UNIVERSITY OF
Southampton

UNIVERSITY OF Southampton

A Brief Introduction to Linked Data and the Semantic Web

Dr Nick Gibbins – nmg@soton.ac.uk

https://edshare.soton.ac.uk/22005/

UNIVERSITY OF Southampton

The World Wide Web: Past, Present and Future

a goal of the Web was that, if the interaction between person and hypertext could be so intuitive that the machine-readable information space gave an accurate representation of the state of people's thoughts, interactions, and work patterns, then machine analysis could become a very powerful management tool, seeing patterns in our work and facilitating our working together

Berners-Lee, T. (1996) *The World Wide Web: Past, Present and Future*. https://www.w3.org/People/Berners-Lee/1996/ppf.html

UNIVERSITY OF Southampton

Weaving the Semantic Web

I have a dream for the Web [in which computers] become capable of analyzing all the data on the Web – the content, links, and transactions between people and computers. A 'Semantic Web', which should make this possible, has yet to emerge, but when it does, the day-to-day mechanisms of trade, bureaucracy and our daily lives will be handled by machines talking to machines.

T. Berners-Lee (1999) *Weaving the Web*. San Francisco, CA: Harper

What is the Semantic Web?

"The goal of the Semantic Web initiative is as broad as that of the Web: to create a universal medium for the exchange of data. It is envisaged to smoothly interconnect personal information management, enterprise application integration, and the global sharing of commercial, scientific and cultural data. Facilities to put machine-understandable data on the Web are quickly becoming a high priority for many organizations, individuals and communities.

The Web can reach its full potential only if it becomes a place where data can be shared and processed by automated tools as well as by people."

W3C (2013) *Semantic Web Activity Statement*. https://www.w3.org/2001/sw/Activity

UNIVERSITY OF Southampton

Rocket Science (not)

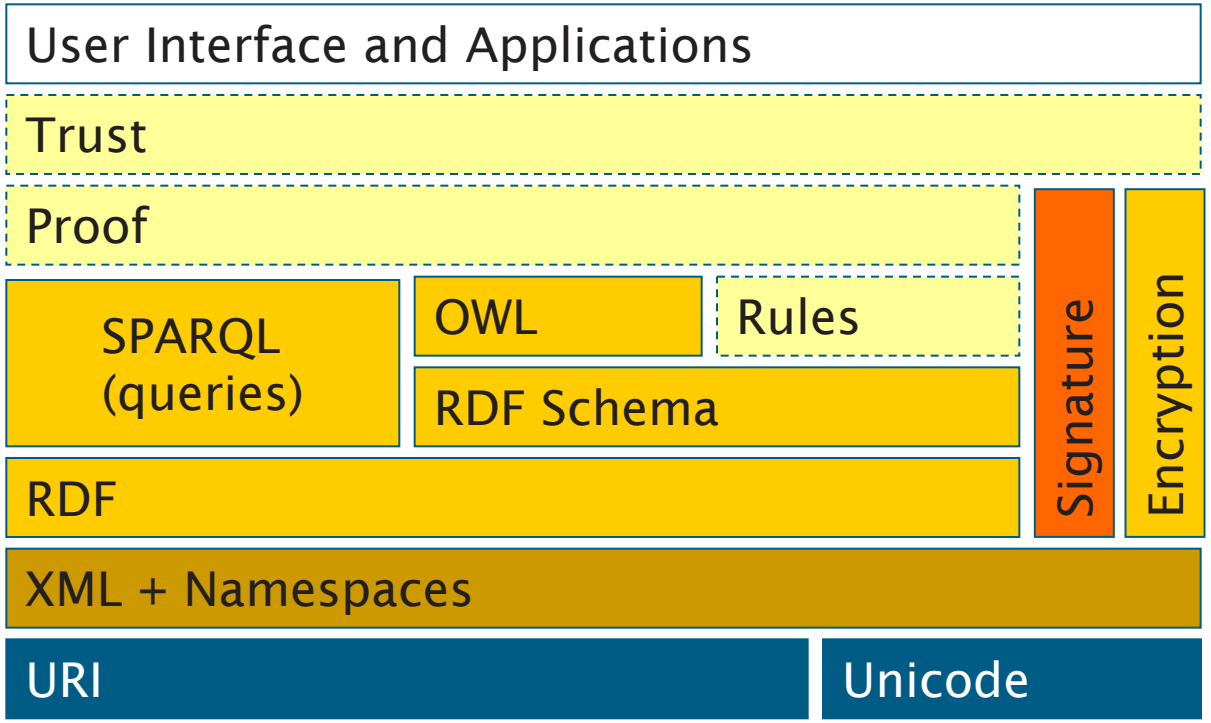
Is this rocket science? Well, not really. The Semantic Web, like the World Wide Web, is just taking well established ideas, and making them work interoperably over the Internet. This is done with standards, which is what the World Wide Web Consortium is all about. We are not inventing relational models for data, or query systems or rule-based systems. We are just webizing them.

Berners-Lee, T. (2001) *Business Model for the Semantic Web*, http://www.w3.org/DesignIssues/Business

UNIVERSITY OF Southampton

Semantic Web Technical Architecture

The Semantic Web layer cake



Attribution

Explanation

Ontologies + Inference

Metadata

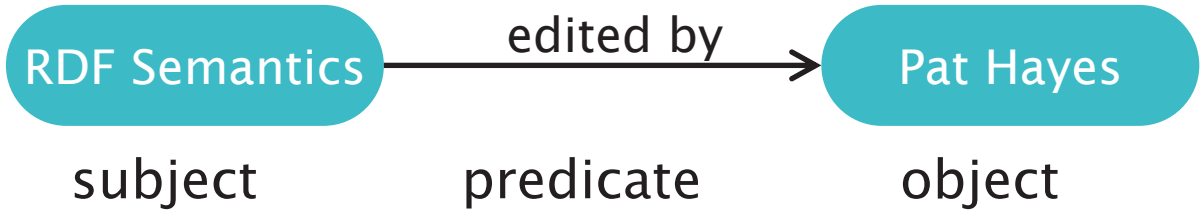
Standard syntax

Identity

Triples

Underlying data model of **triples**

Typed relationship between objects



Example

Take a citation:

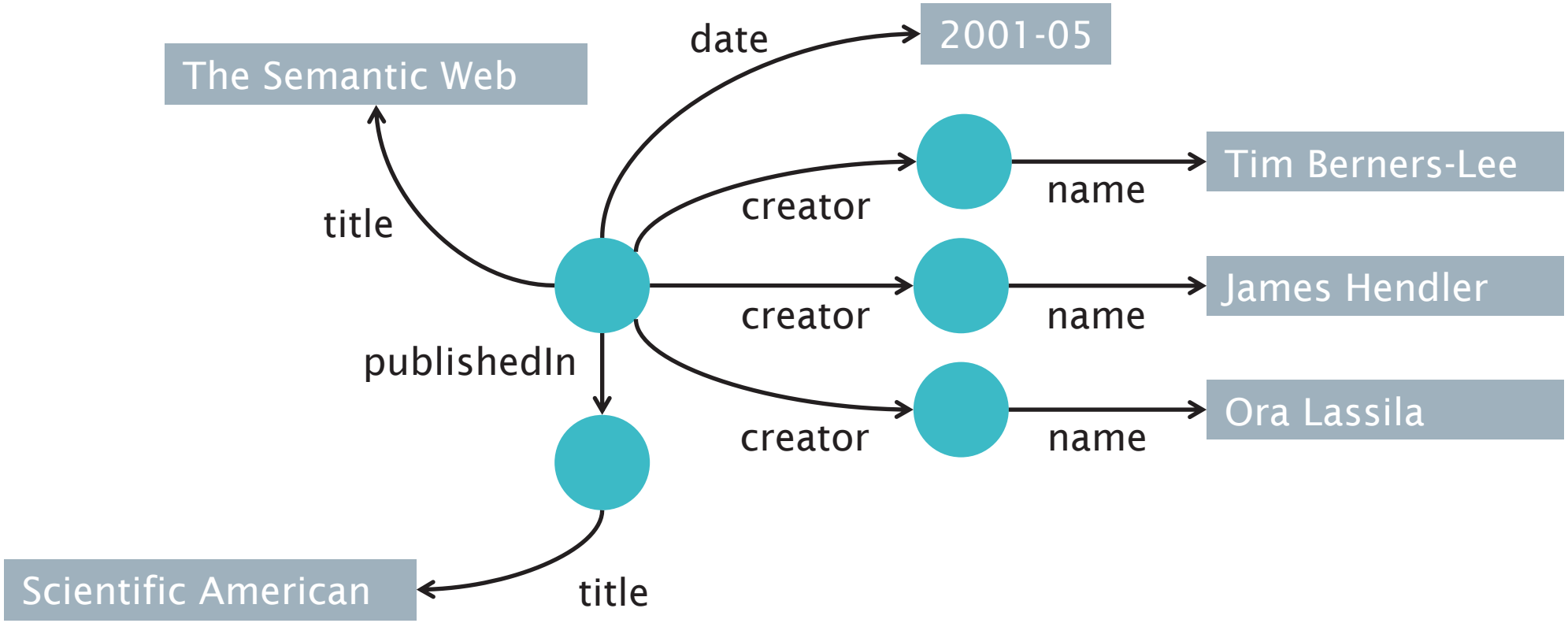
- Tim Berners-Lee, James Hendler and Ora Lassila. The Semantic Web. Scientific American, May 2001

We can identify a number of distinct statements in this citation:

- There is an article titled “The Semantic Web”
- One of its authors is a person named “Tim Berners-Lee” (etc)
- It appeared in a publication titled “Scientific American”
- It was published in May 2001

Example

We can represent these statements graphically:



Example

There are two types of vertex in this graph:

- **Literals**, which have a value but no identity
  (a string, a number, a date)

Scientific American

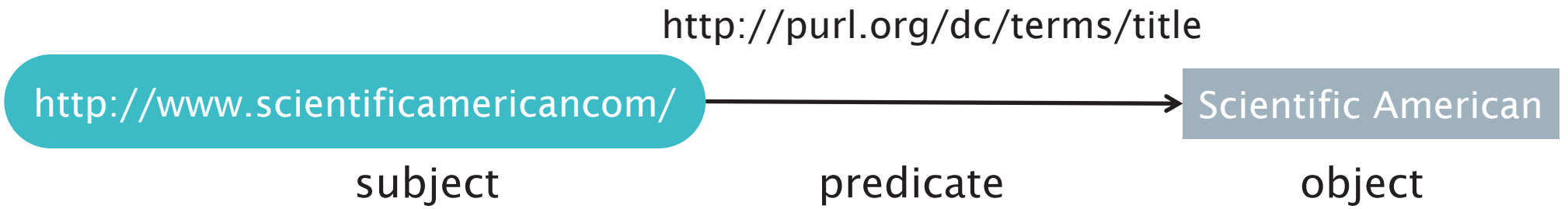
- **Resources**, which represent objects with identity
  (a web page, a person, a journal)

