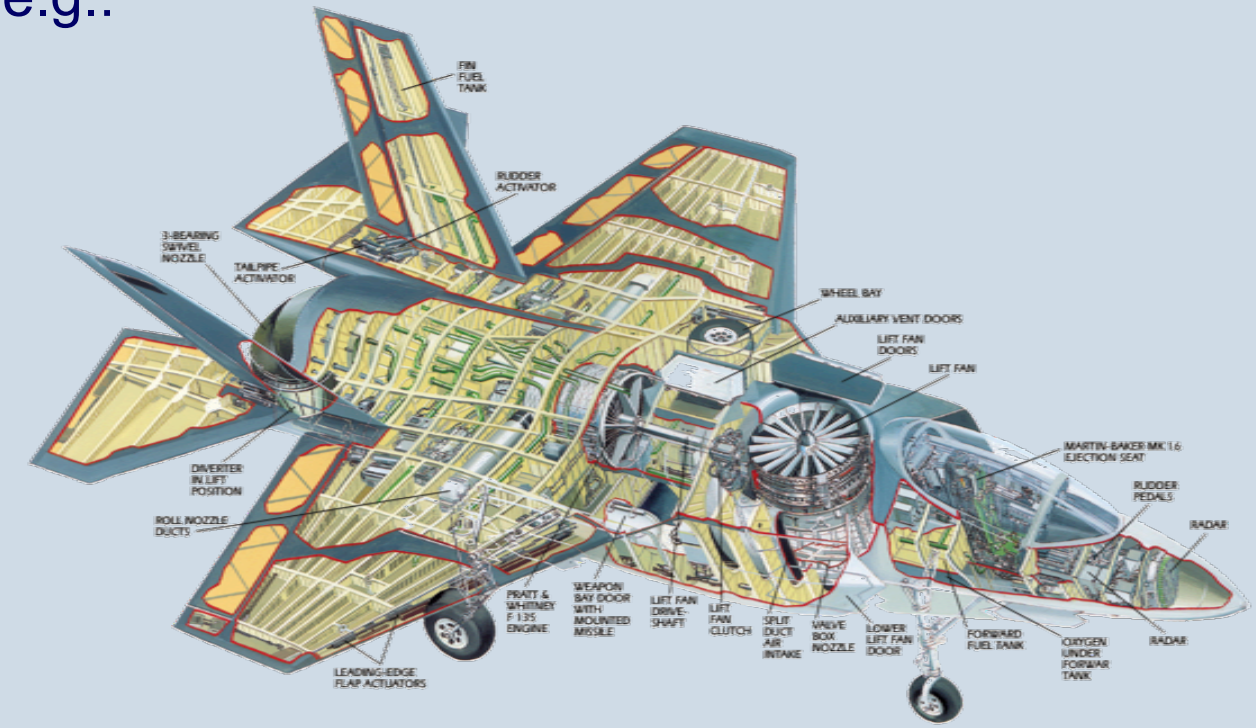
- Introduces **vocabulary** relevant to domain, e.g.:
  - Anatomy
  - Cellular biology



What is an Ontology?

A model of (some aspect of) the world

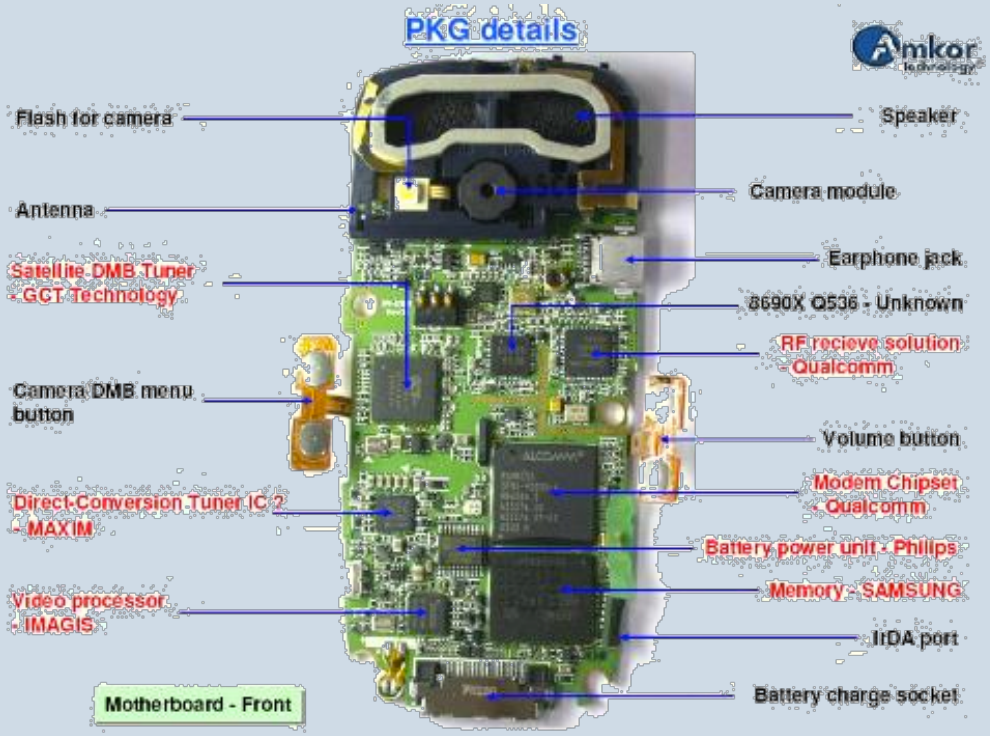
- Introduces **vocabulary** relevant to domain, e.g.:
  - Anatomy
  - Cellular biology
  - Aerospace



What is an Ontology?

A model of (some aspect of) the world

- Introduces **vocabulary** relevant to domain, e.g.:
  - Anatomy
  - Cellular biology
  - Aerospace
  - Cell Phones
  - …

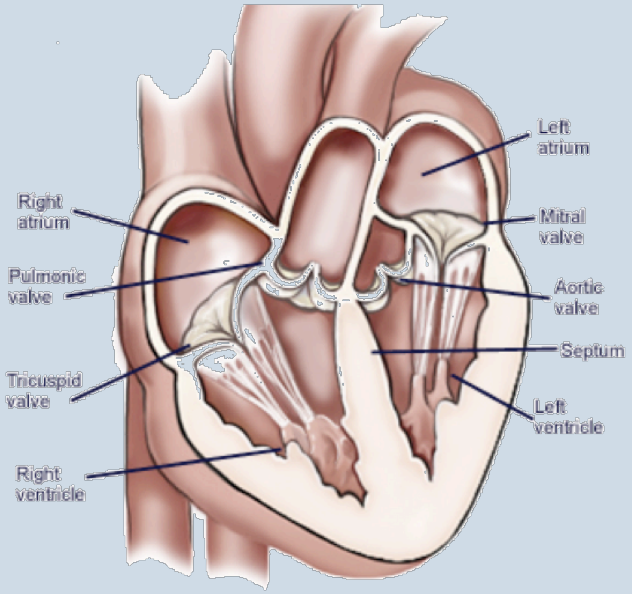


What is an Ontology?

A model of (some aspect of) the world

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

Heart is a muscular organ that is part of the circulatory system



What is an Ontology?

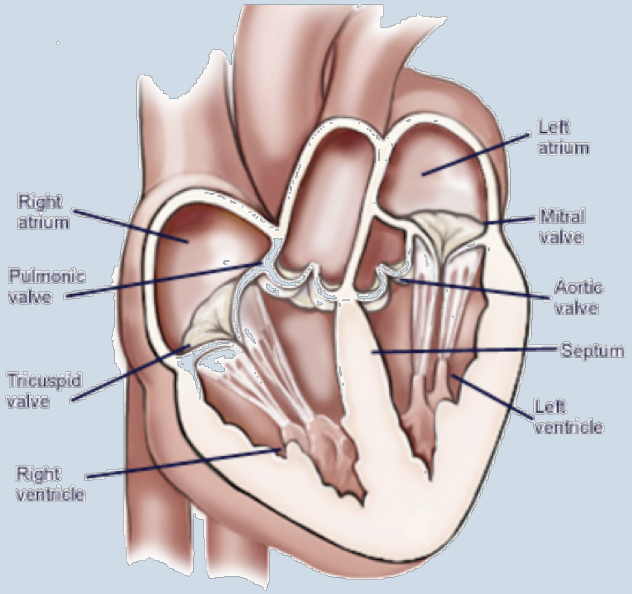
A model of (some aspect of) the world

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

  Heart is a muscular organ that is part of the circulatory system

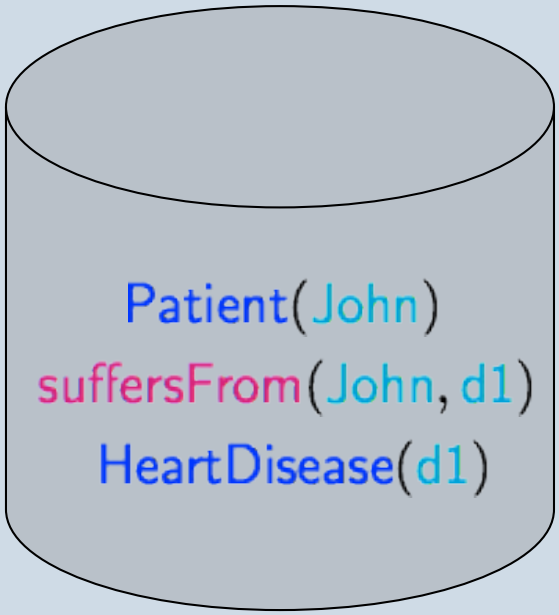
- **Formalised** using suitable logic

$$\forall x.[\mathsf{Heart}(x) \rightarrow \mathsf{MuscularOrgan}(x) \wedge \exists y.[\mathsf{isPartOf}(x,y) \wedge \mathsf{CirculatorySystem}(y)]]$$

