MANCHESTER 1824

The University of Manchester

Programming to the OWL API: Introduction

Sean Bechhofer

University of Manchester

sean.bechhofer@manchester.ac.uk

MANCHESTER 1824

The University of Manchester

Overview

- Motivation
  - Why?
- Details
  - What? Where?
- Samples/Examples
  - How?
- Wrap Up
  - What next?

MANCHESTER 1824

The University of Manchester

Assumptions

> *"The least questioned assumptions are often the most questionable"*
>
> *Paul Broca*

- Familiarity with Java
- (Some) Familiarity with Semantic Web Technologies:
  - RDF
  - RDF Schema
  - OWL
- Being at least aware of the existence of:
  - Description Logics

MANCHESTER 1824

The University of Manchester

The Semantic Web Vision

- The Web made possible through established standards
  - **TCP/IP** for transporting bits down a wire
  - **HTTP & HTML** for transporting and rendering hyperlinked text
- Applications able to exploit this common infrastructure
  - Result is the WWW as we know it
- Evolution
  - 1st generation web mostly handwritten HTML pages
  - 2nd generation (current) web often machine generated/active
- In the next generation web, resources should be more accessible to automated processes
  - To be achieved via semantic markup: metadata annotations that describe content/function
  - Coincides with vision of a Semantic Web

Need to Add "Semantics"

- External agreement on meaning of annotations
  - E.g., *Dublin Core* for annotation of library/bibliographic information
    - Agree on the meaning of a set of annotation tags
  - Problems with this approach
    - Inflexible
    - Limited
- Use Ontologies
  - Ontologies
  - New terms
    - "Conceptual Lego"
  - Meaning (semantics) of such terms is formally specified
  - Can also specify relationships between terms in multiple ontologies

Machine **Processable**

not

Machine **Understandable**

MANCHESTER 1824

The University of Manchester

Ontology in Computer Science

- An ontology is an engineering artifact:
  - It is constituted by a specific vocabulary used to describe a certain reality, plus
  - a set of explicit assumptions regarding the intended meaning of the vocabulary.
    - Almost always including how concepts should be classified

- Thus, an ontology describes a formal specification of a certain domain:
  - Shared understanding of a domain of interest
  - Formal and machine manipulable model of a domain of interest

MANCHESTER 1824

The University of Manchester

Building a Semantic Web

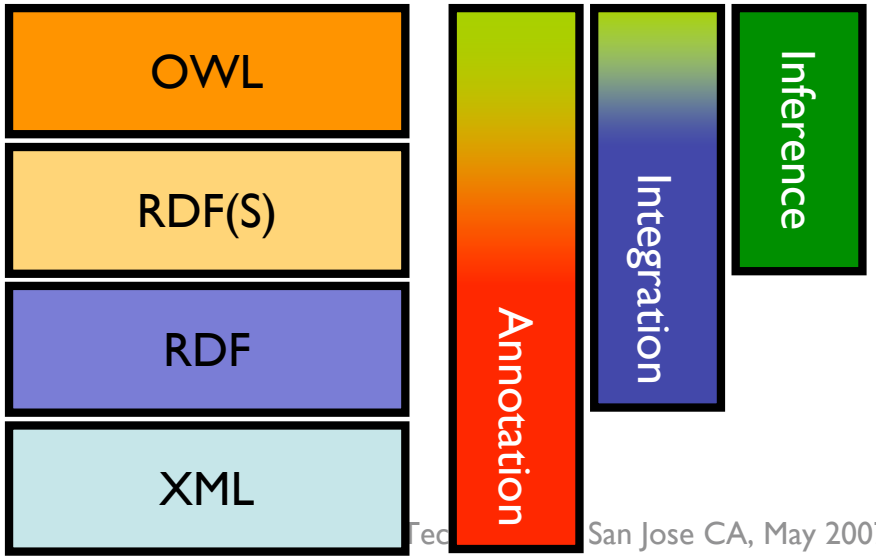
- Annotation
  - Associating metadata with resources
- Integration
  - Integrating information sources
- Inference
  - Reasoning over the information we have.
  - Could be light-weight (taxonomy)
  - Could be heavy-weight (logic-style)
- Interoperation and Sharing are key goals

MANCHESTER 1824

The University of Manchester

Languages

- Work on Semantic Web has concentrated on the definition of a collection or "stack" of languages.
  - These languages are then used to support the representation and use of metadata.
- The languages provide basic machinery that we can use to represent the extra semantic information needed for the Semantic Web
  - XML
  - RDF
  - RDF(S)
  - OWL
  - …



Why (Formal) Semantics?

- Increased formality makes languages more amenable to machine processing (e.g. automated reasoning).
- The formal semantics provides an unambiguous interpretation of the descriptions.
  - What does an expression in an ontology language **mean**?
  - The semantics of a language tell us precisely how to interpret a complex expression.
- Well defined semantics are vital if we are to support machine interpretability
  - They remove ambiguities in the interpretation of the descriptions.