Example

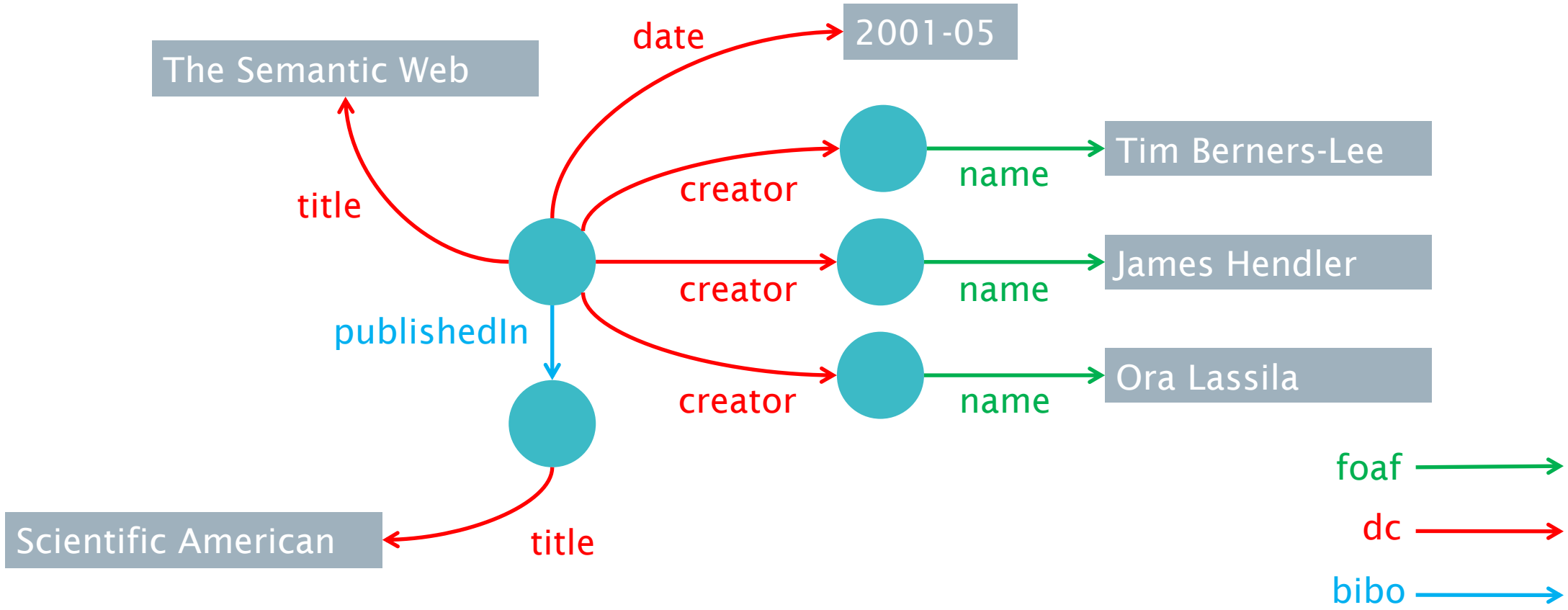
Resources are identified by URIs

Properties are resources that are used as predicates

A collection of properties constitutes a **vocabulary**



Mixing vocabularies



UNIVERSITY OF Southampton

Linked Data

Architecture of the World Wide Web

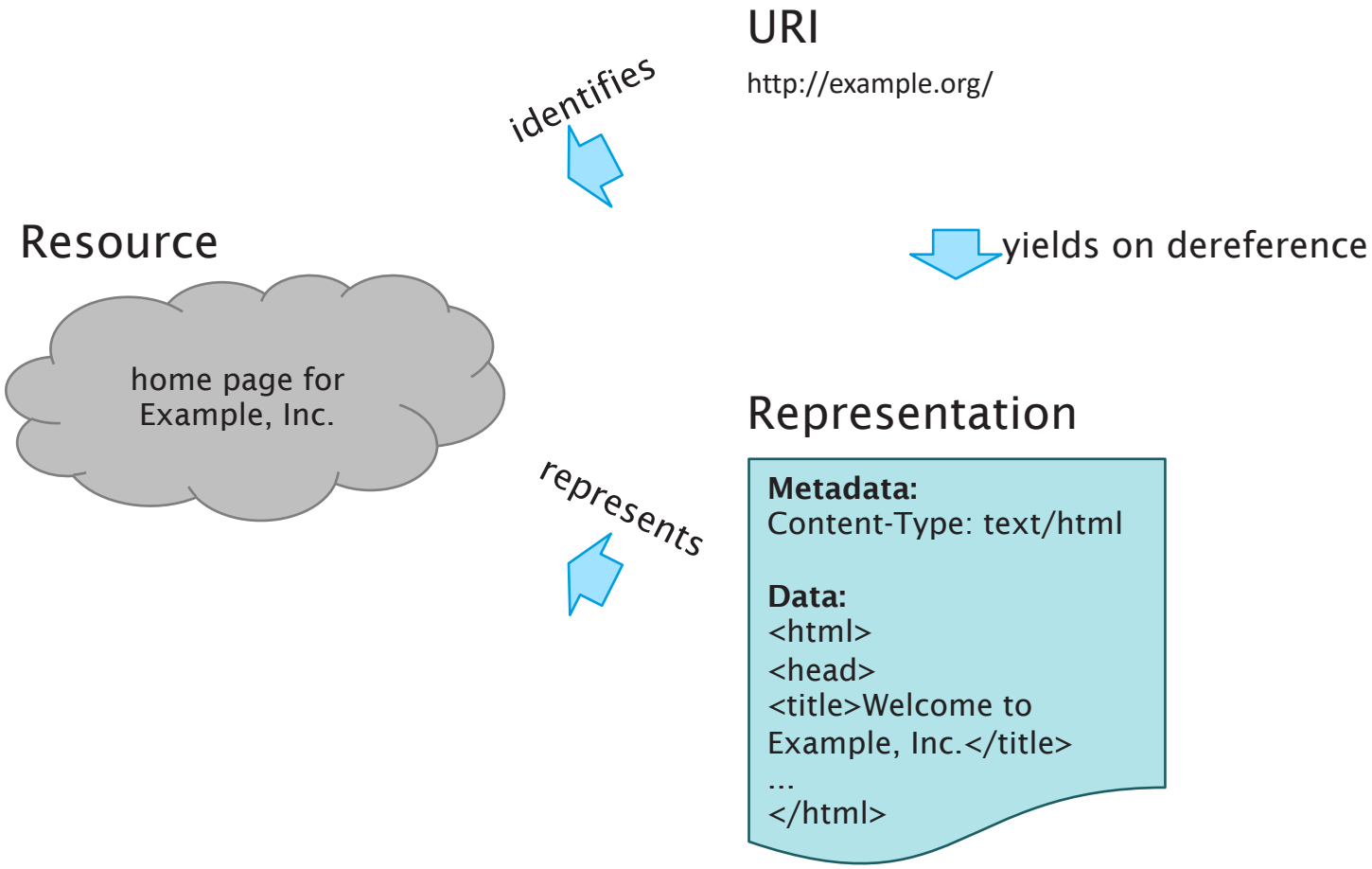
The Web architecture has four main components:

- Resources (webpages, etc)
- Identifiers for resources (URIs)
- Protocols for interacting with resources (HTTP)
- Data formats for representing the state of resources (HTML, XML, etc)

The Semantic Web builds on this foundation

- Rely on hypertextual nature of the Web to create links between data

Architecture of the World Wide Web



http://www.w3.org/TR/webarch/

Linked Data Principles

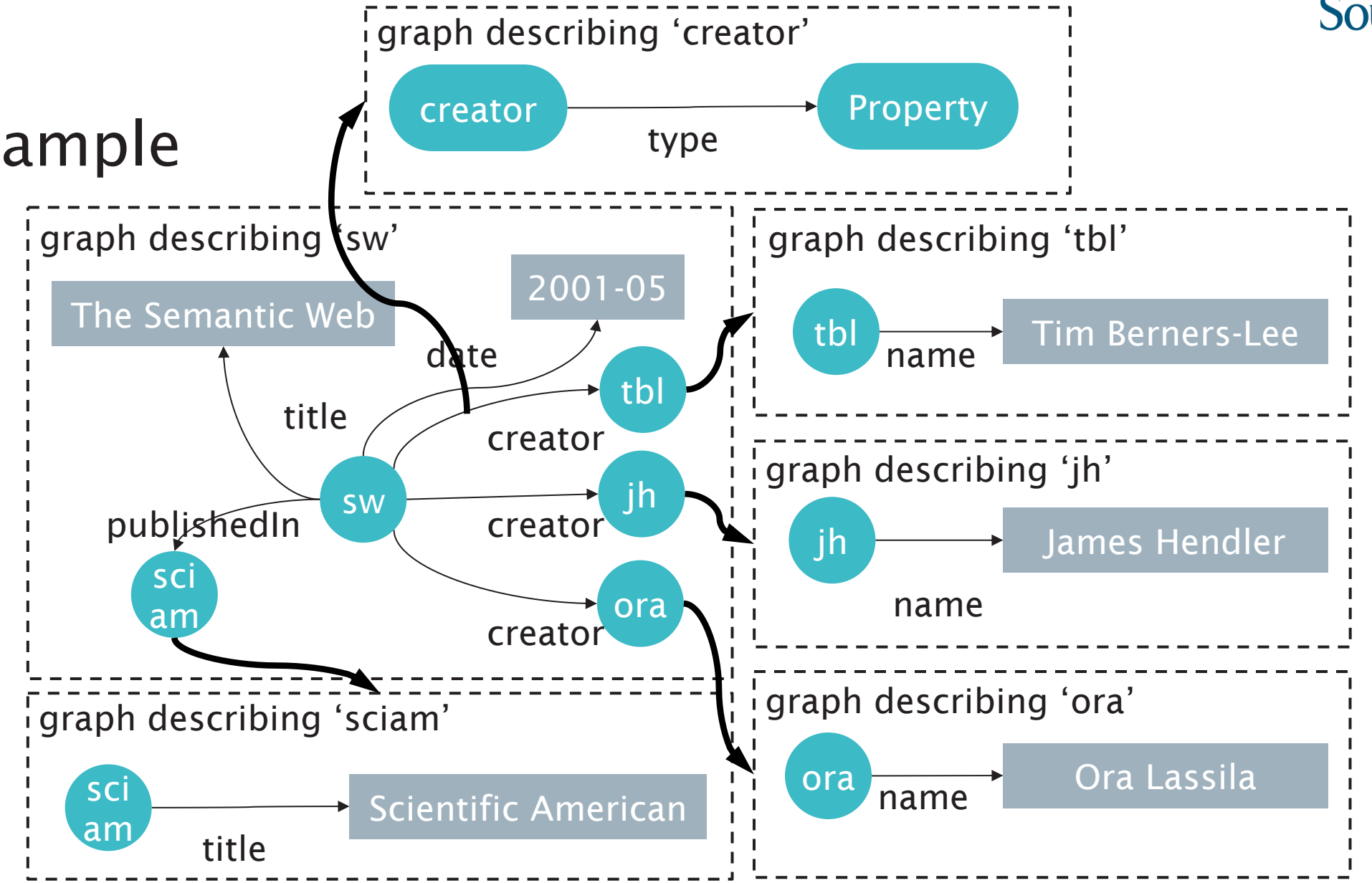
Set of publishing practices for Semantic Web data:

1. Use URIs as names for things
2. Use HTTP URIs so that people can look up those names
3. When someone looks up a URI, provide useful information
4. Include links to other URIs. so that they can discover more things

Effectively, putting the hypertext back into the Semantic Web

Simplifies integration between datasets while maintaining loose coupling

Example


graph describing 'creator'
creator
type
Property
graph describing 'sw'
The Semantic Web
2001-05
date
title
creator
tbl
sw
jh
creator
publishedIn
sci
am
ora
creator
graph describing 'tbl'
tbl
name
Tim Berners-Lee
graph describing 'jh'
jh
name
James Hendler
graph describing 'ora'
ora
name
Ora Lassila
graph describing 'sciam'
sci
am
Scientific American
title

UNIVERSITY OF Southampton

Resource Description Framework

Resource Description Framework

RDF is a framework for representing information about resources

- A data model (triples)
- Builds on Web architecture (uses URIs to identify resources and relations)
- Model-theoretic semantics (machine understandable)
- Many serialisation formats (RDF/XML, Turtle, JSON-LD, RDFa, etc)

Turtle: The Terse RDF Triple Language

Simple syntax derived from earlier RDF/N3 notation designed by Tim Berners-Lee

- Resource URIs are written in angle brackets: `<http://example.org>`
- Literal values are written in double quotes: `"like this”`
- Triples terminated with a full stop: `.`
- Whitespace not relevant

Triples in Turtle

Literals as objects:

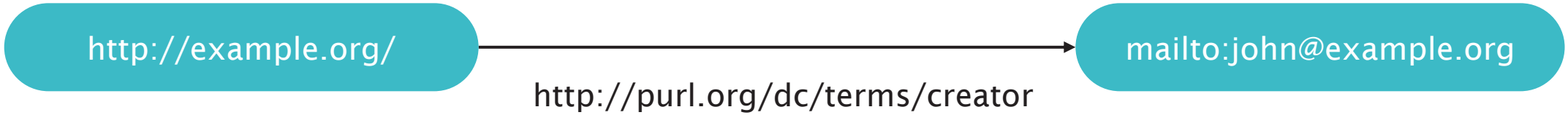
```
<http://www.sciam.com> <http://purl.org/dc/terms/title> “Scientific American” .
```

subject predicate object

Triples in Turtle

Resources as objects:

```
<http://example.org> <http://purl.org/dc/terms/creator> <mailto:john@example.org> .
```



Namespaces and qualified names

- RDF syntaxes use namespaces to abbreviate URIs to qualified names (QNames)
- A QName consists of a namespace prefix, a colon and a local name
  - e.g. rdf:type, dc:creator, foaf:Person
- Namespace prefixes correspond to URI prefixes

For example:

- Given a namespace prefix of `rdf` for the URI `http://www.w3.org/1999/02/22-rdf-syntax-ns#`
- The QName `rdf:type` would expand to `http://www.w3.org/1999/02/22-rdf-syntax-ns#type`

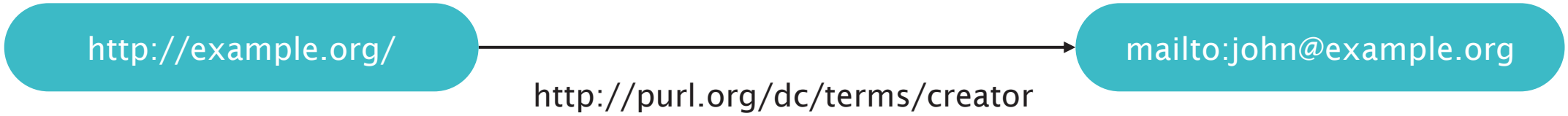
Namespaces in Turtle

Defined using `@prefix`

```
@prefix dc: <http://purl.org/dc/terms/> .

<http://example.org> dc:creator <mailto:john@example.org> .
```

note: no angle brackets

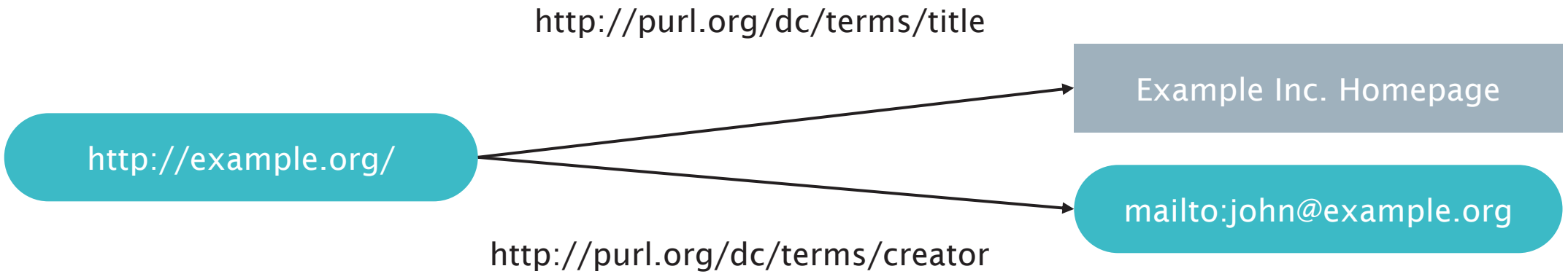


Repeated subjects

Use semicolon ;

```
@prefix dc: < http://purl.org/dc/terms/> .

<http://example.org> dc:title "Example Inc. Homepage" ;
                     dc:creator <mailto:john @example.org> .
```

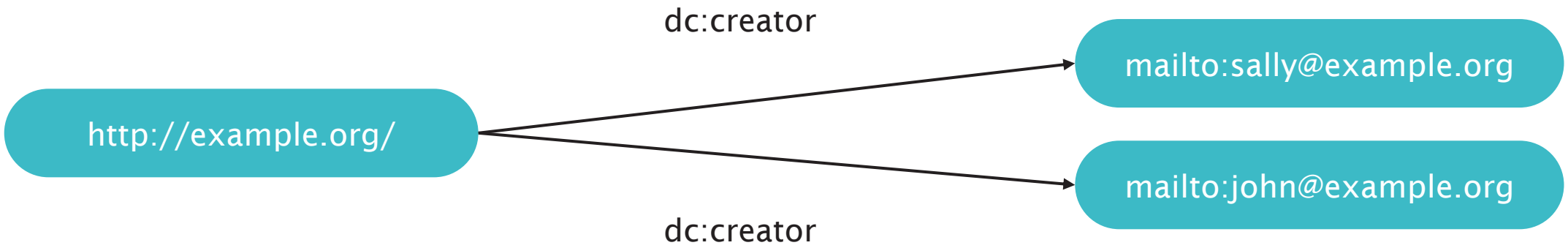


Repeated subjects and predicates

Use comma ,

```
@prefix dc: < http://purl.org/dc/terms/> .

<http://example.org> dc:creator <mailto:john @example.org> ,
                                <mailto:sally@example.org> .
```



Datatypes

Literal values presented so far are plain and do not have a type

- Many applications need to be able to distinguish between different typed literals
- e.g. integer vs. date vs. decimal

RDF uses XML Schema datatypes:

```
@prefix dc: < http://purl.org/dc/terms/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<http://example.org/> dc:date “2024-07-17”^^xsd:date .
```

UNIVERSITY OF Southampton

Knowledge Representation and Ontologies

Knowledge Representation

Long-standing concern in symbolic Artificial Intelligence

Knowledge representation is central to the Semantic Web

- Data published on the Semantic Web must be structured and organised

Most symbolic AI systems (and therefore SW systems) consist of:

- A knowledge base (KB)
  - Structured according to the knowledge representation approach taken
- An inference mechanism
  - Set of procedures that are used to examine the knowledge base to answer questions, solve problems or make decisions within the domain

Ontologies

An ontology is a **specification** of a **conceptualisation**

- **Specification**: A formal description
- **Conceptualisation**: The objects, concepts, and other entities that are assumed to always exist in some area of interest and the relationships that hold among them

An ontology is a **vocabulary** used to describe a certain reality, plus a set of explicit assumptions regarding its intended meaning

The vocabularies we talked about earlier (i.e. collections of properties) are very simple ontologies

UNIVERSITY OF Southampton

RDF Schema

RDF Vocabulary Description Language

RDF lets us make assertions about resources using a given vocabulary

RDF does not let us define these domain vocabularies by itself

RDF Schema is an RDF vocabulary which we can use to define other RDF vocabularies

- Define classes of objects
- Define hierarchies of classes
- Define properties that relate objects to each other
- Define hierarchies of properties
- Define domains/ranges of properties

Notes on RDF and RDFS namespaces

Most terms in RDF Schema are defined as part of the RDFS namespace

- `http://www.w3.org/2000/01/rdf-schema#` , abbreviated here as `rdfs:`

Two terms are defined as part of the RDF namespace: rdf:type and rdf:Property

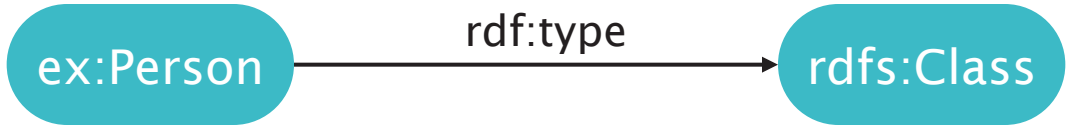
- `http://www.w3.org/1999/02/22-rdf-syntax-ns#` , abbreviated as `rdf:`

This is a historical accident, but can trip up the unwary

Be careful when using these terms in SPARQL queries!

RDF Schema class definitions

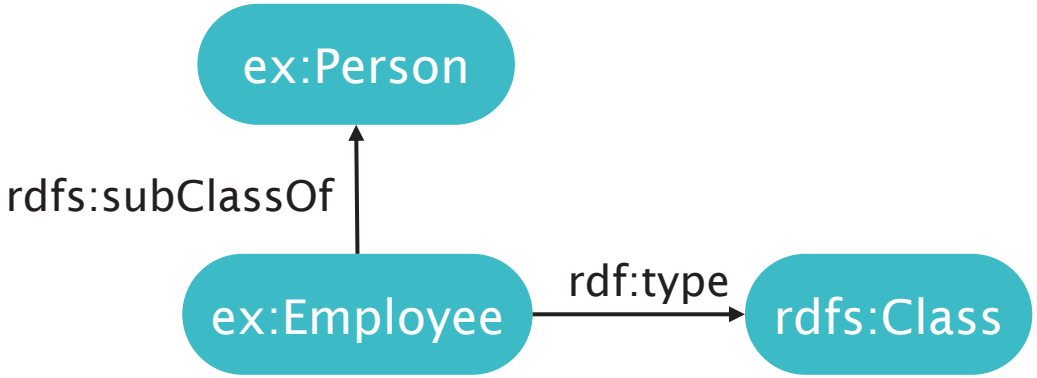
We wish to define the class Person:




```
ex:Person rdf:type rdfs:Class .
```