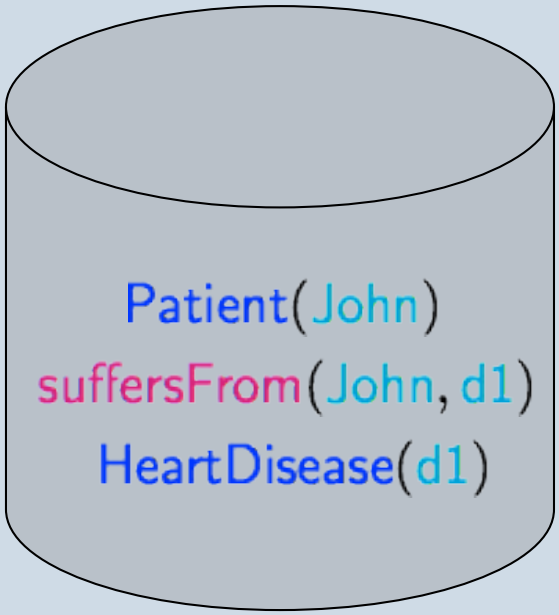


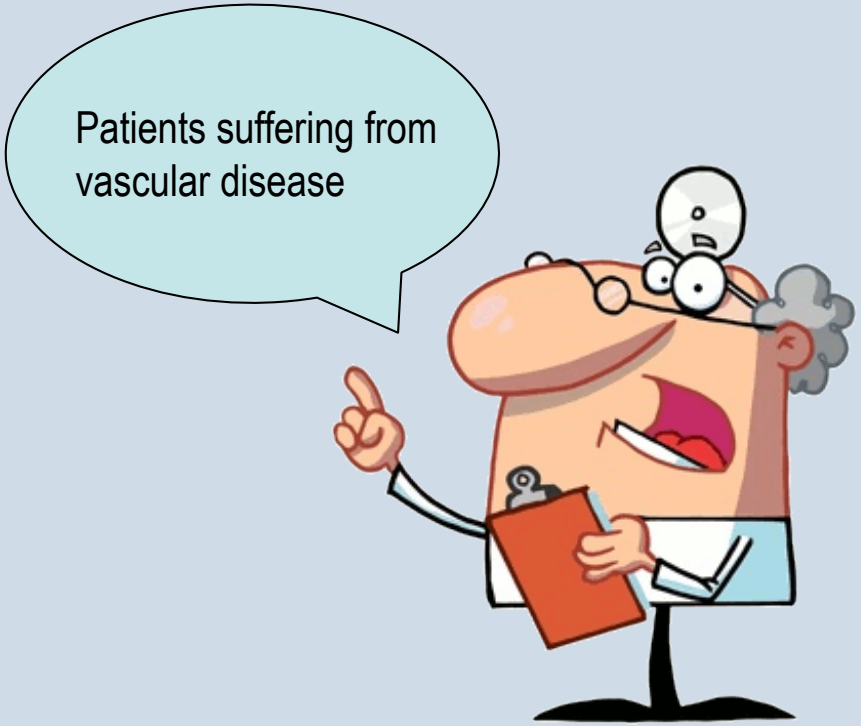
How are ontologies used?

How are ontologies used?

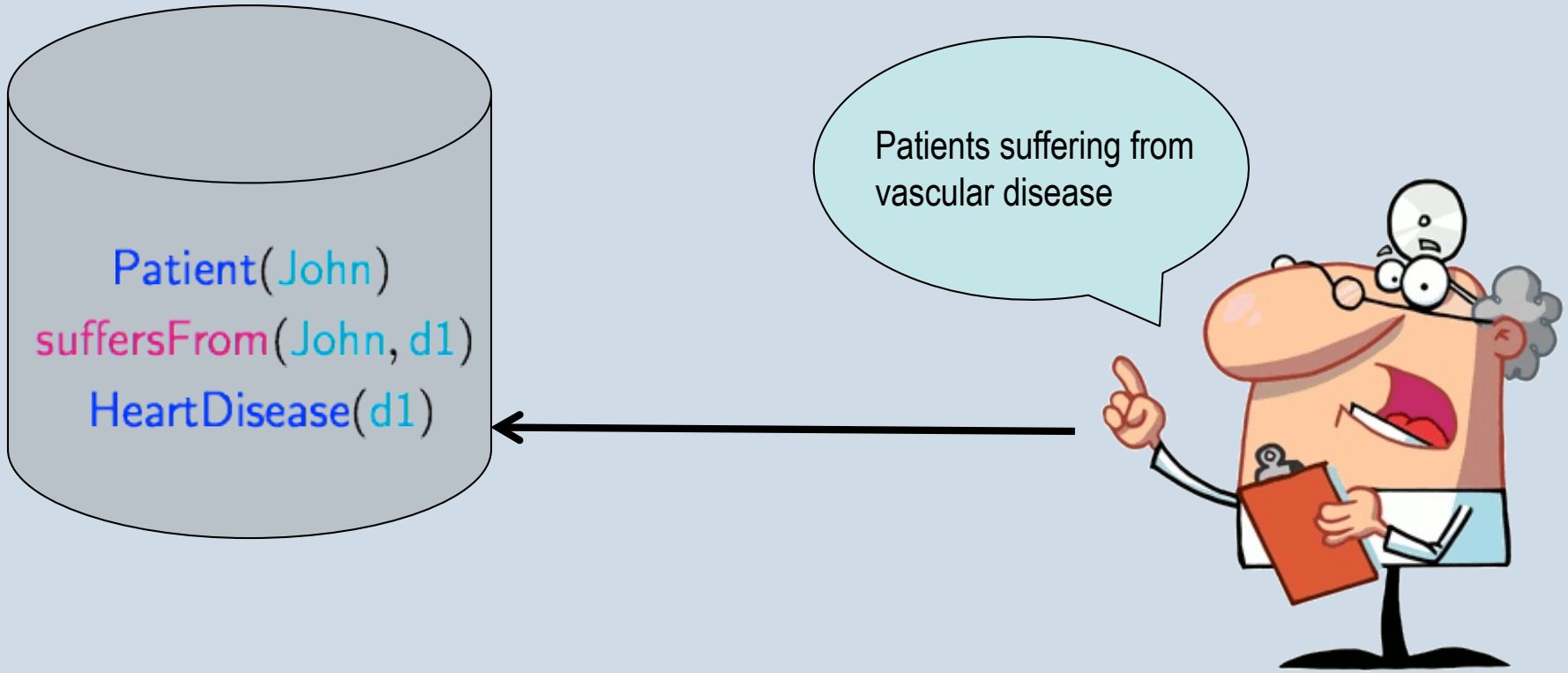


How are ontologies used?