MANCHESTER 1824

The University of Manchester

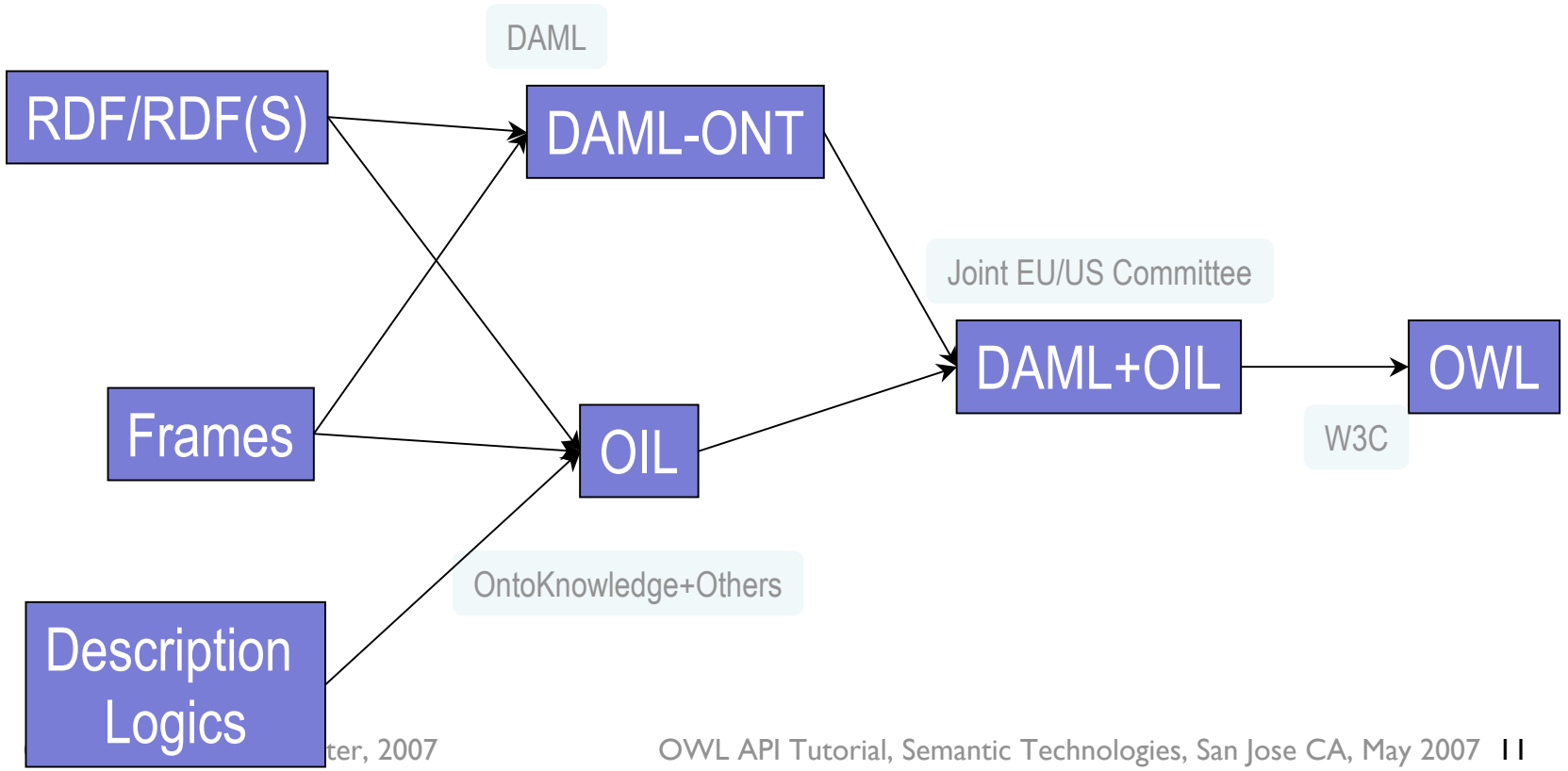
OWL

- OWL is a language for representing Ontologies in a Web context
  - **W**eb **O**ntology **L**anguage
- A W3C Recommendation
  - Since February 2004

MANCHESTER 1824

The University of Manchester

The OWL Family Tree

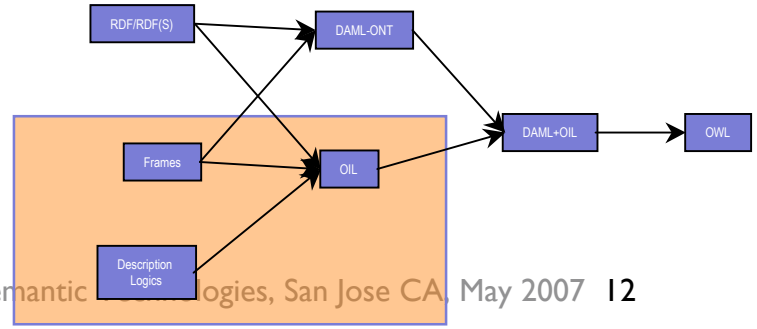


MANCHESTER 1824

The University of Manchester

A Brief History of OWL

- OIL
  - Developed by group of (largely) European researchers (several from EU OntoKnowledge project)
  - Based on frame-based language
  - Strong emphasis on formal rigour.
  - Semantics in terms of Description Logics
  - RDFS based syntax

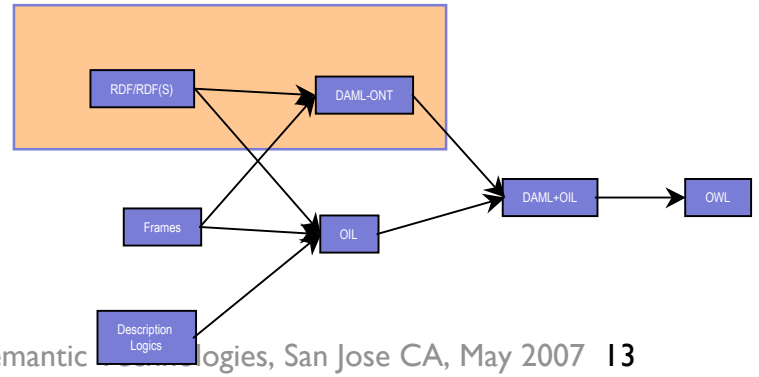


MANCHESTER 1824

The University of Manchester

A Brief History of OWL

- DAML-ONT
  - Developed by DAML Programme.
    - Largely US based researchers
  - Extended RDFS with constructors from OO and frame-based languages
  - Rather weak semantic specification
    - Problems with machine interpretation
    - Problems with human interpretation

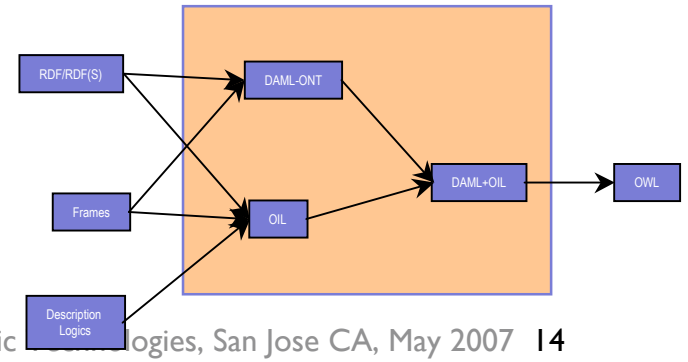


MANCHESTER 1824

The University of Manchester

A Brief History of OWL

- DAML+OIL
  - Merging of DAML-ONT and OIL
  - Basically a DL with an RDFS-based syntax.
  - Development was carried out by "Joint EU/US Committee on Agent Markup Languages"
  - Extends ("DL subset" of) RDF
- DAML+OIL submitted to W3C as basis for standardisation
  - Web-Ontology (WebOnt) Working Group formed