RDF Schema class definitions

Employee is a subclass of Person

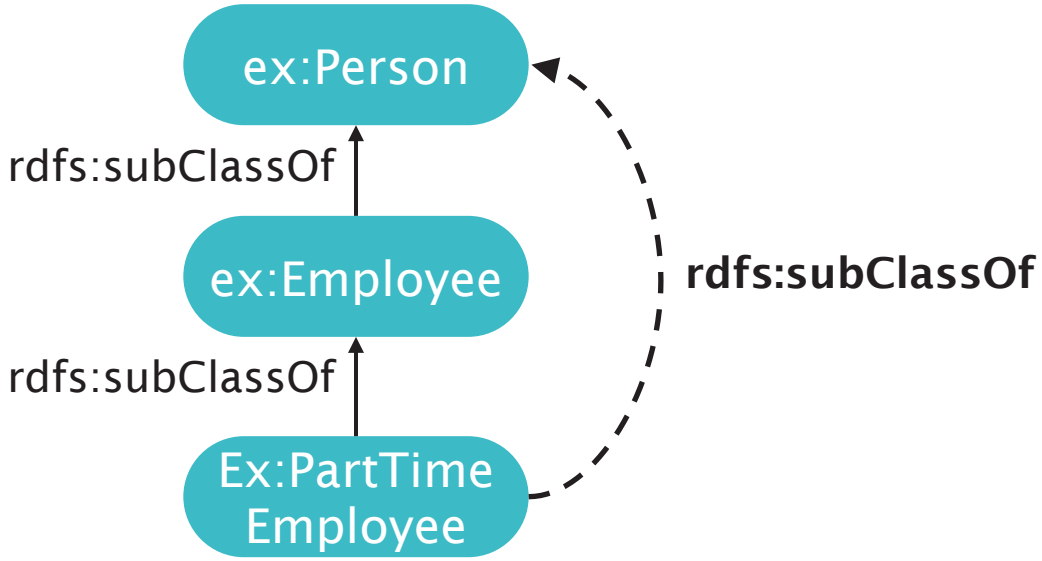



```
ex:Employee rdf:type rdfs:Class ;
            rdfs:subClassOf ex:Person .
```

RDF Schema class semantics

rdfs:subClassOf is transitive

(A rdfs:subClassOf B) and (B rdfs:subClassOf C) implies (A rdfs:subClassOf C)



RDF Schema class semantics

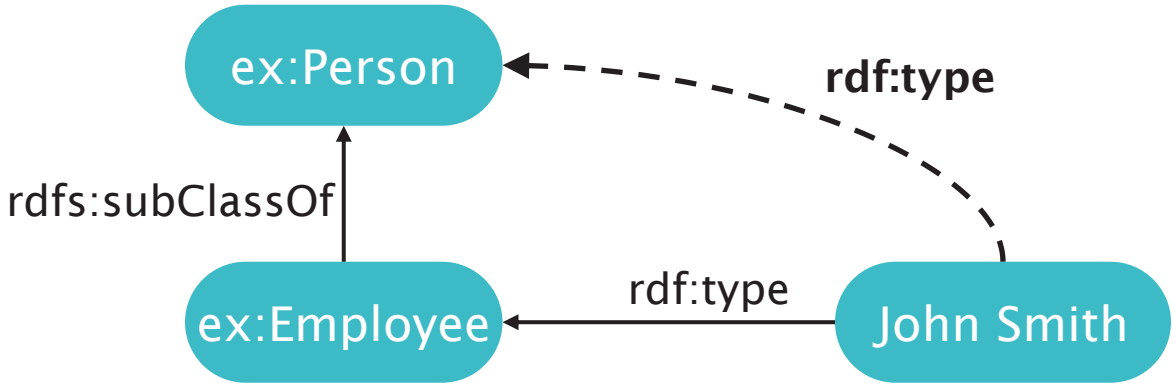
rdfs:subClassOf is reflexive

- All classes are subclasses of themselves

RDF Schema class semantics

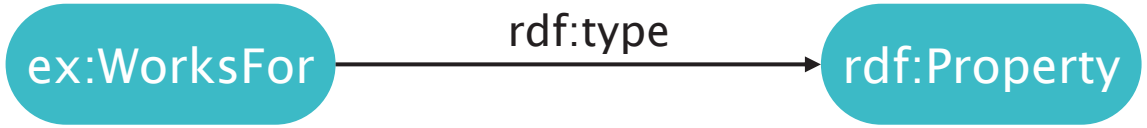
rdf:type distributes over rdf:subClassOf

- (A rdfs:subClassOf B) and (C rdf:type A) implies (C rdf:type B)



RDF Schema property definitions

We wish to define the property worksFor:




```
ex:WorksFor rdf:type rdf:Property .
```

RDF Schema property definitions

Important difference between RDF and object-oriented programming languages

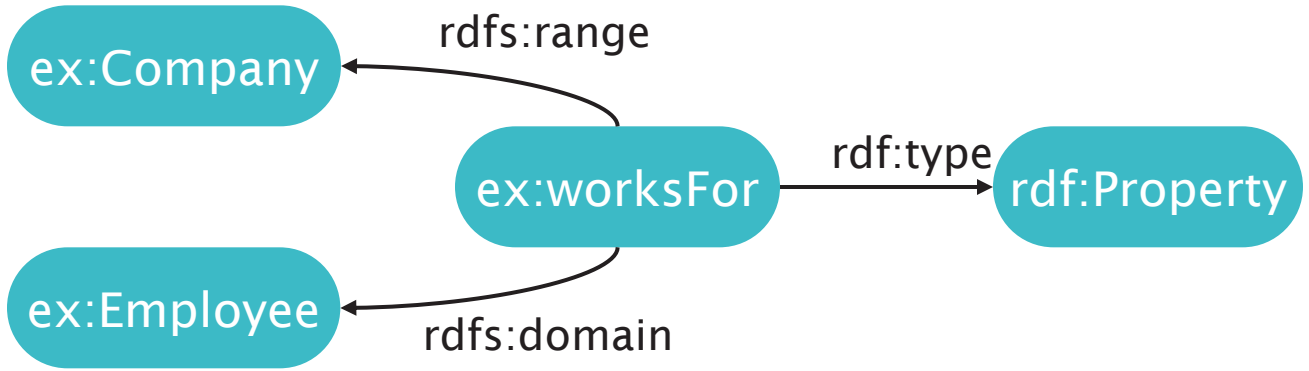
- OO languages define classes in terms of the properties they have
- RDF defines properties in terms of the classes whose instances they relate to each other

The *domain* of a property is the class that the property runs *from*

The *range* of a property is the class that a property runs *to*

RDF Schema property definitions

The property worksFor relates objects of class Employee to objects of class Company

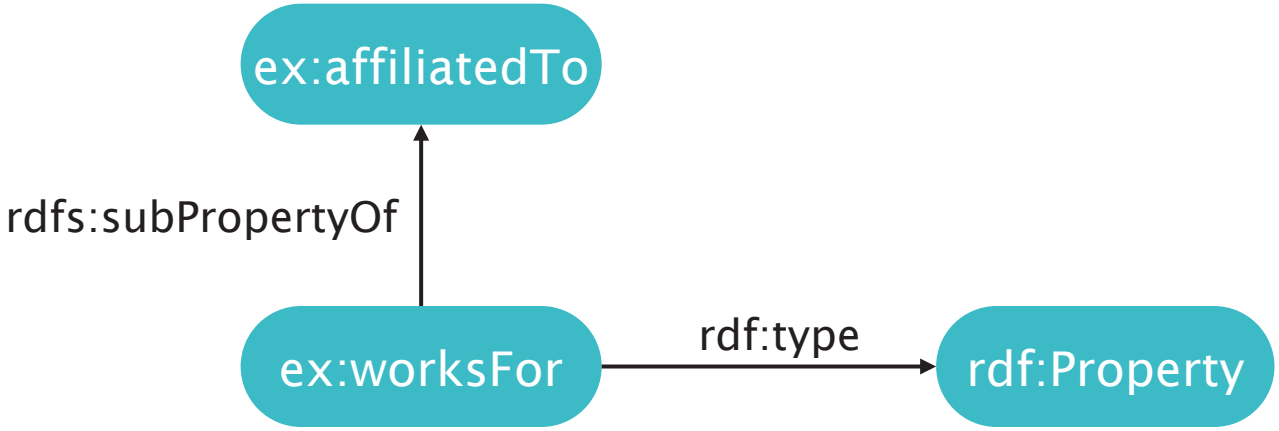



```
ex:worksFor rdf:type rdf:Property ;
            rdfs:domain ex:Employee ;
            rdfs:range ex:Company .
```

RDF Schema property definitions

Specialisation exists in properties as well as classes

- worksFor is a subproperty of affiliatedTo

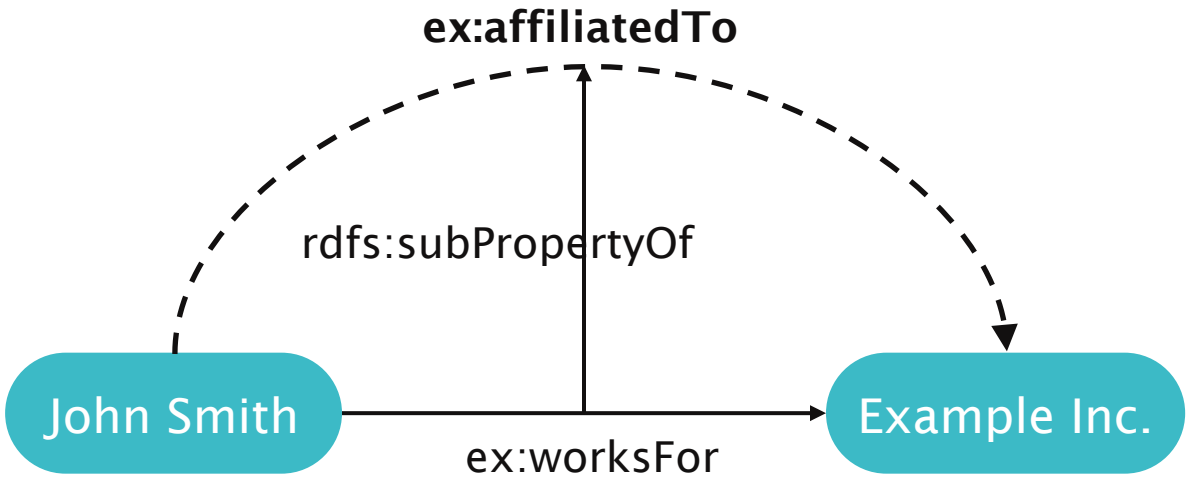



```
ex:worksFor rdf:type rdf:Property ;
            rdfs:subPropertyOf ex:affiliatedTo
```

RDF Schema property semantics

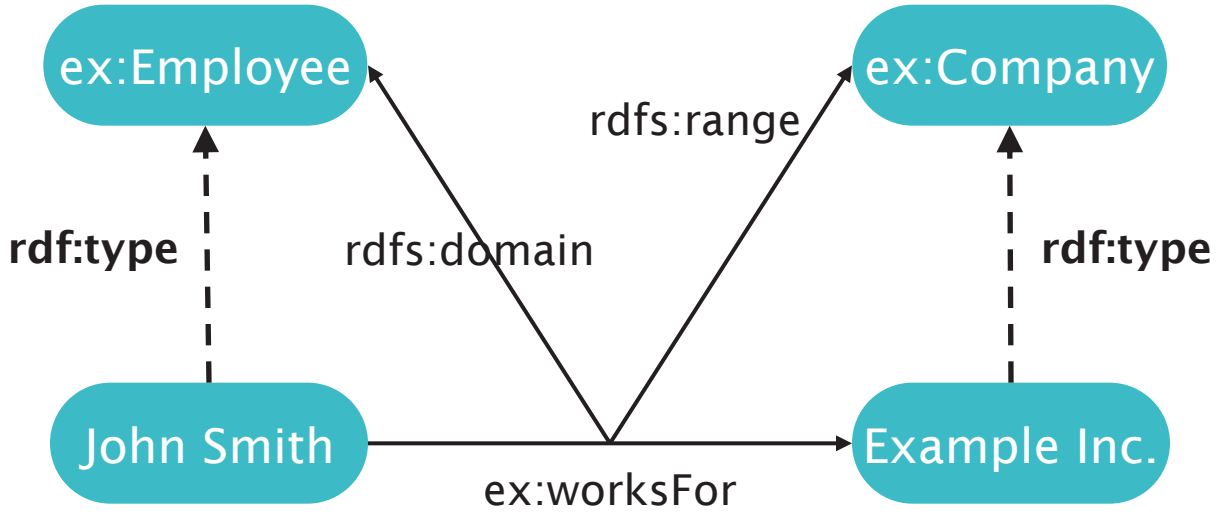
rdfs:subPropertyOf is transitive and reflexive

- Entailment of superproperties



RDF Schema property semantics

Type entailments from range and domain constraints



UNIVERSITY OF Southampton

The Ontology Design Lifecycle

Specification

Why are you building this ontology?