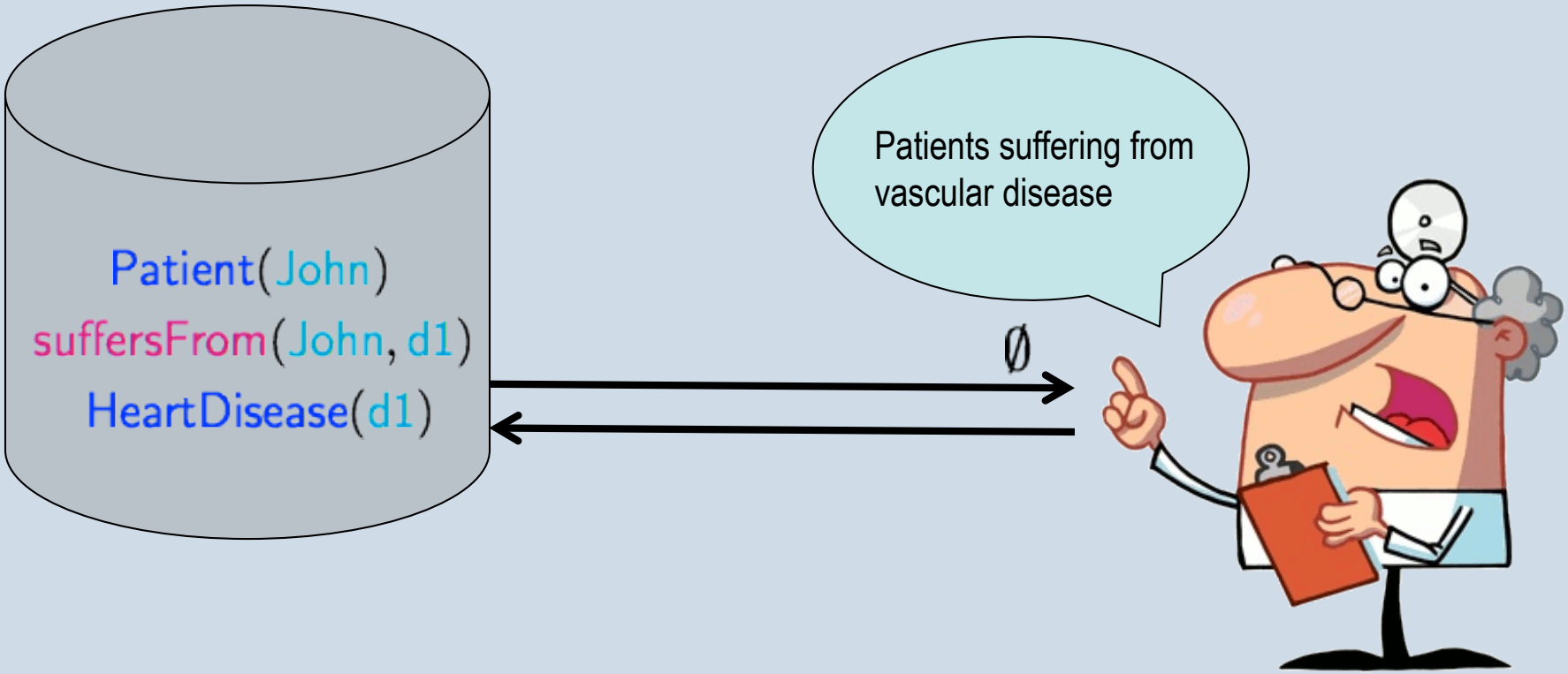




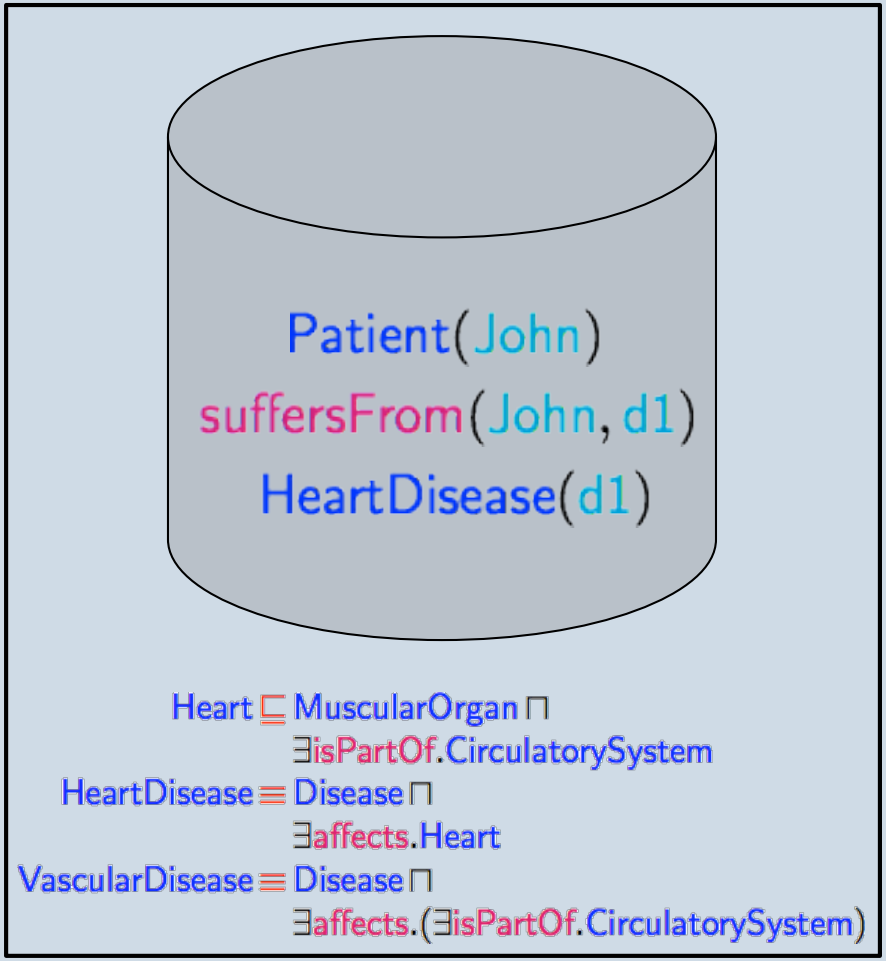
How are ontologies used?

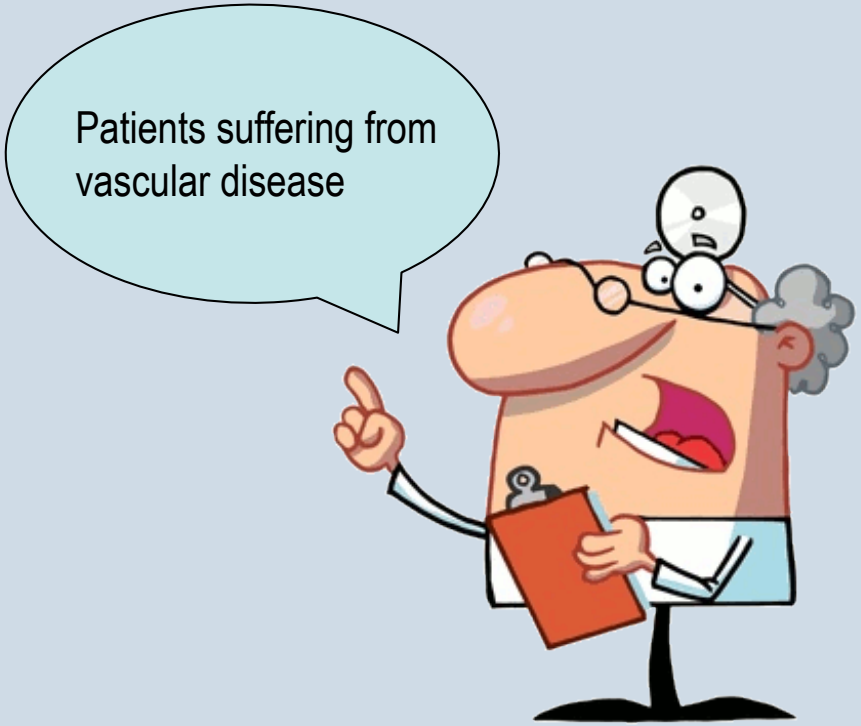


How are ontologies used?

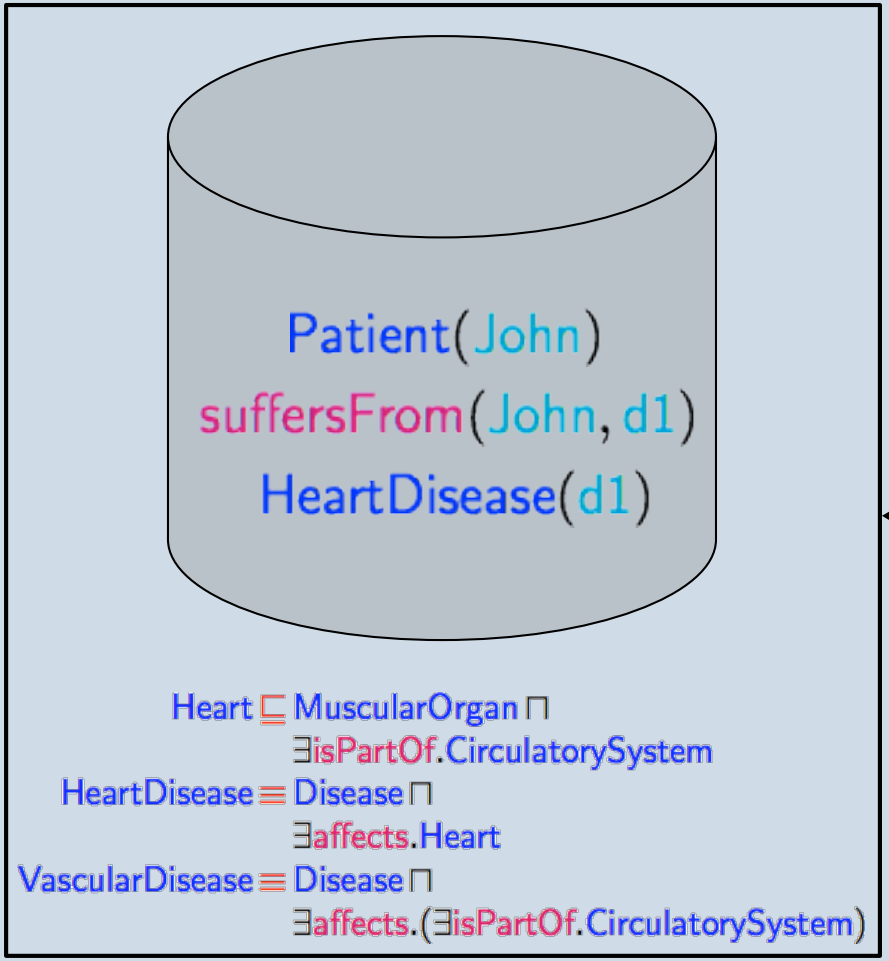


How are ontologies used?