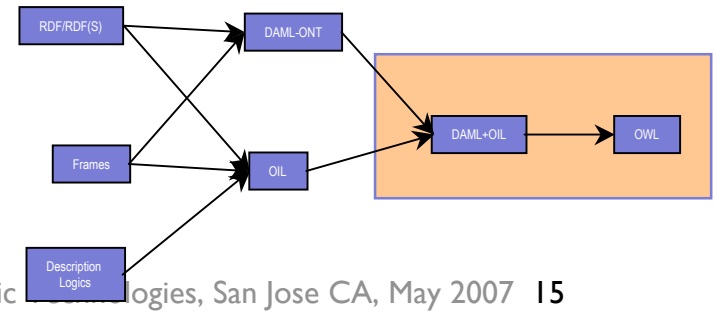


MANCHESTER 1824

The University of Manchester

A Brief History of OWL

- OWL
  - W3C Recommendation (February 2004)
  - Based largely on the DAML+OIL specification from March 2001.
  - Well defined RDF/XML serializations
  - Formal semantics
    - First Order
    - Relationship with RDF
  - Comprehensive test cases for tools/implementations
  - Growing industrial takeup.



MANCHESTER 1824

The University of Manchester

Points from History

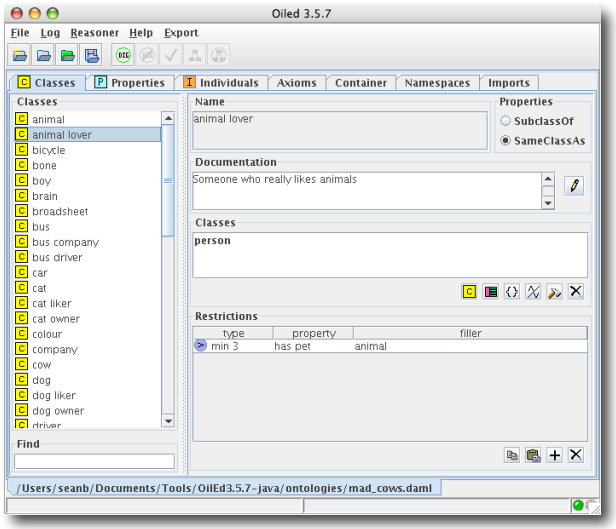
- Influence of frame based modelling approaches
  - Classes, slots, fillers
- Influence of logical foundations
  - Well-formed semantics
  - Inference
- Influence of Web Languages
  - RDF, RDF(S)

MANCHESTER 1824

The University of Manchester

History

- OilEd
  - One of the first ontology editors to employ description logic reasoning
  - Targeted at OIL (then DAML+OIL)
  - ~10,000 registrations for download
  - Used for teaching in both academia and industry
- Data model independent of the underlying concrete syntax
  - But without decent abstractions



MANCHESTER 1824

The University of Manchester

History/Provenance

- WonderWeb
  - http://wonderweb.semanticweb.org/
  - An EU STREP targeted at producing *Ontology Infrastucture for the Semantic Web*
  - Produced the initial implementations of the API
- CO-ODE
  - http://www.co-ode.org/
  - UK JISC funded project to provide support for communities developing OWL
    - Protégé tool development and OWL support
  - Supporting current development.

MANCHESTER 1824

The University of Manchester

OWL

OWL allows us to describe a domain in terms of:

- Individuals
  - Particular objects in our domain
- Classes
  - Collections of objects (usually sharing some common characteristics)
- Properties
  - Binary relationships between individuals.
- Plus a collection of axioms describing how these classes, individuals, properties etc. should be interpreted

Note: This talk will not be discussing whether this is the "right" way to do things….

MANCHESTER 1824

The University of Manchester

OWL

- OWL has a number of operators that allow us to describe the classes and the characteristics that they have
- Boolean operators
  - and, or, not
- Quantification over properties/relationships
  - universal, existential.
- A clear and unambiguous semantics for the operators and composite class expressions

MANCHESTER 1824

The University of Manchester

OWL and RDF

- The normative presentation for OWL is based on RDF
- XML-RDF is one of a number of possible syntactic presentations of OWL.
- Working solely at the RDF triple level can introduce both mechanical and conceptual difficulties.
  - Round tripping. The mapping rules between OWL abstract syntax and triples are not 1:1.
  - Containment. Assertions that statements belong to a particular ontology.
- Alternative syntaxes can be useful
  - OIL's original text format was popular with users – you *can* write XML-RDF in emacs, but I wouldn't recommend it.....

MANCHESTER 1824

The University of Manchester

OWL and RDF

- Sometimes an uneasy relationship between the two
- Syntactic Layering
  - Additional baggage/hoops to jump through
- Semantic Layering
  - Complicated

OWL != RDF

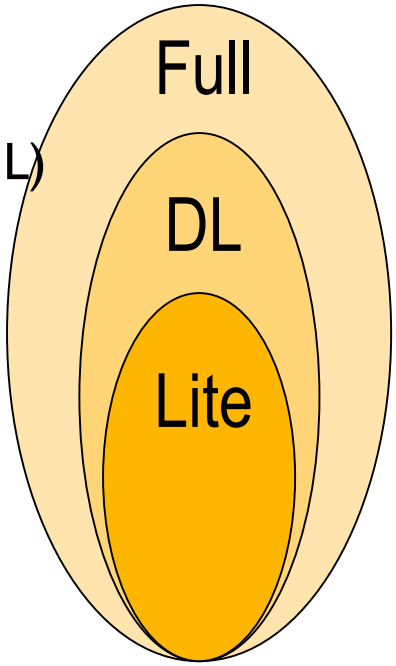
- One of our key motivators was to provide infrastructure that helped insulate applications from these issues

MANCHESTER 1824

The University of Manchester

OWL Layering

- Three species of OWL
  - OWL Full is the union of OWL syntax and RDF
  - OWL DL restricted to FOL fragment (≈ DAML+OIL)
    - Corresponds to *SHOIN*(D_n) Description Logic
  - OWL Lite is "simpler" subset of OWL DL
- Syntactic & Semantic Layering
  - OWL DL semantics = OWL Full semantics (within DL fragment)
  - OWL Lite semantics = OWL DL semantics (within Lite fragment)
- DL semantics are definitive
  - In principle: correspondence proof
  - But: if Full disagrees with DL (in DL fragment), then Full is wrong