- Who will use this ontology?
- What will they use it for?

What is the domain of interest?

- How much detail do you need?

What questions do you need the ontology to answer?
(competency questions)

Conceptualisation

What classes exist in your ontology?

What properties relate them to each other?

What unique names are you going to use for these classes/properties?

Formalisation

What are the class/property hierarchies?

Are things classes or instances of classes?

What other constraints/restrictions do you need to express?

Implementation

Choose a language (RDF Schema, OWL)

Use an editor to build the ontology (Protégé)

Use a reasoner to check it as you build it

Evaluation

Is your ontology consistent?
(use a reasoner to check)

Does your ontology do what you set out to do?
(can it answer your competency questions?)

Documentation

Crucial for future usability and understanding of the ontology

- Brief definition of each class/property
- Synonyms for each class/property
- Rationale and assumptions for each class/property (why model it, how is it distinct from other classes/properties?)
- Example instances of each class/property

UNIVERSITY OF Southampton

Exercise:

Modelling a university

Protégé

Ontology editor developed by Stanford University

- Pre-dates the Semantic Web
- Heavily extended by researchers at the University of Manchester

Integrates reasoning into the ontology design process

- Check your ontology for consistency, subsumption, etc
- Available DL reasoners:
  - HermiT – http://www.hermit-reasoner.com/
  - Pellet – http://pellet.owldl.com/
  - FaCT++ – http://owl.man.ac.uk/factplusplus

UNIVERSITY OF Southampton

Description Logics

Beyond RDF Schema

RDF Schema is not expressive enough for many applications

- Only supports explicit class/property hierarchies
- Only supports global range and domain constraints
- There are things that you can’t infer

Description Logics

A *family* of knowledge representation formalisms

- A subset of first order predicate logic (i.e. more expressive than RDF Schema)
- Decidable – trade-off expressivity against algorithmic complexity
- Well understood – derived from work in the mid-80s to early 90s
- Model-theoretic formal semantics
- Simpler syntax than first order predicate logic

Description Logics

Description logics restrict the predicate types that can be used

- Unary predicates denote class membership

$$Person(x)$$

- Binary predicates denote properties that relate instances

$$hasChild(x, y)$$

Description Logic reasoning tasks

Satisfaction

- “Can this class have any instances?"

Subsumption

- "Is every instance of this class necessarily an instance of that class?"

Classification

- "What classes is this object an instance of?"

Defining ontologies with Description Logics

Describe classes (concepts) in terms of their necessary and sufficient conditions

Consider an attribute A of a class C:

- Attribute A is a **necessary** condition for membership of C
  - If an object is an instance of C, then it has A
- Attribute A is a **sufficient** condition for membership of C
  - If an object has A, then it is an instance of C

Expressions

Description logic expressions consist of:

- Class and property descriptions:
  - Atomic classes: Person
  - Atomic properties: has child
  - Complex classes: “A person with two living parents”
  - Complex properties: “has parent’s brother” (i.e. "has uncle")
- Axioms that make statements about how concepts or roles are related to each other:
  - “Every person with two living parents is thankful”
  - “hasUncle is equivalent to has parent’s brother”

Constructors

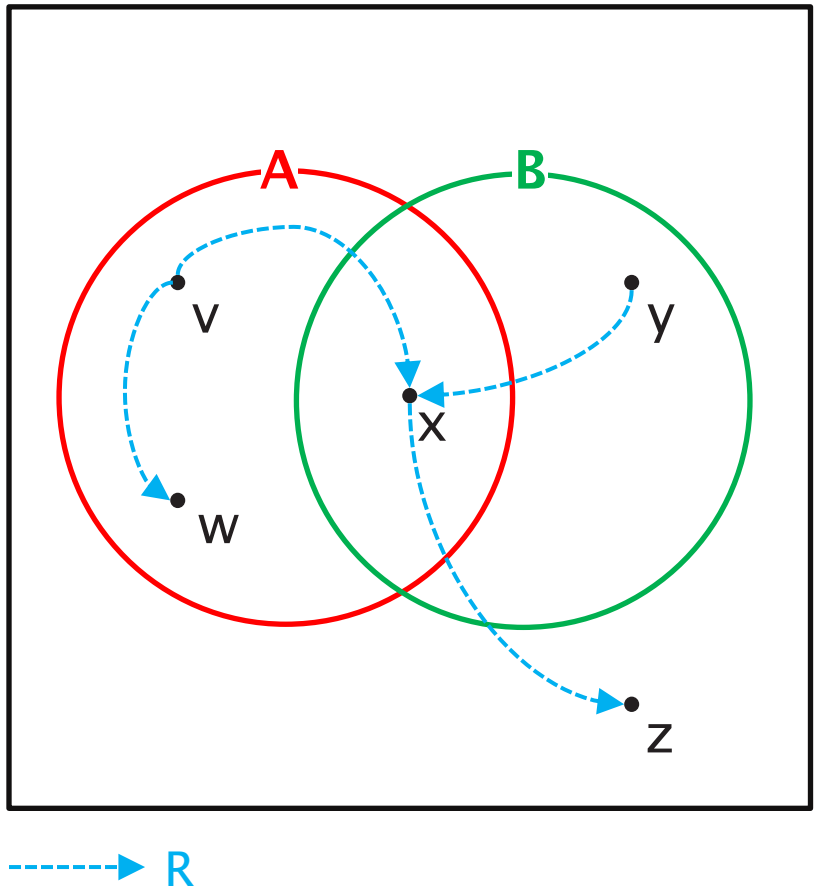
Used to construct complex classes:

- Boolean classes constructors $\neg C$ $C \sqcup D$ $C \sqcap D$
- Restrictions on properties $\forall R.C$ $\exists R.C$
- Number/cardinality restrictions $\leq n\, R$ $\geq n\, R$ $= nR$
- Nominals (singleton classes) $\{x\}$
- Universal class, top $\top$
- Contradiction, bottom $\bot$

Used to construct complex properties:

- Concrete domains (datatypes)
- Inverse properties R^-
- Property composition $R \circ S$
- Transitive properties R^+

Classes as sets

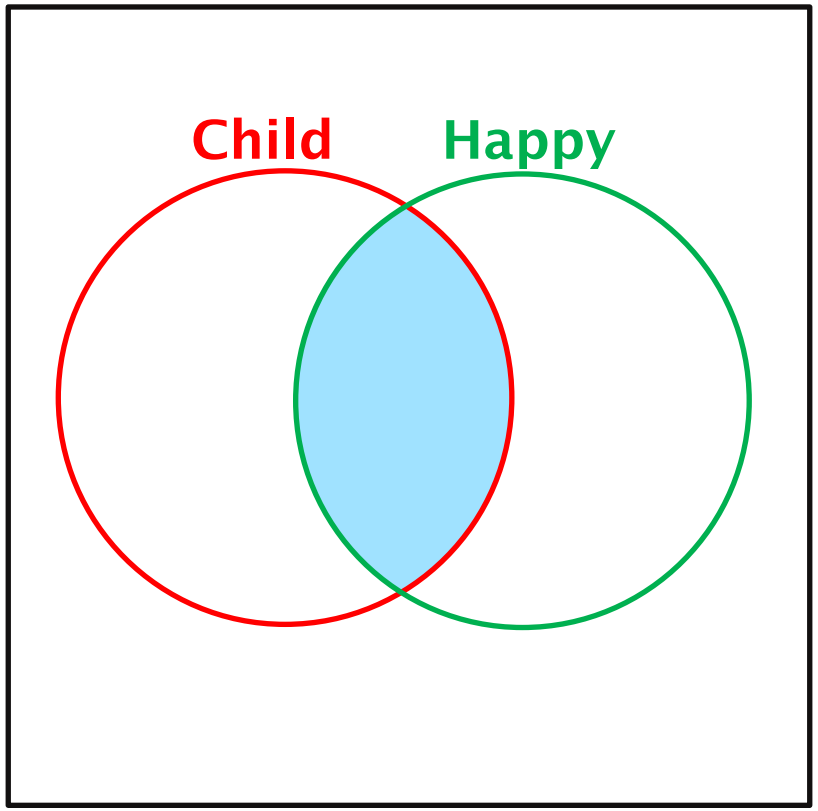


Class Intersection

Child ⊓ Happy

The class of things which are both children and happy

Read as “Child **and** Happy”

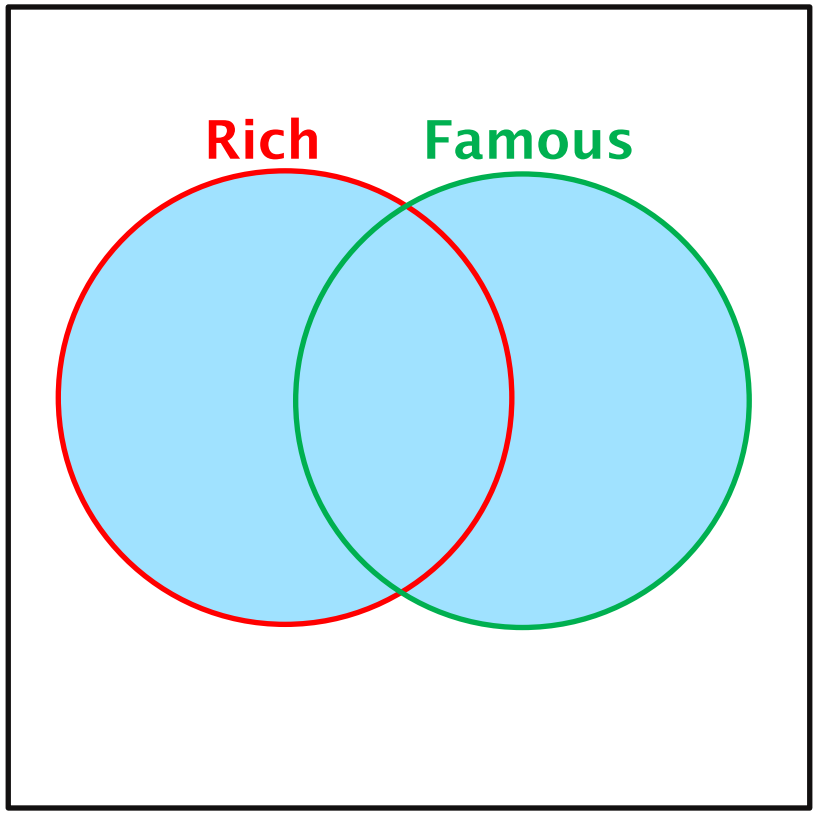


Class Union

Rich ⊔ Famous

The class of things which are rich or famous (or both)