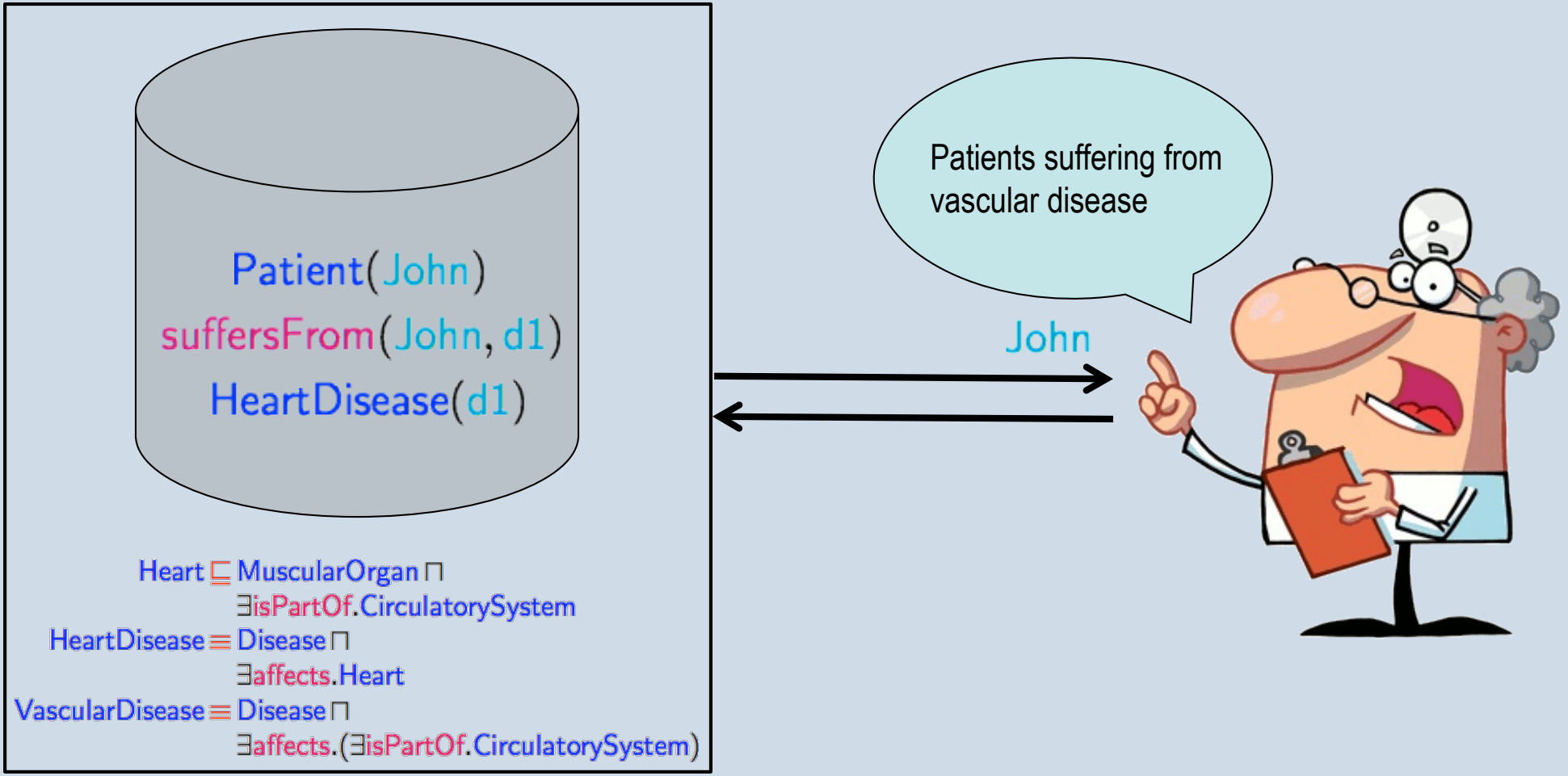




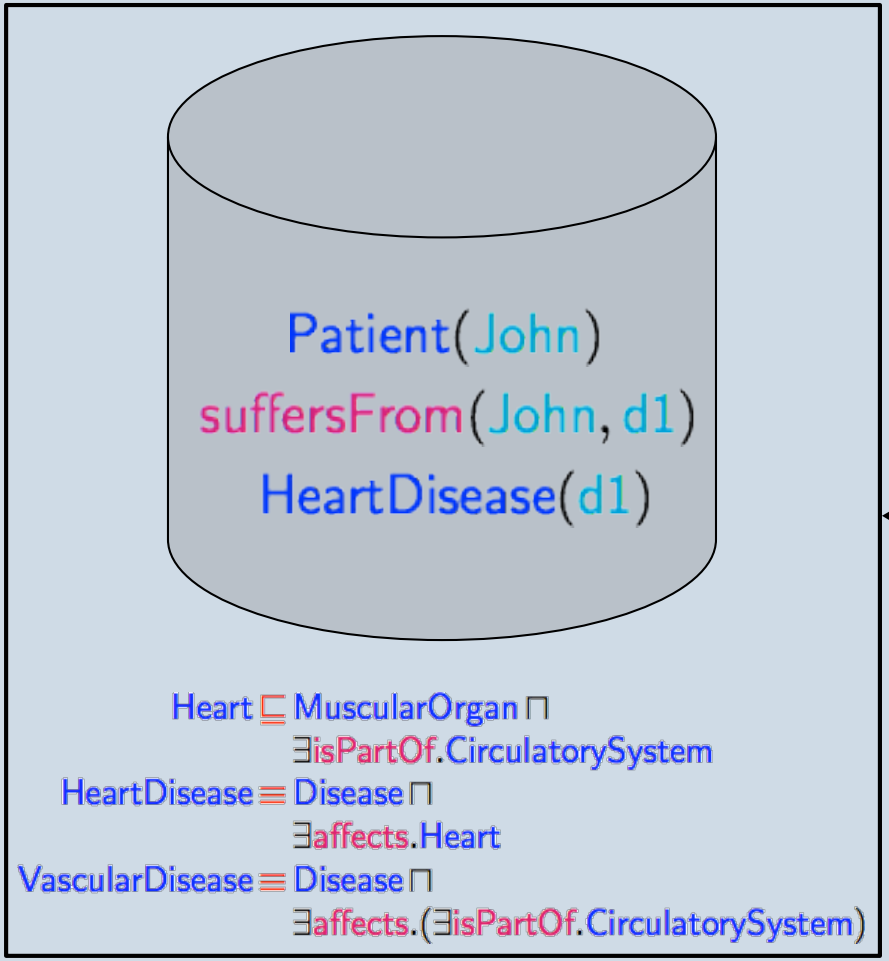
How are ontologies used?



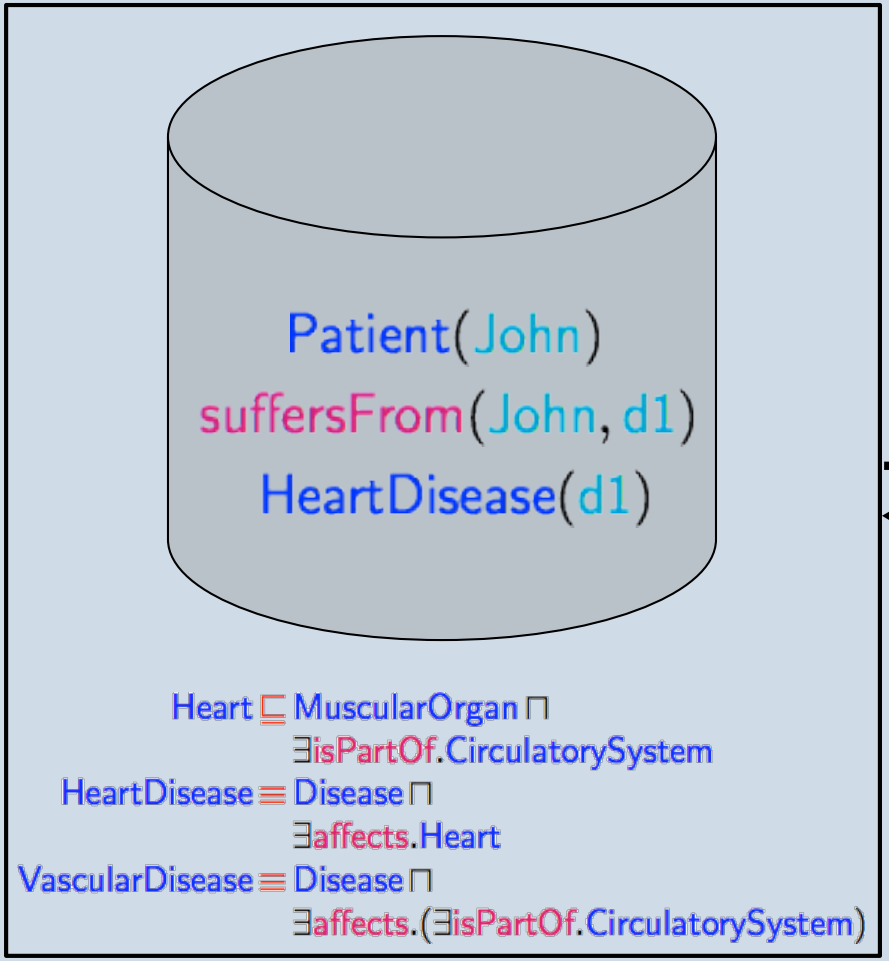
How are ontologies used?



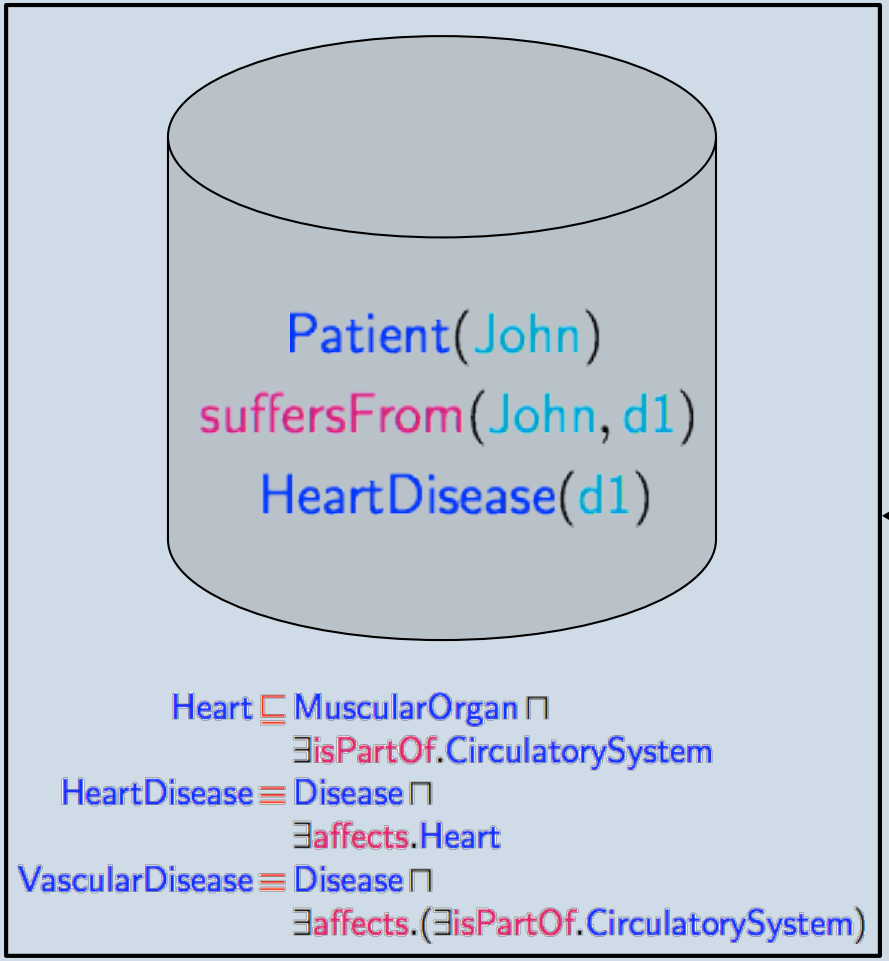
How are ontologies used?