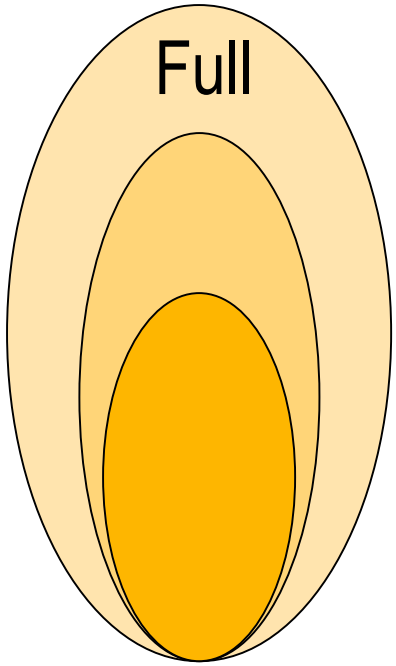


MANCHESTER 1824

The University of Manchester

OWL Full

- No restriction on use of OWL vocabulary (as long as legal RDF)
  - Classes as instances (and much more)
- RDF style model theory
  - Reasoning using FOL engines
    - via axiomatisation
  - Semantics should correspond with OWL DL for suitably restricted KBs

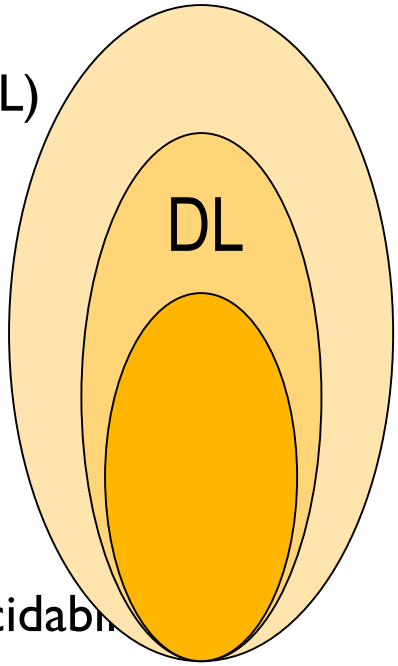


MANCHESTER 1824

The University of Manchester

OWL DL

- Use of OWL vocabulary restricted
  - Can't be used to do "nasty things" (i.e., modify OWL)
  - No classes as instances
  - Defined by abstract syntax + mapping to RDF
- Standard DL/FOL model theory (definitive)
  - Direct correspondence with (first order) logic
- Benefits from underlying logic research
  - Well defined semantics
  - Formal properties well understood (complexity, decidabi
  - Known reasoning algorithms
  - Implemented systems (highly optimised)

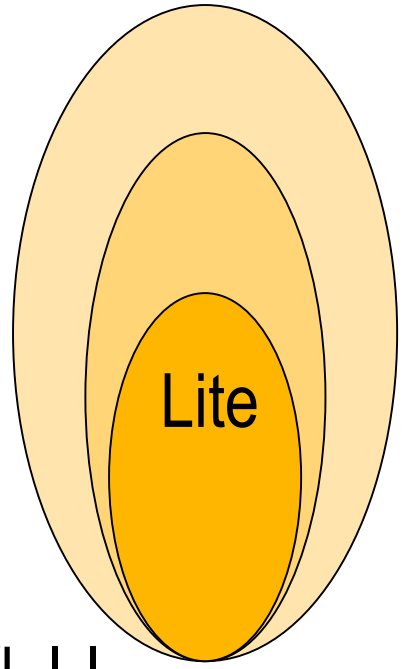


MANCHESTER 1824

The University of Manchester

OWL Lite

- Like DL, but fewer constructs
  - No explicit negation or union
  - Restricted cardinality (zero or one)
  - No nominals (oneOf)
- Semantics as per DL
  - Reasoning via standard DL engines (+datatypes)
    - E.g., FaCT, RACER, Cerebra, Pellet
- In practice, not really used.
  - Alternative "tractable fragments" approach for OWL 1.1



MANCHESTER 1824

The University of Manchester

OWL 1.1

- Some limitations have already been identified in the original OWL specifications
  - Qualified cardinality restrictions
  - Lack of support for richer property axioms (e.g. interaction between partonomies and locations)
  - Metamodelling
  - Annotations
- "OWL 1.1" aims to extend the language in order to address these issues
  - The latest version of the OWL API has support for this additional expressivity
  - W3C Member Submission
  - http://webont.org/owl/1.1/

MANCHESTER 1824

The University of Manchester

Why build an OWL API?

- The use of a higher level data model can help to
  - insulate us from the vagaries of concrete syntax.
  - make it clear what is happening in terms of functionality.
  - increase the likelyhood of interoperating applications.

Experiences

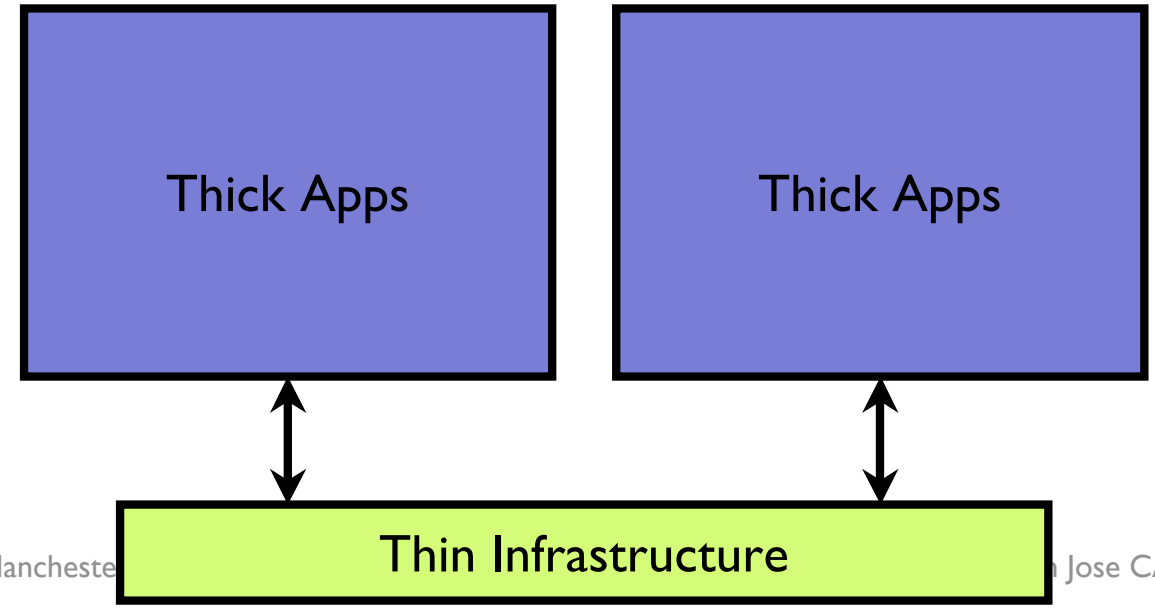
- An early example application built using the API was a web service that performed validation and translation to alternative syntaxes.
  - See *OWL DL: Trees or Triples, WWW2004*
- Ontology level objects made it easy to write code spotting "internal errors"

MANCHESTER 1824

The University of Manchester

How thick is **your** infrastructure?