Read as “Rich **or** Famous”

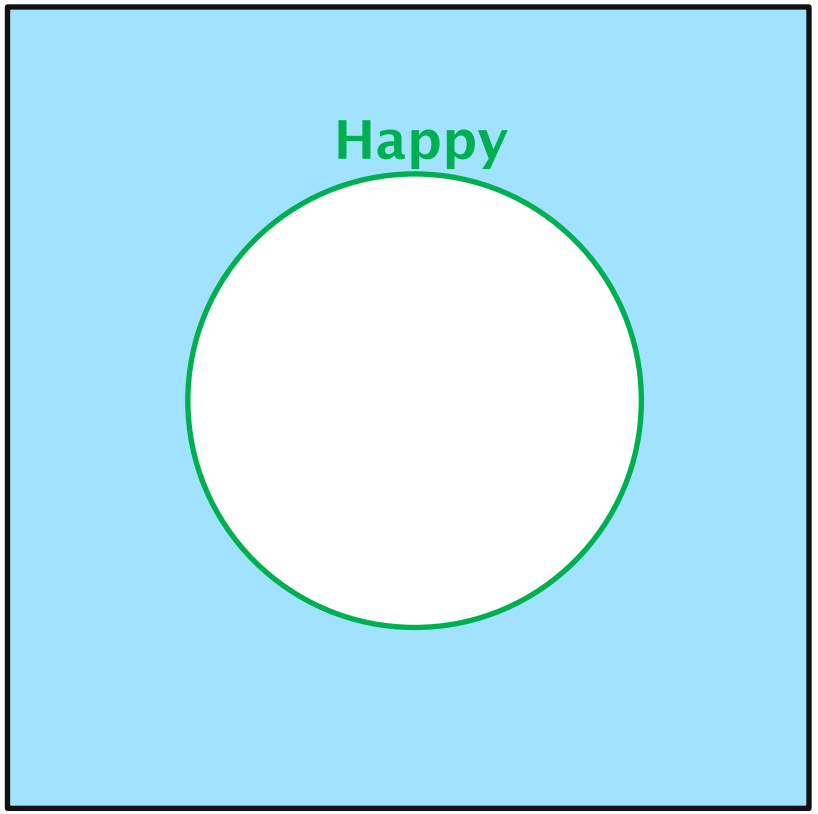


Class Complement

¬Happy

The class of things which are not happy

Read as “**not** Happy”



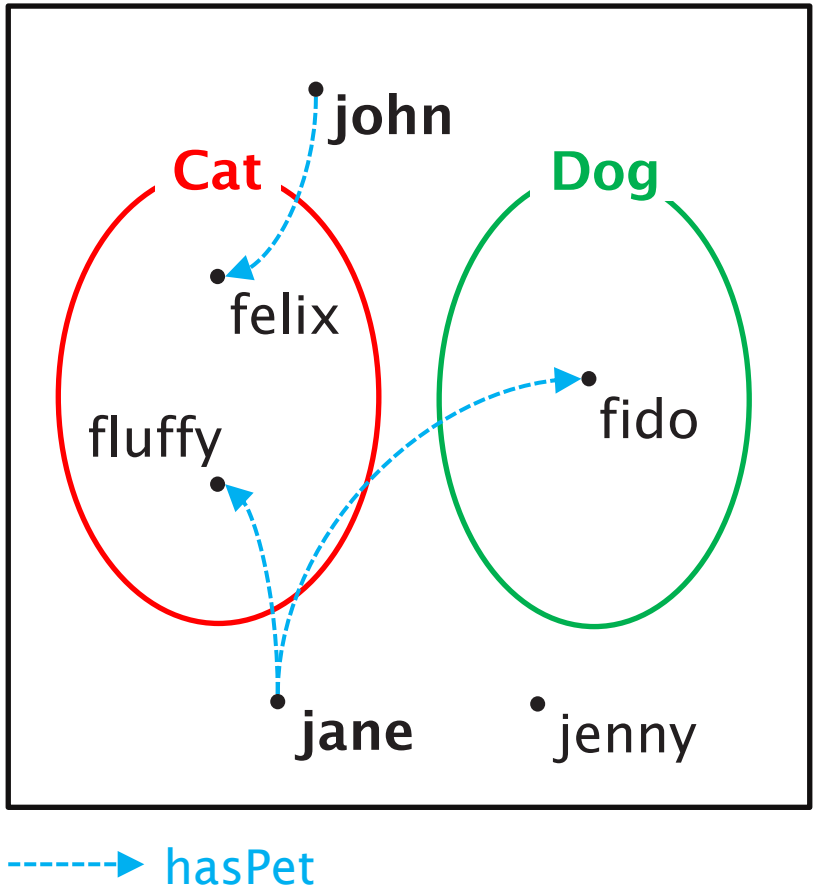
Existential Restriction

∃hasPet. Cat

The class of things which have some pet that is a cat

- must have at least one pet

Read as “hasPet **some** Cat”



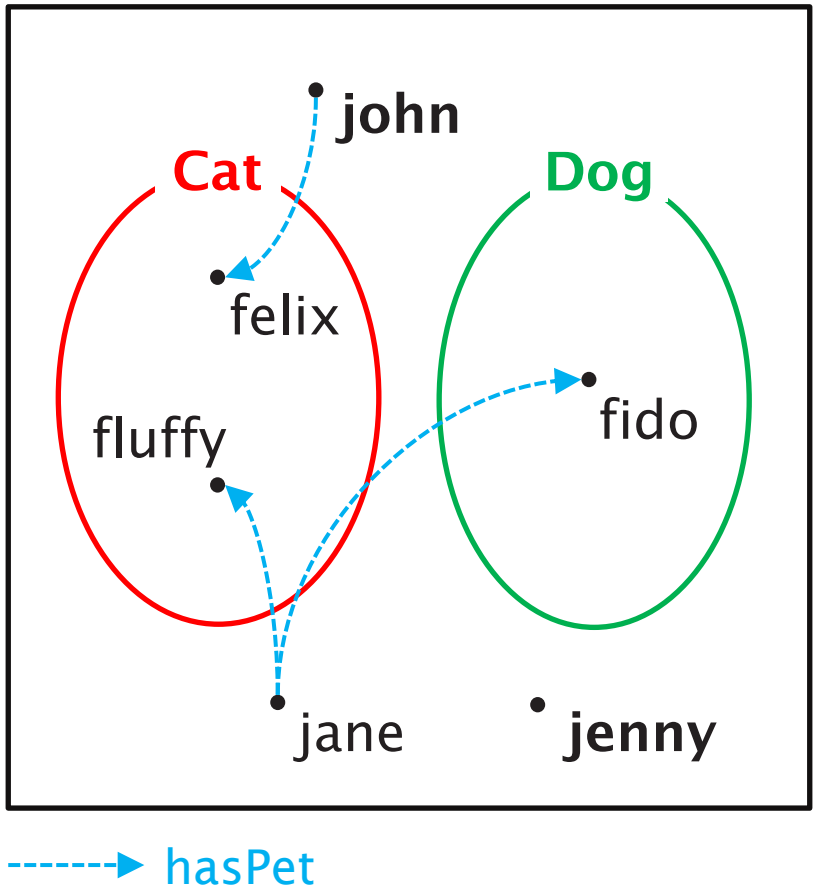
Universal Restriction

∀hasPet. Cat

The class of things all of whose pets are cats

- Or, which only have pets that are cats
- includes those things which have no pets

Read as “hasPet **only** Cat”



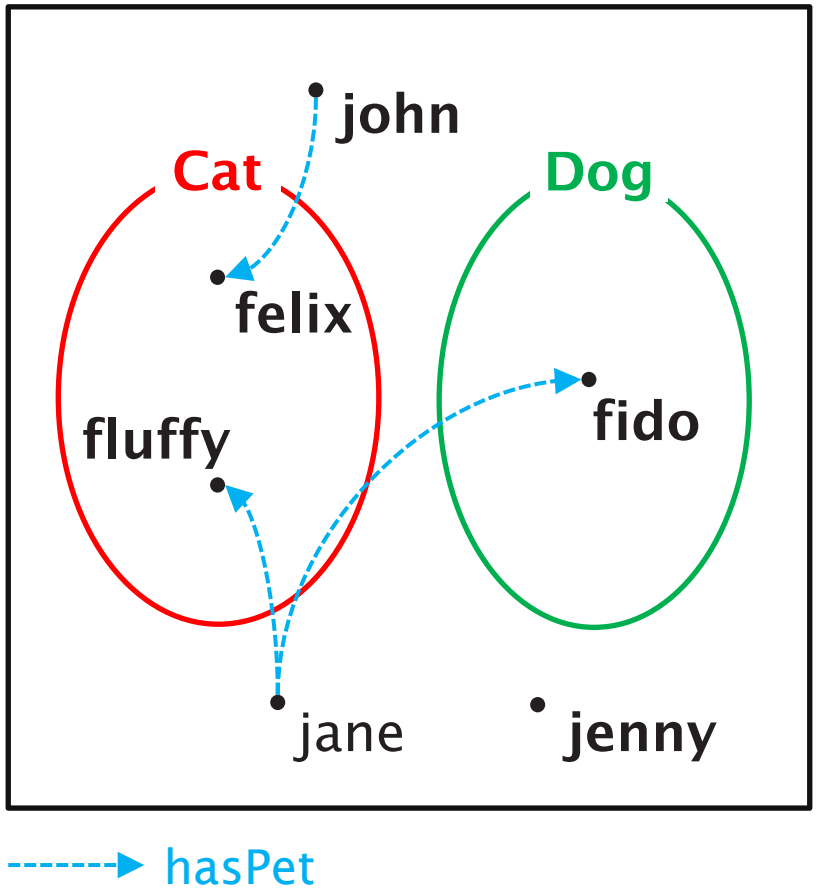
Universal Restriction

∀hasPet. Cat

The class of things all of whose pets are cats

- Or, which only have pets that are cats
- includes those things which have no pets

Read as “hasPet **only** Cat”