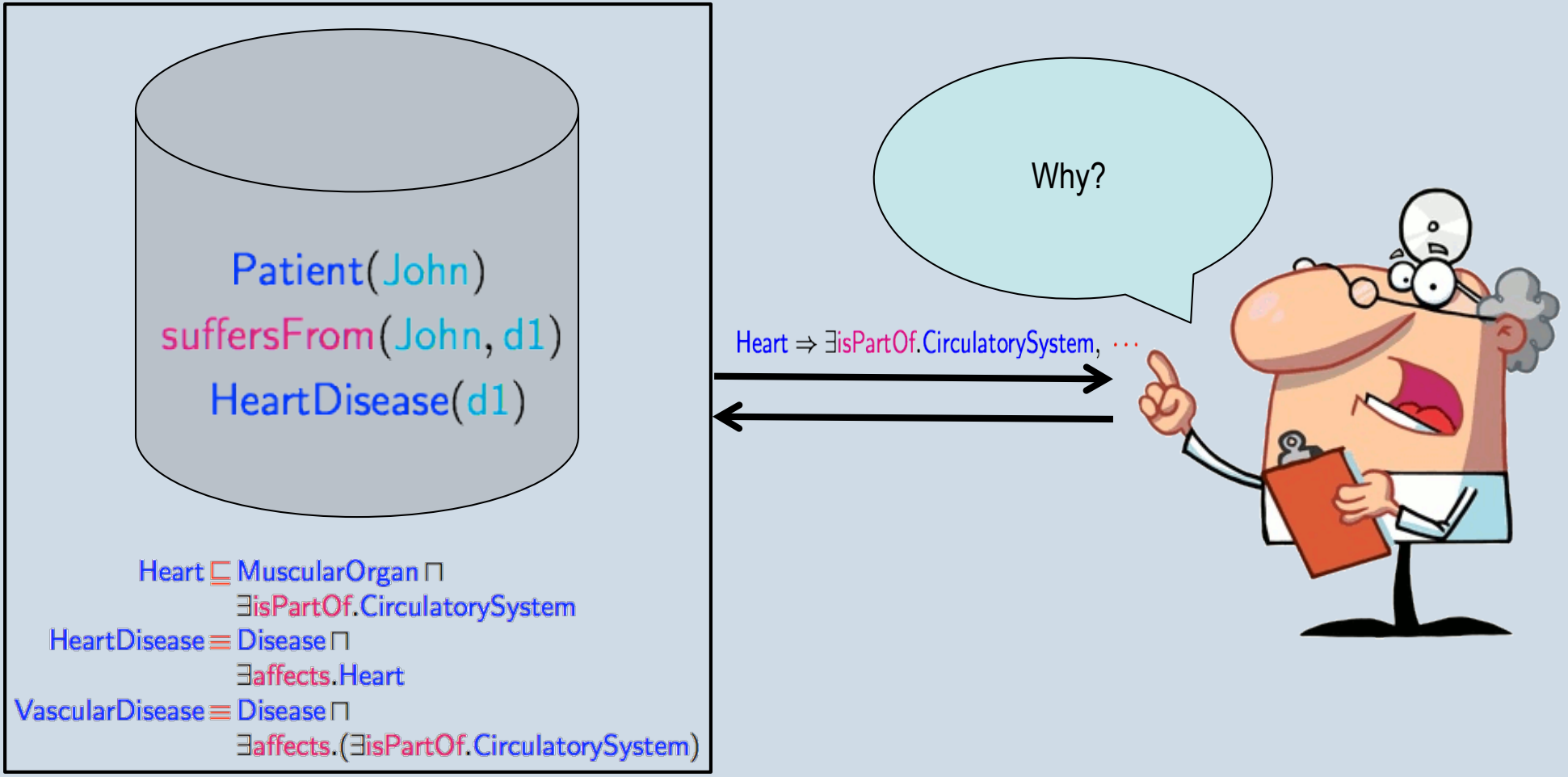
How are ontologies used?



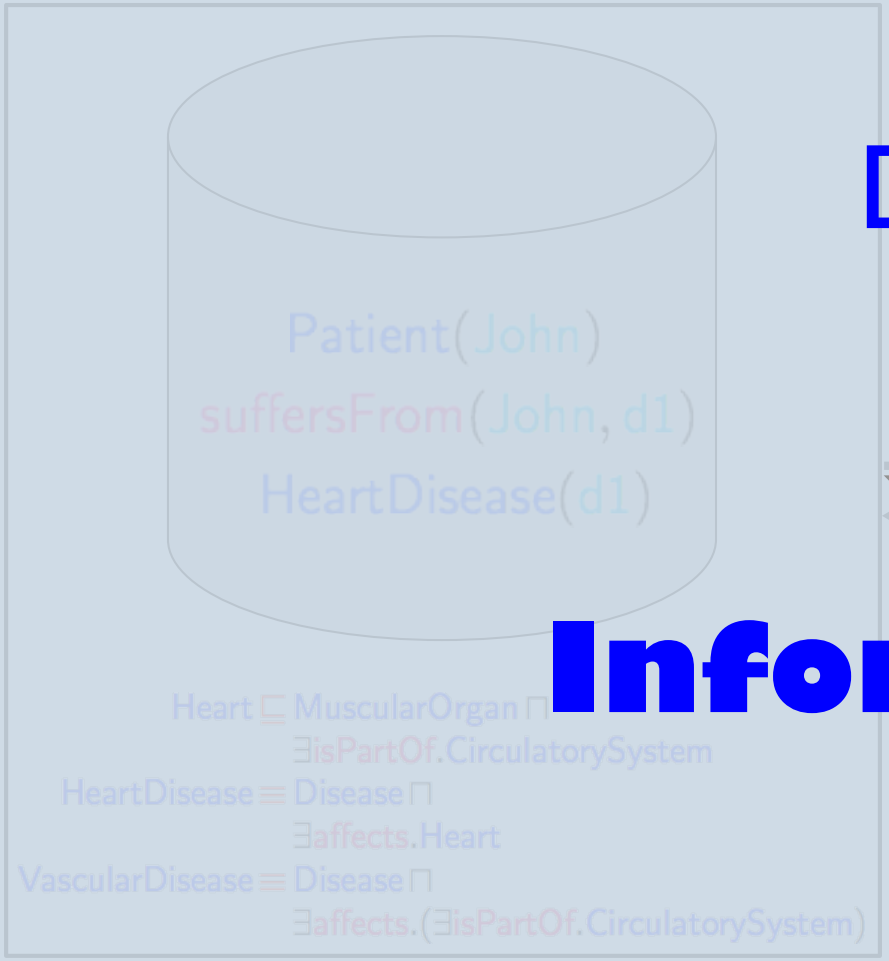
How are ontologies used?



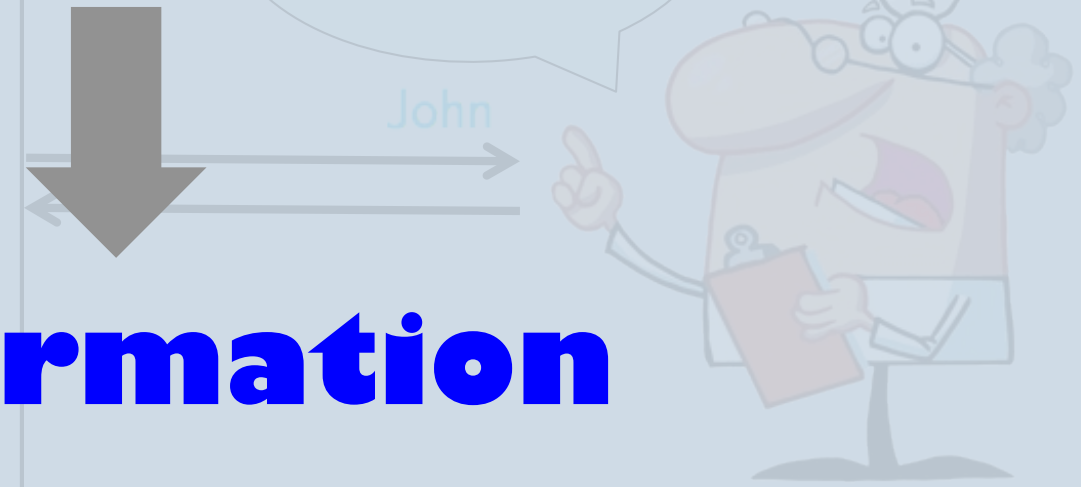
How are ontologies used?



How are ontologies used?