- A lightweight infrastructure (e.g. RDF) means that clients/apps have to do more. And *may* do it differently.
- Metadata can end up being locked away within the applications where others can't get at it. Is that sharing? Are you exposing the semantics?

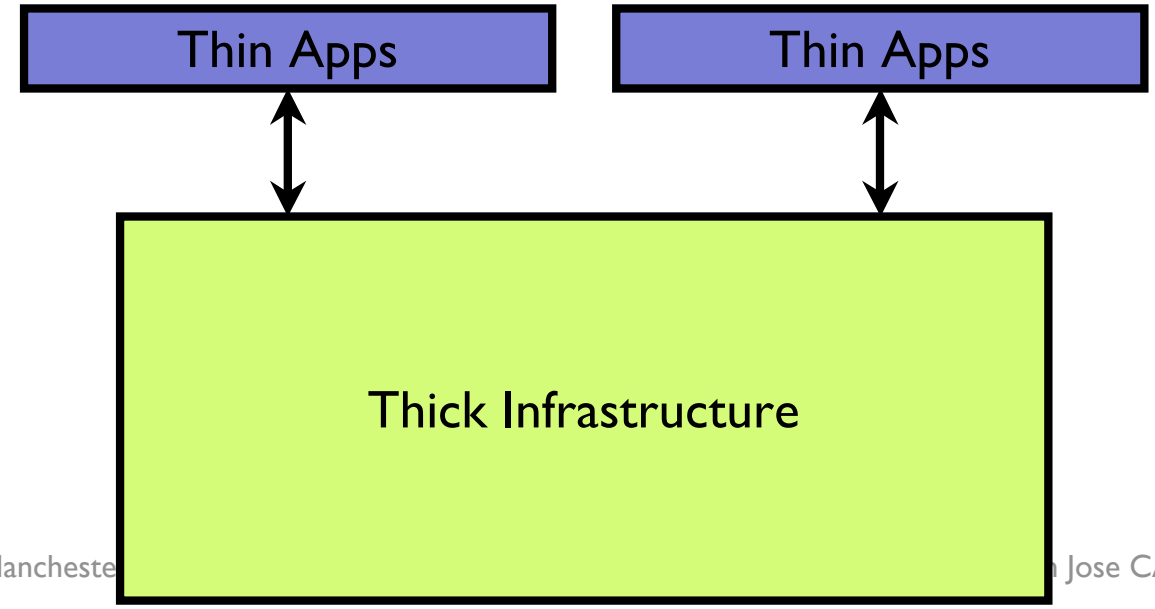


MANCHESTER 1824

The University of Manchester

How thick is **your** infrastructure?

- Sharing is about interoperations. Ensuring that when you look at or process my data, you do it in a consistent way.
- "Thick" infrastructure can help interoperability. Clients don't have to guess how to interpret things.
  - But can be harder to build

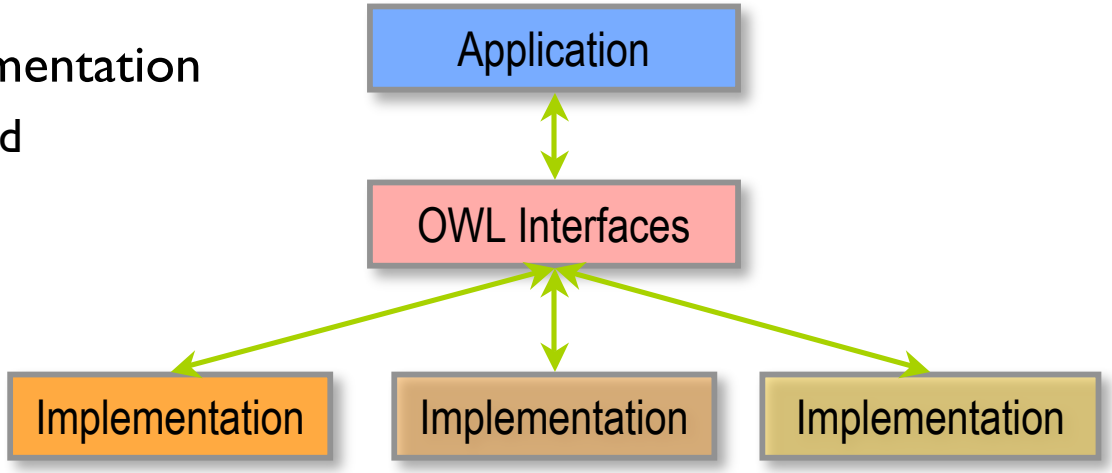


MANCHESTER 1824

The University of Manchester

Assumptions

- Primarily targeted at OWL-DL
  - This does not mean that we cannot handle OWL-Full ontologies, but a number of design decisions reflect this assumption.
- Java based
  - Interfaces
  - Java reference implementation
    - Main memory based