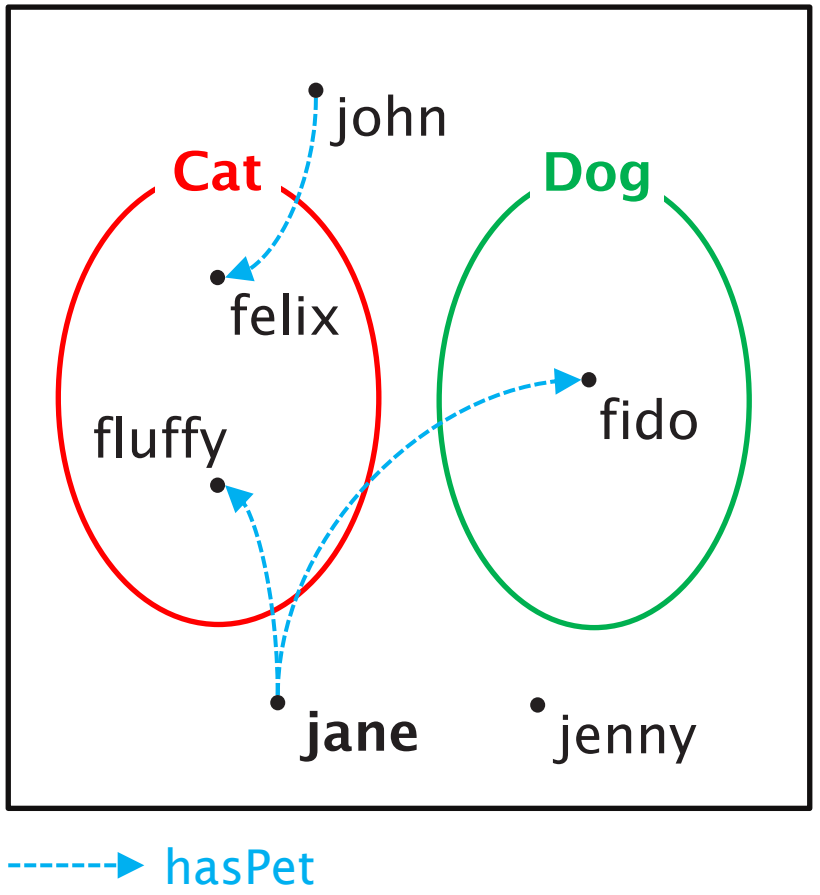


Value Restriction

∃hasPet.{fido}

The class of things which have a particular pet called Fido

Read as “hasPet **value** fido”

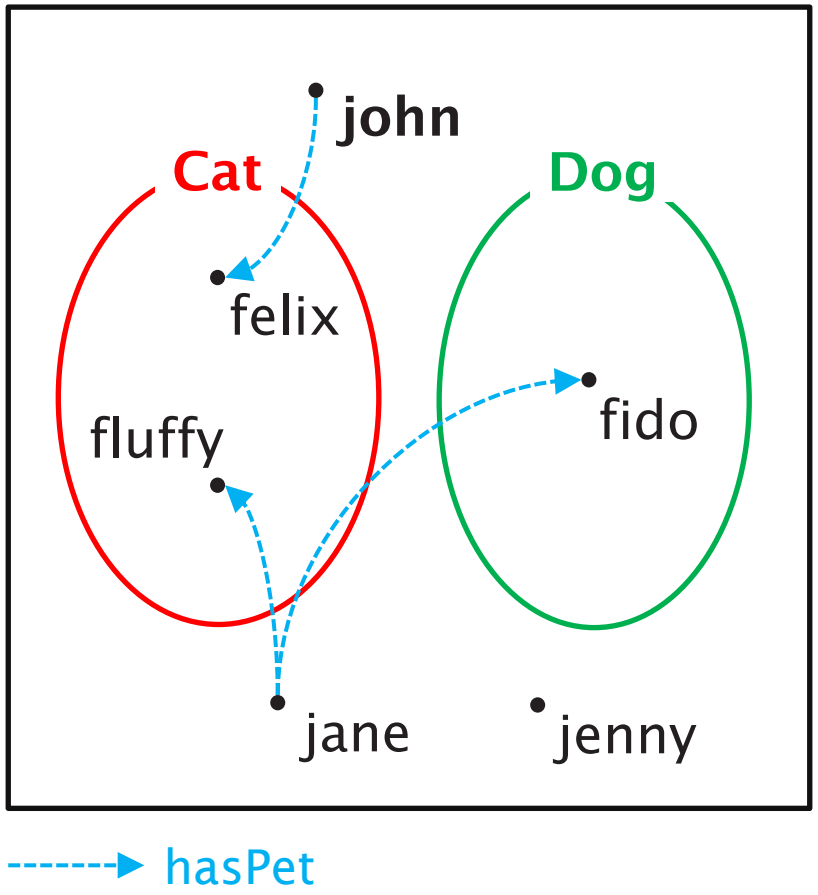


Cardinality Restriction

$= 1$ hasPet

The class of things which have exactly one pet

Read as “hasPet **exactly** 1”

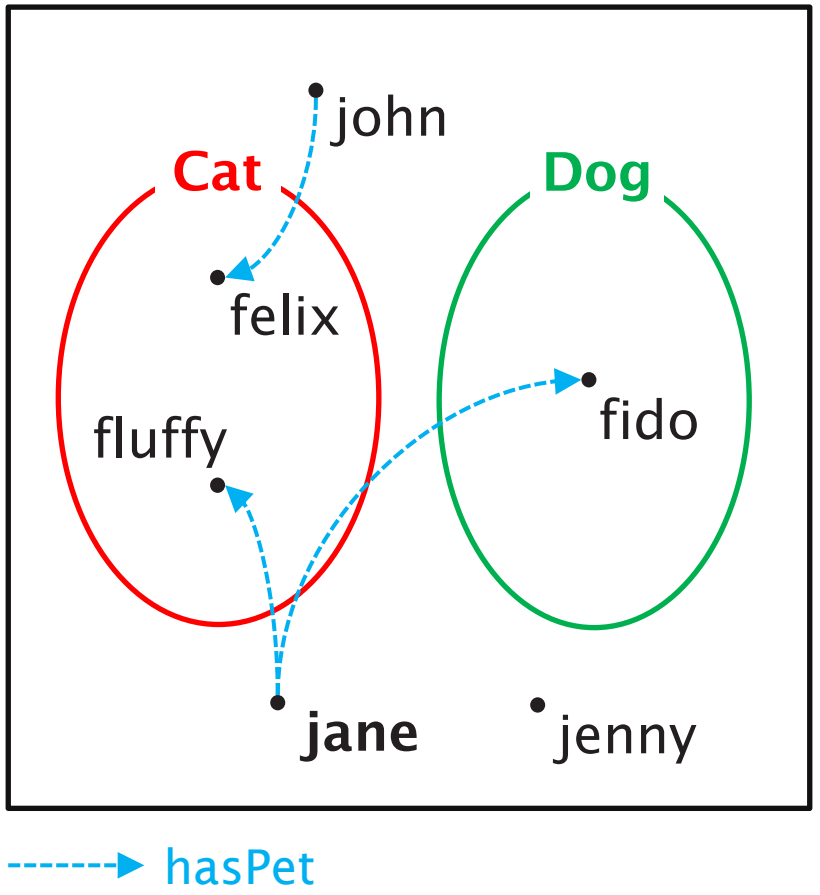


Cardinality Restriction

≥ 2 hasPet

The class of things which have at least two pets

Read as “hasPet **min** 2”

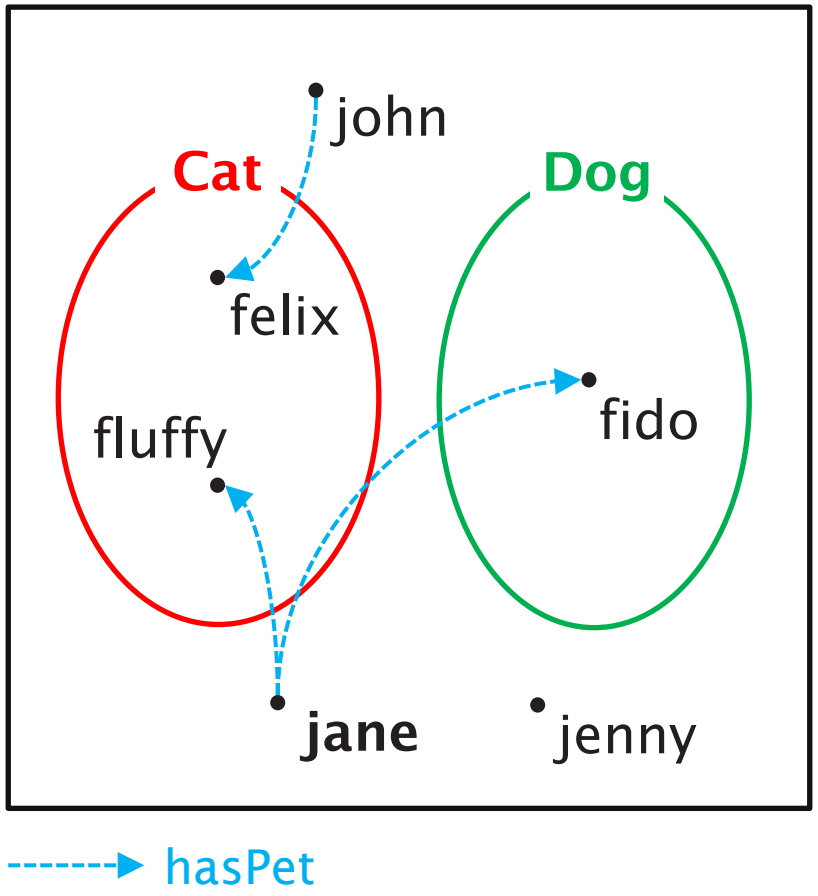


Qualified Cardinality Restriction

$\geq 2\ \mathrm{hasPet}\,.\,\mathrm{Cat}$

The class of things which have at least two pets that are cats

Read as “hasPet **min** 2 Cat”



Property characteristics

- Functional
- Inverse functional
- Transitive
- Symmetric (is its own inverse)
- Asymmetric (is disjoint with its inverse)
- Reflexive (relates every object to itself)
- Irreflexive (relates no object to itself)

Revisiting Necessary and Sufficient Conditions

Membership of D is a **necessary** condition for membership of C

$C \sqsubseteq D$ (C is a **primitive** or **partial** class)

Membership of D is a **sufficient** condition for membership of C

$C \sqsupseteq D$

Membership of D is both a **necessary** and a **sufficient** condition for membership of C

$C \equiv D$ (C is a **defined** class)

Ontology axioms (TBox)

C is a subclass of D	$C \sqsubseteq D$
C is equivalent to D	$C \equiv D$
R is a subproperty of S	$R \sqsubseteq S$
R is equivalent to S	$R \equiv S$

Ontology axioms (TBox)

C is disjoint with D	$C \sqcup D \equiv \bot$
R has a domain of C	$\exists R.\top \sqsubseteq \mathrm{C}$
R has a range of C	$\top \sqsubseteq \forall R.C$
R is symmetric	$R \equiv R^{-}$
R is transitive	$R^{+} \sqsubseteq R$
R is functional	$\top \sqsubseteq (\leq 1\, R)$

Instance assertions (ABox)

Axioms describing a concrete situation

Class instantiation

$C(x)$

- x is of type C

Property instantiation

$R(x, y)$

- x has R of y

Description Logic Semantics

Δ is the domain (non-empty set of individuals)

Interpretation function $\cdot^{\mathcal{I}}$ maps:

- Class expressions to their extensions
  (set of instances of that class, subsets of Δ)
- Properties to subsets of $\Delta \times \Delta$
- Individuals to elements of Δ

Examples:

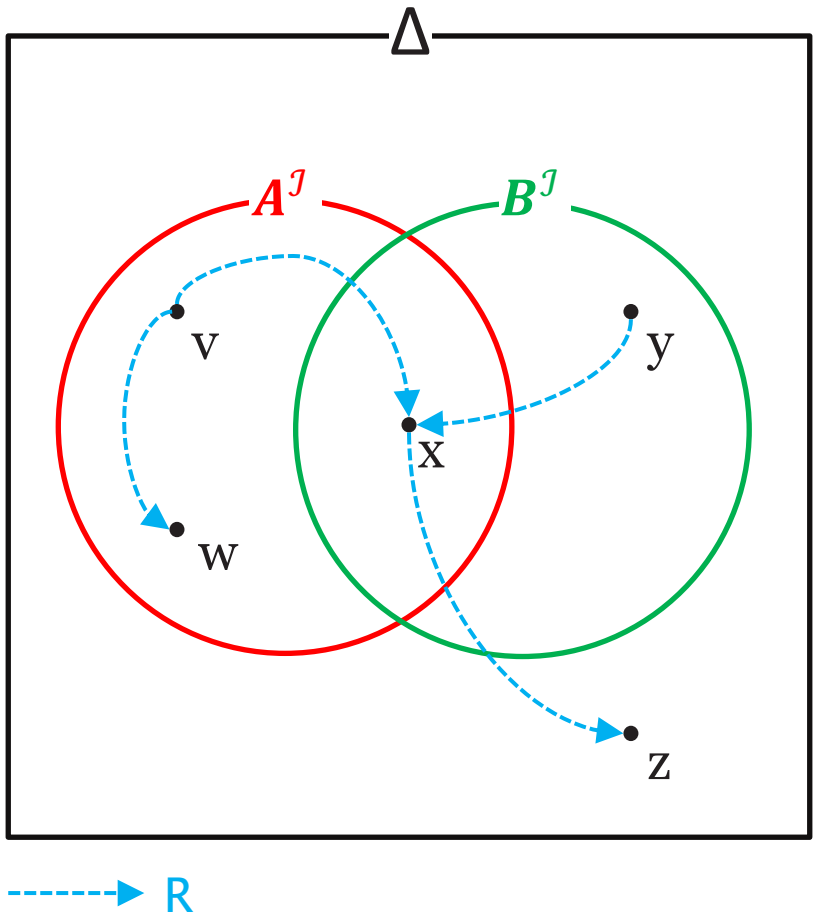
- $C^{\mathcal{I}}$ is the set of members of C
- $(C \sqcup D)^{\mathcal{I}}$ is the set of members of either C or D

Description Logic Semantics

Syntax	Semantics	Notes
$(C \sqcap D)^{\mathcal{J}}$	$C^{\mathcal{J}} \cap D^{\mathcal{J}}$	Conjunction
$(C \sqcup D)^{\mathcal{J}}$	$C^{\mathcal{J}} \cup D^{\mathcal{J}}$	Disjunction
$(\neg C)^{\mathcal{J}}$	$\Delta \backslash C^{\mathcal{J}}$	Complement
$(\exists R.C)^{\mathcal{J}}$	$\{x \mid \exists y\,.\langle x,y\rangle \in R^{\mathcal{J}} \wedge y \in C^{\mathcal{J}}\}$	Existential
$(\forall R.C)^{\mathcal{J}}$	$\{x \mid \forall y\,\langle x,y\rangle \in R^{\mathcal{J}} \Rightarrow y \in C^{\mathcal{J}}\}$	Universal
$(\geq n\ R)^{\mathcal{J}}$	$\{x \mid \#\{y \mid \langle x,y\rangle \in R^{\mathcal{J}}\} \geq n\}$	Min cardinality
$(\leq n\ R)^{\mathcal{J}}$	$\{x \mid \#\{y \mid \langle x,y\rangle \in R^{\mathcal{J}}\} \leq n\}$	Max cardinality
$(= n\ R)^{\mathcal{J}}$	$\{x \mid \#\{y \mid \langle x,y\rangle \in R^{\mathcal{J}}\} = n\}$	Exact cardinality
$(\bot)^{\mathcal{J}}$	$\emptyset$	Bottom
$(\top)^{\mathcal{J}}$	Δ	Top

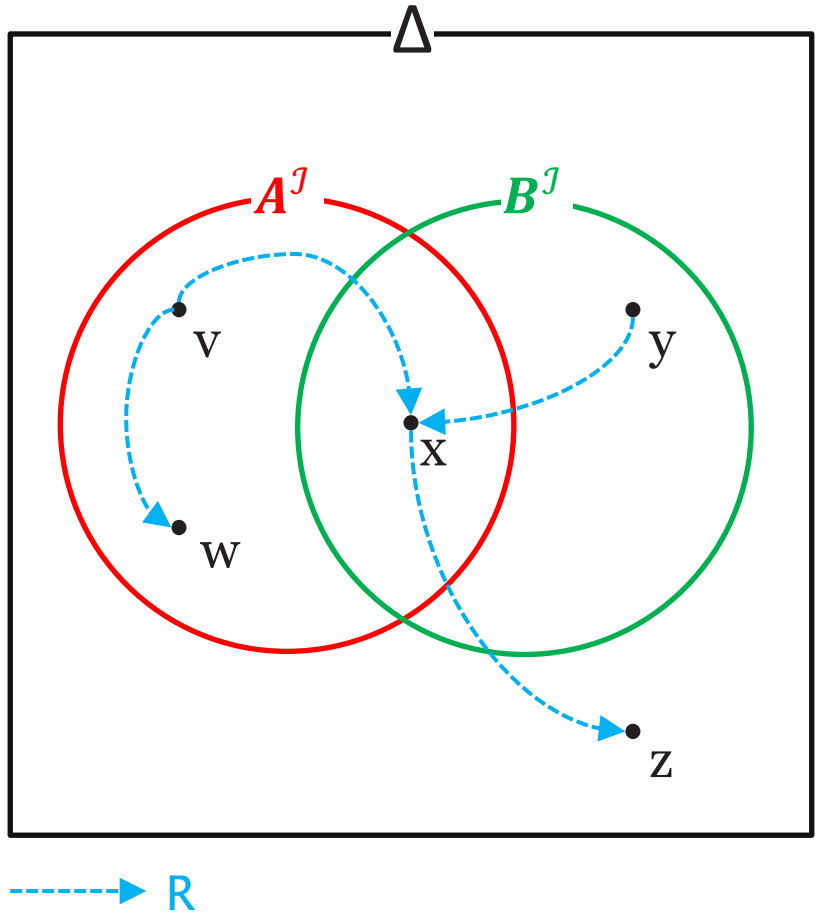
Interpretation Example

$\Delta = \{v, w, x, y, z\}$
$A^{\mathcal{I}} = \{v, w, x\}$
$B^{\mathcal{I}} = \{x, y\}$
$R^{\mathcal{I}} = \{\langle v, w\rangle, \langle v, x\rangle, \langle y, x\rangle, \langle x, z\rangle\}$



Interpretation Example

$(\neg B)^{\mathcal{J}} = \{\mathrm{v}, \mathrm{w}, \mathrm{z}\}$
$(A \sqcup B)^{\mathcal{J}} = \{\mathrm{v}, \mathrm{w}, \mathrm{x}, \mathrm{y}\}$
$(\neg A \sqcap B)^{\mathcal{J}} = \{\mathrm{y}\}$
$(\exists R.B)^{\mathcal{J}} = \{\mathrm{v}, \mathrm{y}\}$
$(\forall R.B)^{\mathcal{J}} = \{\mathrm{y}, \mathrm{w}, \mathrm{z}\}$
$\left(\exists R.(\exists R.A)\right)^{\mathcal{J}} = \{\}$
$\left(\exists R.\neg(A \sqcap B)\right)^{\mathcal{J}} = \{\mathrm{v}, \mathrm{x}\}$
$(\exists R^{-}.A)^{\mathcal{J}} = \{w, x, z\}$
$(R^{+})^{\mathcal{J}} = \{\langle v, w\rangle, \langle v, x\rangle, \langle v, z\rangle, \langle y, x\rangle, \langle y, z\rangle, \langle x, z\rangle\}$



Description Logic Reasoning Tasks Revisited

A description logic knowledge base comprises:

- A TBox defining classes and properties
- An ABox containing assertations about instances

$$K = \langle TBox, ABox \rangle$$

We have an interpretation $\mathcal{I} = \langle \Delta, \cdot^{\mathcal{I}} \rangle$ which maps the instances, classes and properties in K onto a domain Δ via an interpretation function $\cdot^{\mathcal{I}}$

We can redefine the reasoning tasks in terms of $\mathcal{I}$

Description Logic Reasoning Tasks Revisited

Satisfaction: Can this class have any instances?

A class C is satisfiable w.r.t a KB K iff there exists an interpretation $\mathcal{I}$ of K with $C^{\mathcal{I}} \neq \emptyset$

Subsumption: Is every instance of this class necessarily an instance of this other class?

A class C is subsumed by a class D w.r.t. a KB K iff for every interpretation $\mathcal{I}$ of K, $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$

Classification: Is this individual necessarily an instance of this class?

An individual x is an instance of class C w.r.t. a KB K iff for every interpretation $\mathcal{I}$ of K, $x^{\mathcal{I}} \in C^{\mathcal{I}}$

Reduction to Satisfaction

Tableau-based description logic reasoners reduce all reasoning tasks to satisfaction:

Subsumption

C is subsumed by $D \Leftrightarrow (C \sqcap \neg D)$ is unsatisfiable

Classification

x is an instance of $C \Leftrightarrow (\neg C \sqcap \{x\})$ is unsatisfiable

UNIVERSITY OF Southampton

Web Ontology Language (OWL)

Introducing the OWL Web Ontology Language

OWL is an ontology language that is:

- More expressive than RDF Schema
- Based on description logics
- Compatible with the semantics of RDF and RDF Schema

OWL Features

- Property restrictions (local range/cardinality/value/self constraints)
- Equivalence and identity relations
- Property characteristics (transitive, symmetric, asymmetric, functional, inverse, reflexive, irreflexive, disjoint)
- Complex classes (set operators, enumerated classes, disjoint classes)

UNIVERSITY OF Southampton

Exercise:

Modelling pizzas

Manchester DL syntax

The DL syntax we've used so far is a 'traditional' syntax for logical expressions

- Not well understood by non-logicians
- Not easy to type (lots of special symbols)

The Manchester DL syntax is a more user-friendly syntax for use in tools

- Used extensively in Protégé for class restrictions
- http://www.w3.org/TR/owl2-manchester-syntax/

Manchester Syntax Summary

Traditional DL Syntax	Manchester Syntax
$C \sqcap D$	C and D
$C \sqcup D$	C or D
$\neg C$	not C
$\exists R.C$	R some C
$\forall R.C$	R only C
$\geq n\ R$	R min n
$\leq n\ R$	R max n
$= n\ R$	R exactly n
$\exists R.\{x\}$	R value x
$\geq n\ R.C$	R min n C
Reflexive property	R Self
Datatype restrictions	int[>=2, <=15]