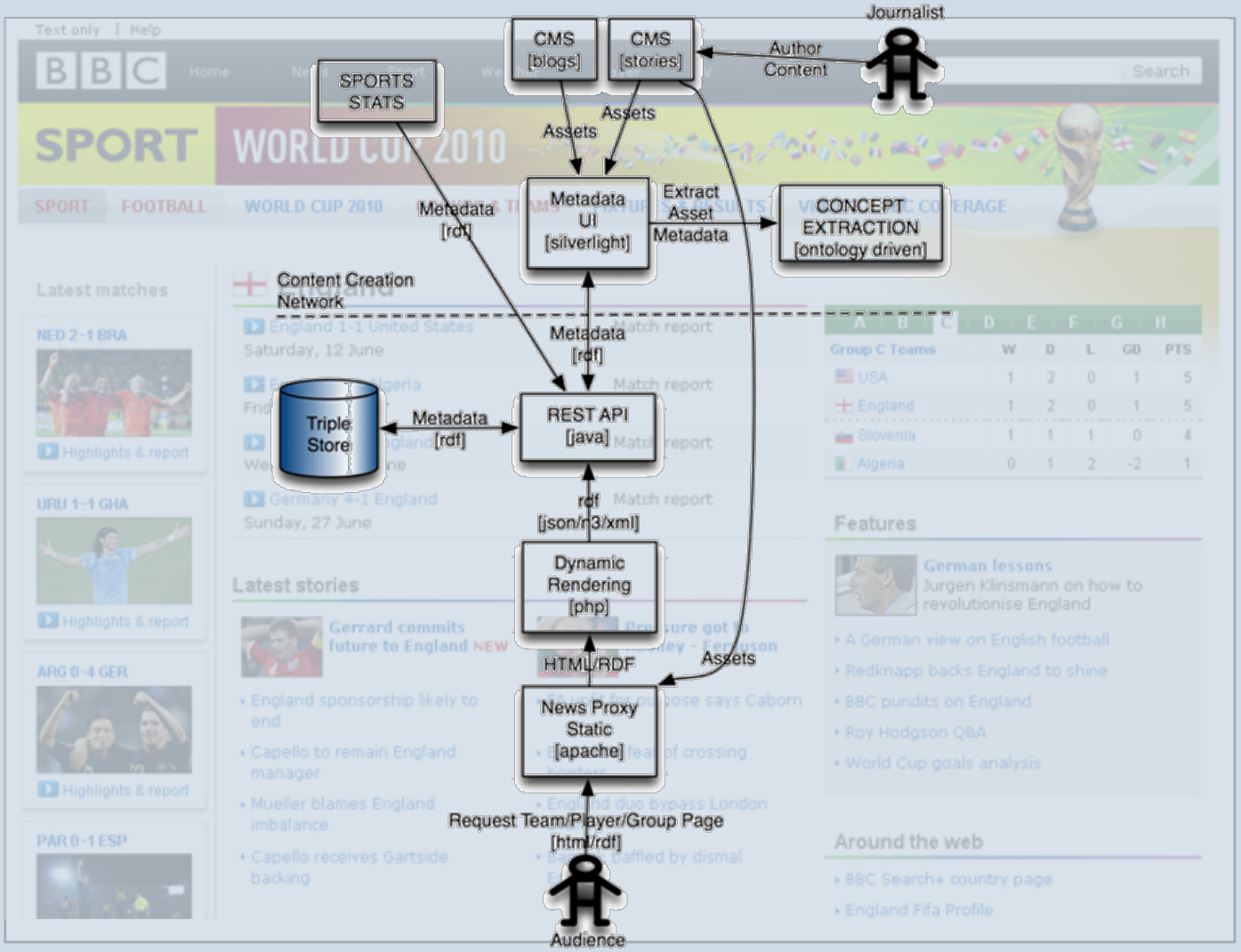
Data



Information

Applications: Semantic Web

Applications: Semantic Web



Applications: Semantic Web

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
- Using ontologies to **improve access** to relevant data
- Statoil estimate that this could add **€1B/year** to their net income

Applications: Energy Supply Industry

- **EDF Energy** offer personalised energy saving advice to every customer

- **OWL ontology** used to model relevant environmental factors

- **Oxford's HermiT reasoner** used to match customer circumstances with relevant pieces of advice

Applications: Intelligent Mobile Platform

- **Samsung** developing Intelligent Moblile Platform to support context-aware applications
- IMP monitors environment via **sensor data** (GPS, compass, accelerometer, ...)
- Reasoning enabled semantic data store used to **infer context**
- Applications exploit context to enable more **intelligent behaviour**

Research ⇝ Practice

- Selecting **suitable language**/logic
  - expressiveness -v- computability tradeoff

Research ⇝ Practice

- Selecting **suitable language**/logic
  - expressiveness -v- computability tradeoff
  - Description Logics are a family of FOL fragments with useful computational properties

Research ⇝ Practice

- Selecting **suitable language**/logic
  - expressiveness -v- computability tradeoff
  - Description Logics are a family of FOL fragments with useful computational properties

I1 $\dfrac{}{M \sqcap A \sqsubseteq A}$

I2 $\dfrac{}{M \sqsubseteq \top}$

R1 $\dfrac{M \sqsubseteq A_1 \quad \ldots \quad M \sqsubseteq A_n}{M \sqsubseteq C} : \prod_{i=1}^{n} A_i \sqsubseteq C \in \mathcal{O}$

R2 $\dfrac{M \sqsubseteq \exists R.N \quad N \sqsubseteq \bot}{M \sqsubseteq \bot}$

R3 $\dfrac{M \sqsubseteq \exists R_1.N \quad M \sqsubseteq \forall R_2.A}{M \sqsubseteq \exists R_1.(N \sqcap A)} : R_1 \sqsubseteq_{\mathcal{O}} R_2$

R4 $\dfrac{M \sqsubseteq \exists R_1.N \quad N \sqsubseteq \forall R_2.A}{M \sqsubseteq A} : R_1 \sqsubseteq_{\mathcal{O}} R_2^-$

R5 $\dfrac{\begin{array}{l} M \sqsubseteq \exists R_1.N_1 \quad N_1 \sqsubseteq B \\ M \sqsubseteq \exists R_2.N_2 \quad N_2 \sqsubseteq B \\ M \sqsubseteq \leqslant 1 S.B \end{array}}{M \sqsubseteq \exists R_1.(N_1 \sqcap N_2)} : \begin{array}{l} R_1 \sqsubseteq_{\mathcal{O}} S \\ R_2 \sqsubseteq_{\mathcal{O}} S \end{array}$

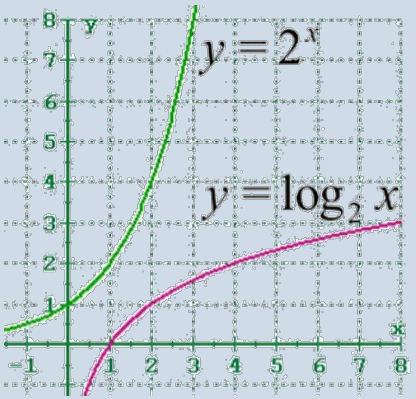
R6 $\dfrac{\begin{array}{l} M \sqsubseteq \exists R_1.N_1 \quad M \sqsubseteq B \\ N_1 \sqsubseteq \exists R_2.(N_2 \sqcap A) \\ N_1 \sqsubseteq \leqslant 1 S.B \quad N_2 \sqcap A \sqsubseteq B \end{array}}{M \sqsubseteq A \quad M \sqsubseteq \exists R_2^-.N_1} : \begin{array}{l} R_1 \sqsubseteq_{\mathcal{O}} S^- \\ R_2 \sqsubseteq_{\mathcal{O}} S \end{array}$

Research ⇝ Practice

- Selecting **suitable language**/logic
  - expressiveness -v- computability tradeoff
  - Description Logics are a family of FOL fragments with useful computational properties
  - OWL based on $\mathcal{SROIQ}$ DL

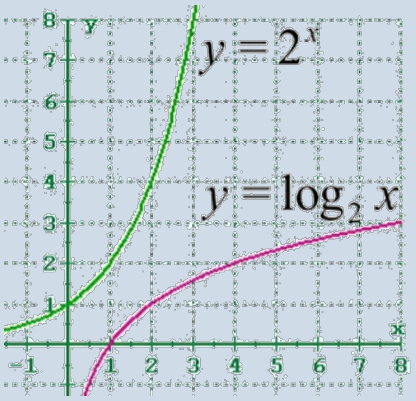
Research $\rightsquigarrow$ Practice

- Selecting **suitable language**/logic
  - expressiveness -v- computability tradeoff
  - Description Logics are a family of FOL fragments with useful computational properties
  - OWL based on $\mathcal{SROIQ}$ DL
- Dealing with **computational intractability**
  - $\mathcal{SROIQ}$ is 2NExpTime (-v- logspace for databases)



Research ⇝ Practice

- Selecting **suitable language**/logic
  - expressiveness -v- computability tradeoff
  - Description Logics are a family of FOL fragments with useful computational properties
  - OWL based on $\mathcal{SROIQ}$ DL
- Dealing with **computational intractability**
  - $\mathcal{SROIQ}$ is 2NExpTime (-v- logspace for databases)
  - Goal directed algorithms and highly optimised implementations effective for schema reasoning



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	—

Query Answering Still Problematical

Large datasets → conflicting requirements

- Modeling complex application domains requires:
  Rich ontology languages
- Fine-grained information access requires:
  Powerful query languages
- Processing large datasets requires:
  Scalable reasoning
- Applications (often) require:
  Reliable answers (guaranteed sound and complete)

Various Approaches — Different Tradeoffs

❶ Use full power of OWL and complete reasoner

input: any OWL ontology, dataset and query

output: sound and complete query answer

❷ Use a suitable "profile" and specialised reasoner:

input: restricted OWL ontology; any dataset and query

output: sound and complete query answer

❸ Use full power of OWL and incomplete reasoner

input: any OWL ontology, dataset and query

output: sound but (possibly) incomplete query answer

Various Approaches — Different Tradeoffs

❶ Use full power of OWL and complete reasoner:

✓ Well-suited for modeling complex domains

✓ Reliable answers

✗ High worst-case complexity

✗ Scalability problems for large datasets

Complete ontology reasoners:

- E.g., FaCT++, **HermiT**, Pellet, ...
- Based on (hyper)tableau (model construction) theorem provers
- Proof needed for each answer tuple
- Unlikely to scale to very large datasets

Various Approaches — Different Tradeoffs

❷ Use a suitable "profile" and specialised reasoner:

✓ Tractable 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
- Data in scalable (relational) data stores