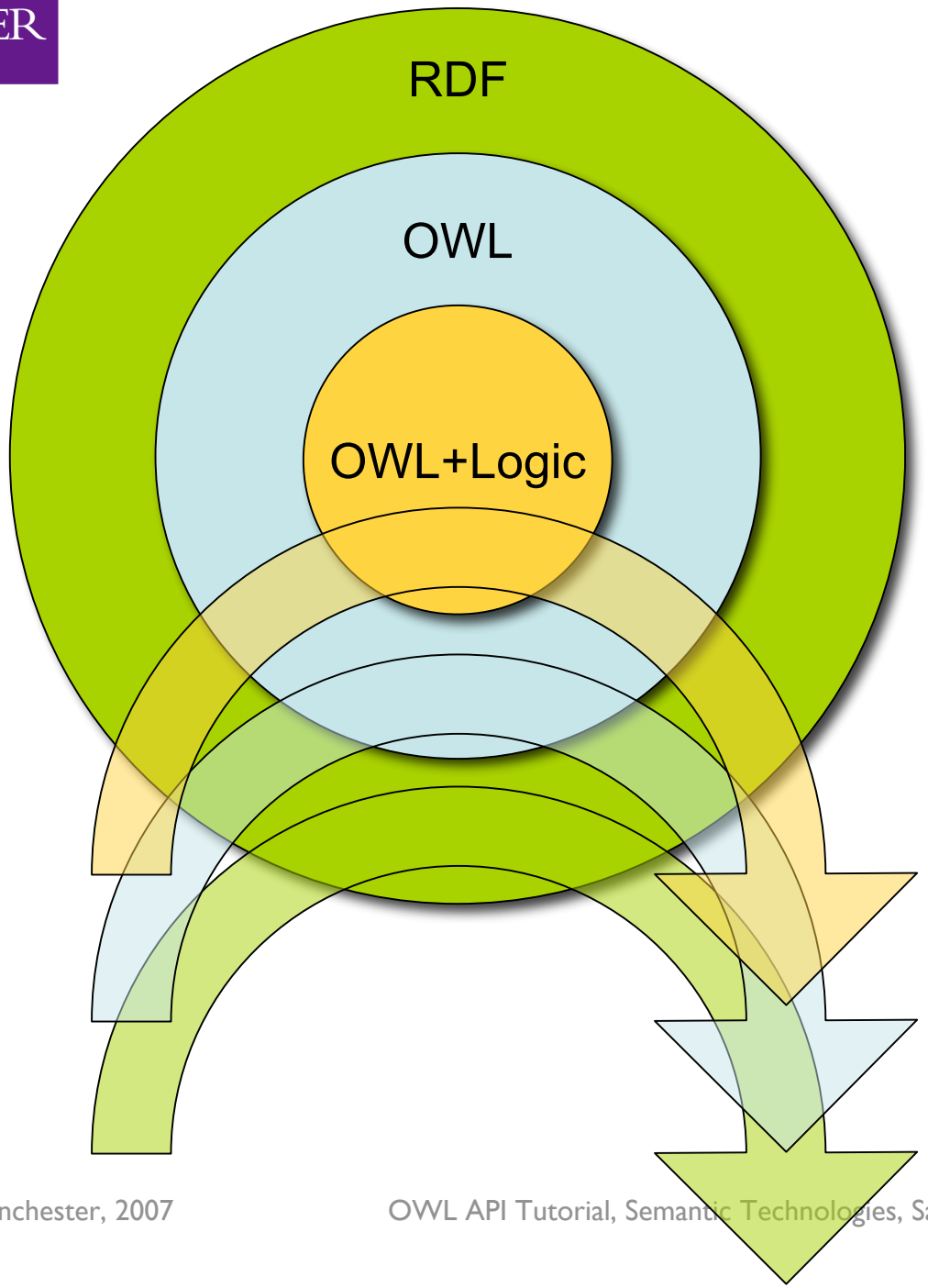


MANCHESTER 1824

The University of Manchester

What *is* an "OWL Implementation"?

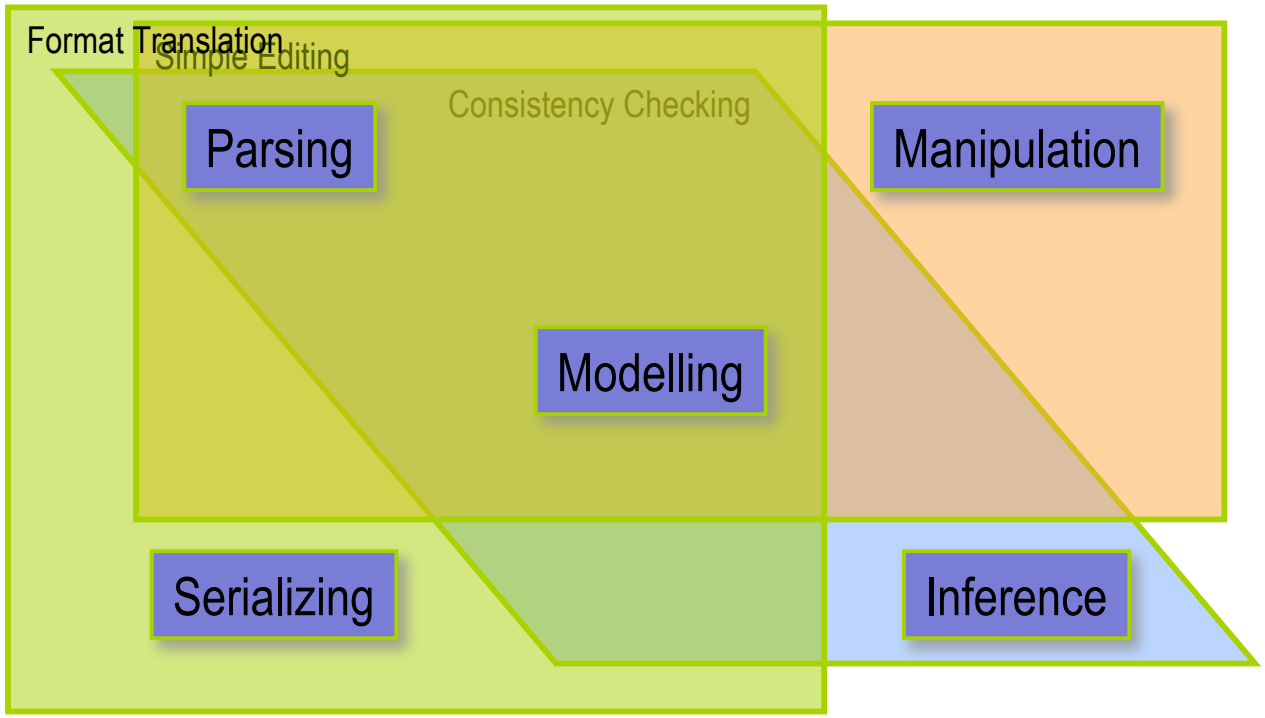
- Modelling
  - Provide data structures that represent OWL ontologies/documents.
- Parsing
  - Taking some syntactic presentation, e.g. OWL-RDF and converting it to some [useful] internal data structure.
- Serializing
  - Producing a syntactic presentation, e.g. OWL-XML from a local data structure.
- Manipulation/Change
  - Being able to manipulate the underlying objects.
- Inference
  - Providing a representation that implements/understands the formal semantics of the language.