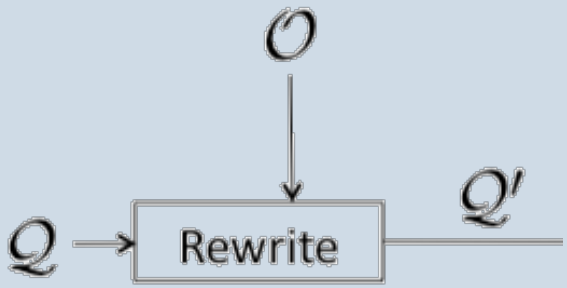
Query Rewriting

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

Query Rewriting

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

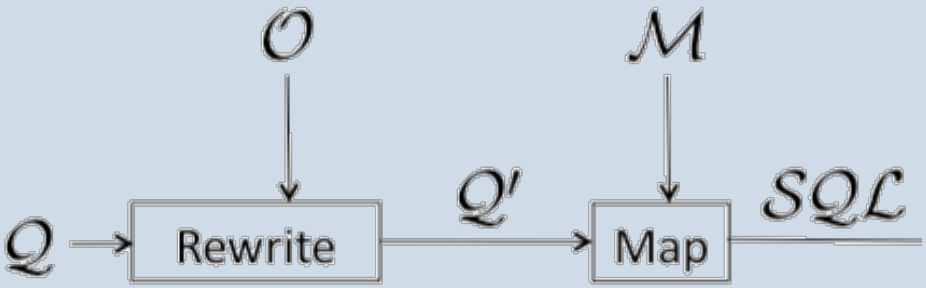
- **Rewrite** $\mathcal{Q} \rightarrow \mathcal{Q}'$ s.t. answering $\mathcal{Q}'$ without $\mathcal{O}$ equivalent to answering $\mathcal{Q}$ w.r.t. $\mathcal{O}$ *for any dataset*



Query Rewriting

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

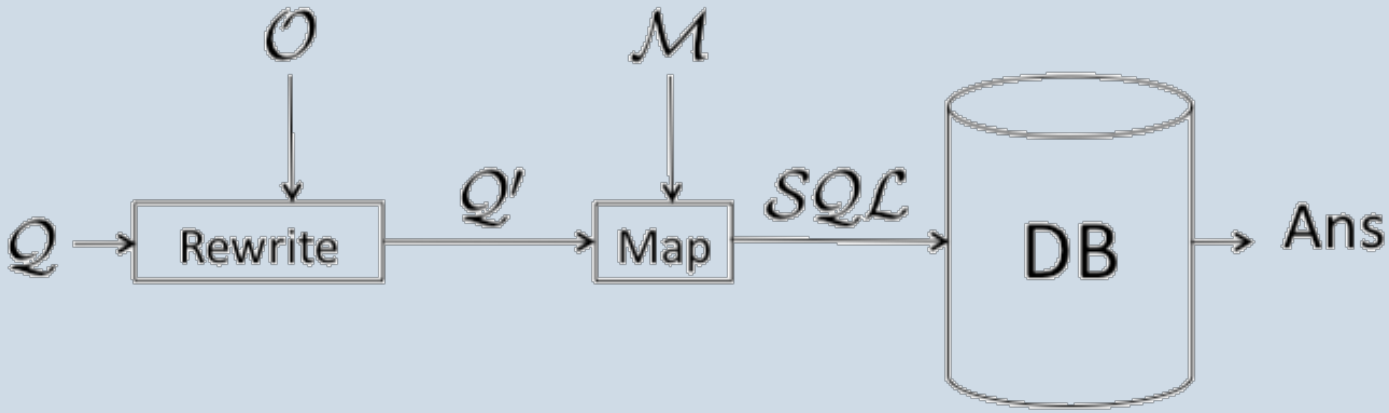
- **Rewrite** $\mathcal{Q} \rightarrow \mathcal{Q}'$ s.t. answering $\mathcal{Q}'$ without $\mathcal{O}$ equivalent to answering $\mathcal{Q}$ w.r.t. $\mathcal{O}$ *for any dataset*
- **Map** ontology queries → DB queries (typically SQL) using mappings $\mathcal{M}$ to rewrite $\mathcal{Q}$' into a DB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



Query Rewriting

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

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

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

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

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

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

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)
- ✗ Materialisation based techniques assume static data

Incomplete ontology reasoners:

- E.g., Oracle's Semantic Datastore, Sesame, Jena, OWLim, ...
- Based on RDF triple stores and deductive DB technologies
- Widely used in practice to reason with large datasets

Materialisation

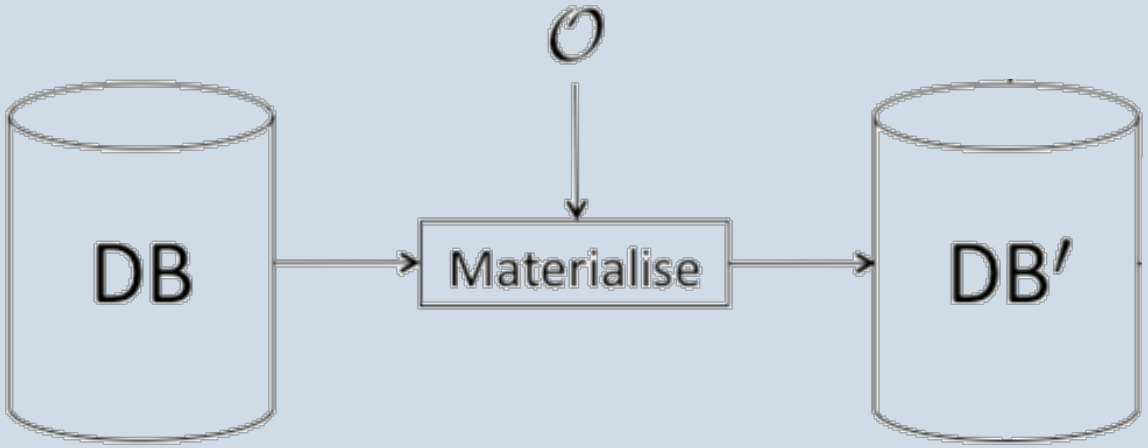
Given (RDF) data DB, ontology $\mathcal{O}$ and query $\mathcal{Q}$:

Materialisation

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



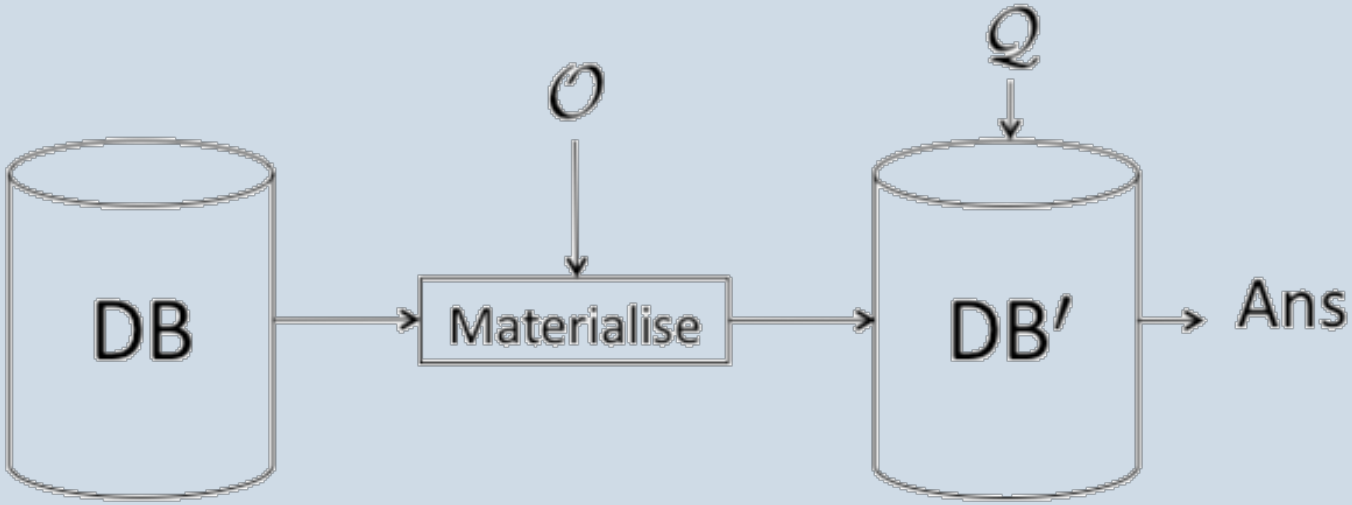
Materialisation

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

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

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

Materialisation — Example

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

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

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

Materialisation — Example

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

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

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

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

Materialisation — Example

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

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

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

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

Materialisation — Example

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

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

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

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

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

Materialisation — Example

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

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

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

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

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

Materialisation — Issues

❶ **Frequently changing data**

- Materialisation is time consuming
- Need to re-materialise whenever ontology or data changes
- Data may change very frequently

❷ **Incompleteness**

- Difficult to predict and/or quantify
- May be unacceptable in some applications

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

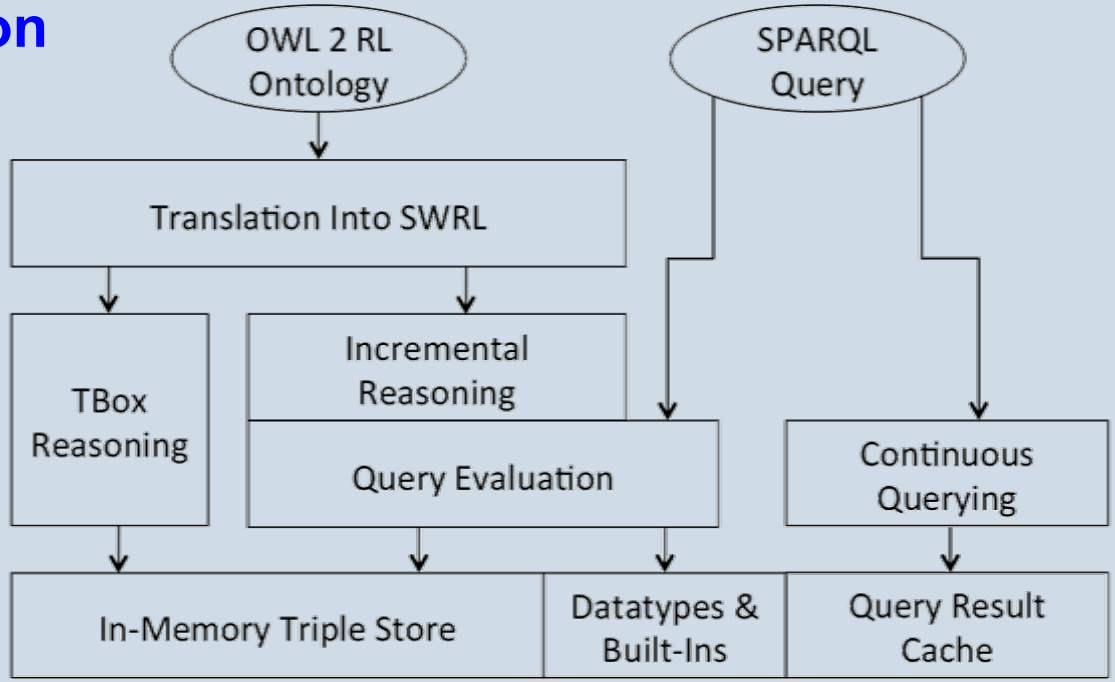
Changing/retracting data is much harder

- Naive solution requires all materialised facts to be discarded
- Re-materialisation very costly for large data sets

Samsung Delta-Reasoner

Incremental materialisation

- Ontology translated into rules
- Separates TBox and ABox reasoning
- Initial materialisation of ABox
- View Maintenance for incremental reasoning



Supports

- OWL 2 RL ontology axioms and SWRL rules
- SPARQL queries, including registration of continuous queries

Samsung Delta-Reasoner

Test Ontologies:

	TBox axioms	Class assertions	Property assertions	DL expressivity
VICODI	223	33,238	82,943	$\mathcal{ALHI}(\mathbf{D})$
SEMINTEC	219	17,941	47,299	$\mathcal{ALHIF}$
LUBM	93	18,128	82,415	$\mathcal{ALEHI}^{+}(\mathbf{D})$

Loading and (incremental) materialisation times (ms):

	Load	Materialise	Update
VICODI	2127	2820	156
SEMINTEC	1048	1123	157
LUBM	2597	818	135

After changing 1,000 triples at random

Dealing with Incompleteness

- Materialisation based reasoning complete for **OWL 2 RL** profile
- But for ontologies **outside the profile**:
  - Reasoning may be incomplete
  - Incompleteness difficult to measure via empirical testing
- Possible solutions offered by recent work on:
  - **Measuring and repairing incompleteness**
  - **Chase materialisation**

Measuring and Repairing Incompleteness

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

Measuring and Repairing Incompleteness

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

Measuring and Repairing Incompleteness

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

Measuring and Repairing Incompleteness

Example — consider the following reasoners:

rdf: ignores $\mathcal{T}$ and evaluates Q w.r.t. $\mathcal{A}$ only

Measuring and Repairing Incompleteness

Example — consider the following reasoners:

rdf: ignores $\mathcal{T}$ and evaluates $\mathcal{Q}$ w.r.t. $\mathcal{A}$ only

rdfs: translates the RFDS subset of $\mathcal{T}$ into a datalog program and then evaluates such program w.r.t. $\mathcal{Q}$ and. $\mathcal{A}$

Measuring and Repairing Incompleteness

Example — consider the following reasoners:

rdf: ignores $\mathcal{T}$ and evaluates $\mathcal{Q}$ w.r.t. $\mathcal{A}$ only

rdfs: translates the RFDS subset of $\mathcal{T}$ into a datalog program and then evaluates such program w.r.t. $\mathcal{Q}$ and. $\mathcal{A}$

rl: translates the OWL 2 RL subset of $\mathcal{T}$ into a datalog program and then evaluates such program w.r.t. $\mathcal{Q}$ and. $\mathcal{A}$

Measuring and Repairing Incompleteness

Example — consider the following reasoners:

rdf: ignores $\mathcal{T}$ and evaluates $\mathcal{Q}$ w.r.t. $\mathcal{A}$ only

rdfs: translates the RFDS subset of $\mathcal{T}$ into a datalog program and then evaluates such program w.r.t. $\mathcal{Q}$ and. $\mathcal{A}$

rl: translates the OWL 2 RL subset of $\mathcal{T}$ into a datalog program and then evaluates such program w.r.t. $\mathcal{Q}$ and. $\mathcal{A}$

classify: classifies $\mathcal{T}$ with complete reasoner; extends $\mathcal{T}$ with new subsumptions; calls rl with input $\mathcal{Q}$, $\mathcal{A}$ and extended TBox

Measuring and Repairing Incompleteness

Example — consider the following $\mathcal{T}$, $\mathcal{Q}$ and test suite:

$$\mathcal{T} = \left\{ \begin{array}{r} \exists \text{takes.MathCo} \sqsubseteq \text{St} \\ \text{CalcCo} \sqsubseteq \text{MathCo} \\ \text{MathSt} \sqsubseteq \exists \text{takes.MathCo} \\ \text{St} \sqcap \text{Prof} \sqsubseteq \bot \end{array} \right\}$$

$$\mathcal{Q} = \{\text{St}(x) \wedge \text{takesCo}(x, y) \wedge \text{MathCo}(y) \rightarrow Q(x)\}$$

$$\mathcal{A}_1 = \{\text{takesCo}(c, d), \text{MathCo}(d)\}$$
$$\mathcal{A}_3 = \{\text{takesCo}(c, d), \text{CalcCo}(d)\}$$
$$\mathcal{A}_5 = \{\text{MathSt}(c)\}$$

$$\mathcal{A}_2 = \{\text{takesCo}(c, c), \text{MathCo}(c)\}$$
$$\mathcal{A}_4 = \{\text{takesCo}(c, c), \text{CalcCo}(c)\}$$
$$\mathcal{A}_6 = \{\text{St}(c), \text{Prof}(c)\}$$

rdf: Fails all tests (✗)

rdfs: Fails all tests (✗)

rl: Fails for $\mathcal{A}_5$ (✗)

classify: Passes all tests (✓)

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

Chase Materialisation — Example

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

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

Chase Materialisation — Example

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

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

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

Skolems

Chase Materialisation — Example

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

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

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

Skolems

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

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

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

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

Discussion

Numerous **exciting developments** & research areas

- Rewriting: optimisations, extensions (datalog engines), etc.
- Materialisation: chase, repair, truth maintenance etc.
- Hybrid techniques
- Column stores, massive parallelism, etc

Discussion

Numerous **exciting developments** & research areas

- Rewriting: optimisations, extensions (datalog engines), etc.
- Materialisation: chase, repair, truth maintenance etc.
- Hybrid techniques
- Column stores, massive parallelism, 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 etc.
- Hybrid techniques
- Column stores, massive parallelism, etc

Consider **progress on schema reasoning**:

Year	$\mathcal{O}$-size	Complete	Time (s)
1998	3,000	Yes	300
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 etc.
- Hybrid techniques
- Column stores, massive parallelism etc.

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

Consider **progress on schema reasoning**:

Year	$\mathcal{O}$ size	Complete	Time (s)
1998	3,000	Yes	10
2005	30,000	Yes	50
2010	400,000	Yes	5

Looking forward to similar progress on query answering!

Acknowledgements

EPSRC Engineering and Physical Sciences Research Council

SIXTH FRAMEWORK PROGRAMME

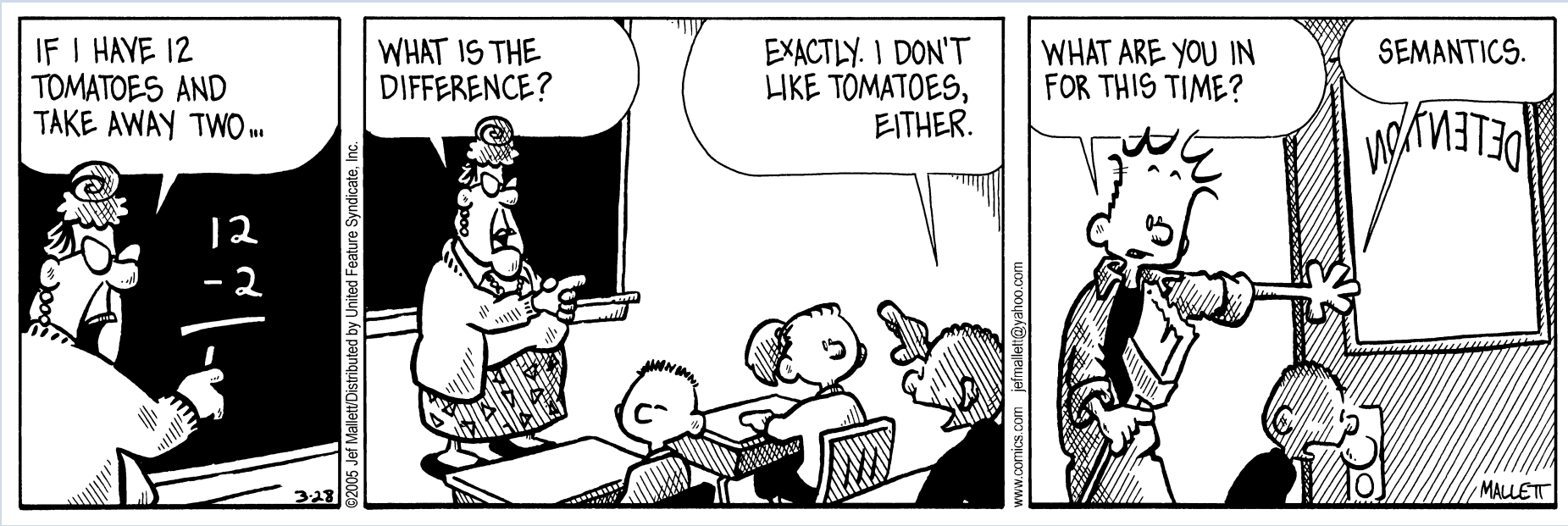
SEVENTH FRAMEWORK PROGRAMME

DARPA

UNIVERSITY OF OXFORD DEPARTMENT OF COMPUTER SCIENCE

Information Systems Group

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