RDF
OWL
OWL+Logic

MANCHESTER 1824

The University of Manchester

Implementation Aspects



MANCHESTER 1824

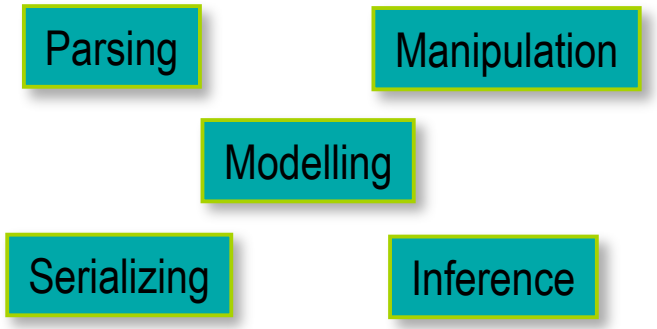
The University of Manchester

OWL Abstract Syntax

- Provides a definition of the language in terms of the constructs and assertions allowed.
- Semantics are then defined in terms of this abstract syntax.
- Our OWL API data model is based largely on this abstract syntax presentation.
  - Conceptually cleaner.
  - Syntax doesn't get in the way

Considerations

- Clear identification of functionalities and a separation of concerns
- Representation
  - Syntax vs. Data Model
  - Interface vs. Implementation
  - Locality of Information
- Parsing/Serialization
  - Insulation from underlying concrete presentations
  - Insulation from triples

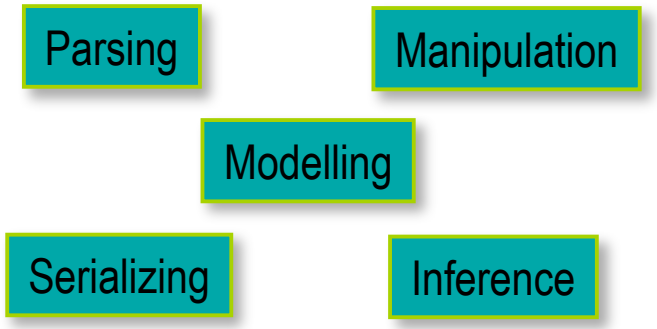


MANCHESTER 1824

The University of Manchester

Considerations

- Manipulation/Change
  - Granularity
  - Dependency
  - User Intention
  - Strategies
- Inference
  - Separation of explicit assertions from inferred consequences
  - External reasoning implementations



MANCHESTER 1824

The University of Manchester

Programming to the OWL API: The API

Sean Bechhofer

University of Manchester

sean.bechhofer@manchester.ac.uk

MANCHESTER 1824

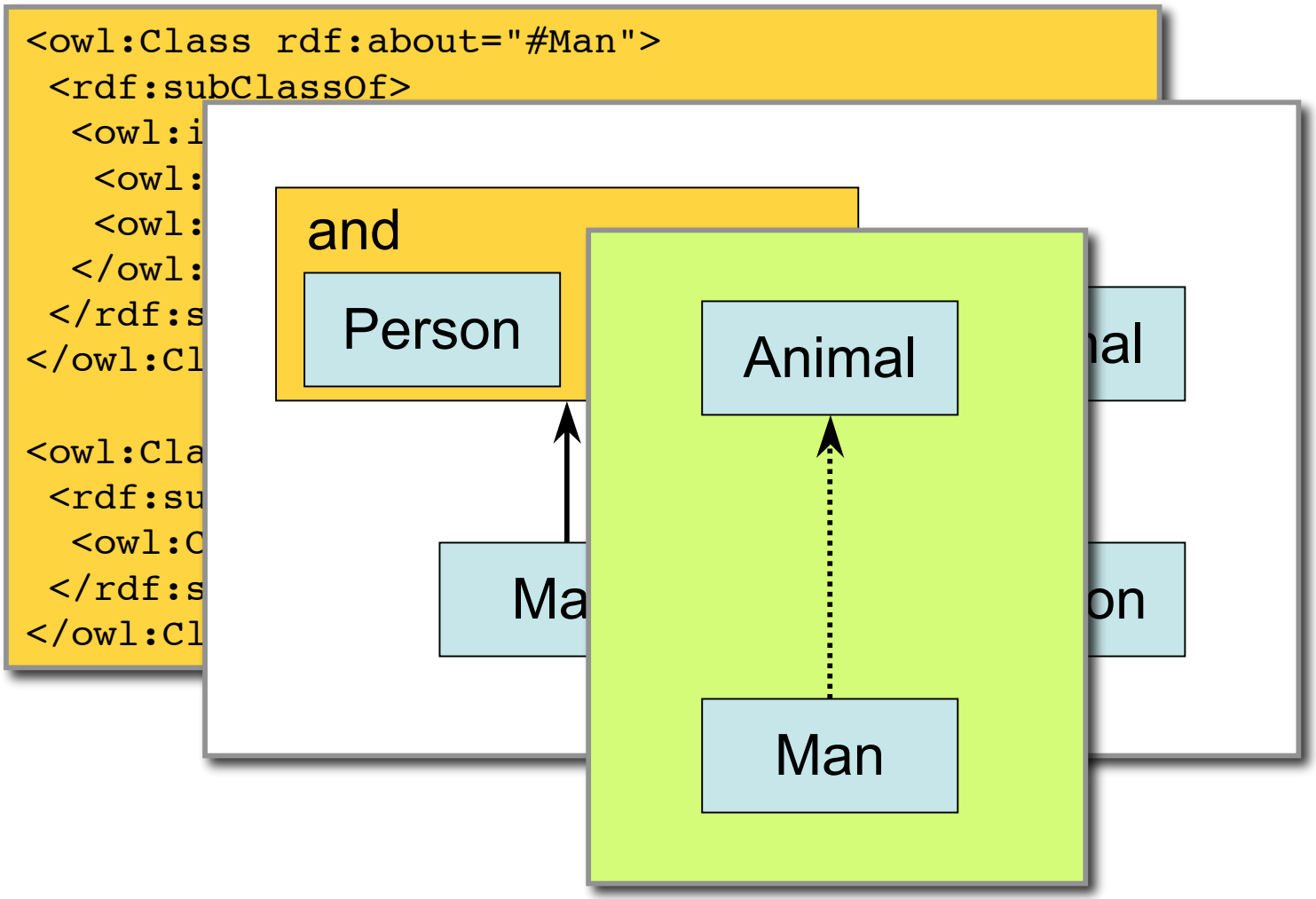
The University of Manchester

What is an Ontology?

- A particular syntactic presentation
- The facts represented by that syntactic presentation
- The information that can be inferred from those facts
  - E.g. consequences of the assertions plus the underlying semantics of the language.

- The difference between these becomes important with languages like OWL.
- What answers will I expect when interrogating the structure?

What is an Ontology?



MANCHESTER 1824

The University of Manchester

Overall Philosophy

- The OWL API is targeted primarily at representing OWL-DL
- An Ontology is represented as a collection of axioms that assert information about the classes, properties and individuals that are in the ontology
  - This is a change from the original approach
  - Fits better with current thinking surrounding OWL 1.1
  - Provides a uniform view on the ontology
- *When is a class or property "in" an Ontology?*
  - This isn't explicitly captured in the OWL specs
  - Somewhat application dependent

MANCHESTER 1824

The University of Manchester

Basic Data Structures

- At its heart, the OWL API provides data structures representing OWL ontologies
- Plus classes to help
  - Create;
  - Manipulate;
  - Parse;
  - Render; and
  - Reason about those structures
- The basic data structure represents the objects in the ontology and corresponds roughly to the abstract syntax of OWL.

MANCHESTER 1824

The University of Manchester

OWLOntology

- An OWLOntology represents an ontology
- It consists of a collection of OWLAxioms
- Note that the OWLOntology object doesn't explicitly contain classes, properties etc.
  - The classes "in" the ontology are those that are referenced by the axioms that are in the ontology
  - We believe this provides a *clean* and *consistent* story as regards the objects that make up the ontology
- The OWLOntology provides an explicit context within which we can assert information about classes and properties.
  - Locality of information

MANCHESTER 1824

The University of Manchester

Names and URIs

- Ontologies in OWL are named using URIs
- Ontologies can also be retrieved from URLs.
- This can cause some confusion!
  - There's no explicit requirement that the location you retrieve the ontology from is the same as the name that it's given.
- The situation isn't made any easier by OWL's import mechanism.

MANCHESTER 1824

The University of Manchester

OWL Import

- OWL provides a mechanism for importing one ontology into another
- The intuition is that the axioms in the imported ontology will be added to the theory defined by the importing ontology
- Note that the recommendation doesn't really say much about how this will be achieved

MANCHESTER 1824

The University of Manchester

Imports

*Importing another ontology brings the entire set of assertions provided by that ontology **into** the current ontology. In order to make best use of this imported ontology it would normally be coordinated with a namespace declaration. Notice the distinction between these two mechanisms. The **namespace** declarations provide a convenient means to reference names defined in other OWL ontologies. Conceptually, owl:imports is provided to indicate your intention to include the assertions of the target ontology. Importing another ontology, O2, will also import all of the ontologies that O2 imports.*

OWL Guide

MANCHESTER 1824

The University of Manchester

Imports

An owl:imports statement references another OWL ontology containing definitions, whose meaning is considered to be part of the meaning of the importing ontology. Each reference consists of a URI specifying from where the ontology is to be imported. Syntactically, owl:imports is a property with the class owl:Ontology as its domain and range.

The owl:imports statements are ***transitive****, that is, if ontology A imports B, and B imports C, then A imports both B and C.*
Importing an ontology into itself is considered a null action, so if ontology A imports B and B imports A, then they are considered to be equivalent.

Note that whether or not an OWL tool must load an imported ontology ***depends on the purpose of the tool****. If the tool is a complete reasoner (including complete consistency checkers) then it must load all of the imported ontologies. Other tools, such as simple editors and incomplete reasoners, may choose to load only some or even none of the imported ontologies.*

OWL Reference

MANCHESTER 1824

The University of Manchester

Imports

*Imports annotations, in effect, are directives to retrieve a Web document and treat it as an OWL ontology. However, most aspects of the Web, including missing, unavailable, and time-varying documents, reside outside the OWL specification; all that is carried here is that a URI can be "**dereferenced**" into an OWL ontology. In several places in this document, therefore, idealizations of this operational meaning for imports are used.*

*…an owl:imports annotation also imports the contents of another OWL ontology into the current ontology. The imported ontology is the one, if any, that has as name the argument of the imports construct. (This treatment of imports is **divorced from Web issues**. The intended use of names for OWL ontologies is to make the name be the location of the ontology on the Web, but this is outside of this formal treatment.)*

OWL Semantics

MANCHESTER 1824

The University of Manchester

Imports

- In essence, an owl:imports statement means:

  Dereference the URI that's given, and make sure that any time you do some reasoning, you take into account the axioms in the imported ontology

- Questions like "which classes are *in* this ontology" can be answered in an application specific way.
  - If Ontology A imports Ontology B and Class X is "in" ontology B is it also "in" Ontology A?
- To complicate matters further, OWL-DL requires that owl:imports only applies to things that are explicitly typed as owl:Ontology.

Logical and Physical

- To get round these problems, we define the notion of logical and physical URIs
  - Logical: the URI used to name the ontology
  - Physical: the location where the ontology was retrieved from
- An OntologyURIMapper is then used to map between physical and logical URIs
  - Allows for local copies or repositories of ontologies
- We are in some way playing "fast and loose" with the spec here, but there is no real practical alternative.

MANCHESTER 1824

The University of Manchester

OWLEntity

- OWLEntity is the fundamental building block of the ontology
  - Classes
  - Properties
  - Individuals
  - Datatypes
- Named using URIs
  - But be aware of punning

MANCHESTER 1824

The University of Manchester

OWLClass

- Represents an OWL Class.
- The Class itself is a relatively lightweight object
  - A Class doesn't hold information about definitions that may apply to it.
- Axioms relating to the class are held by an OWLOntology object
  - E.g. a superclass axiom must be stated within the context of an OWLOntology
  - Thus alternative characterisations/perspectives can be asserted and represented for the same class.
- Axioms do not relate only to a class, but to a class within the context of an ontology

OWLClass

- Methods are available on OWLClass that give access to the information within a particular ontology

```
java.util.Set<OWLDescription> getDisjointClasses(OWLOntology ontology)
java.util.Set<OWLDescription> getEquivalentClasses(OWLOntology ontology)
```

- But these are simply *convenience* methods.

MANCHESTER 1824

The University of Manchester

OWLProperty

- OWL makes a distinction between
  - Object Properties: those that relate two individuals
    - E.g. hasBrother
  - Data Properties: those that relate an individual to a concrete data value
    - E.g. hasName
- There is a strict separation between the two and two explicit classes representing them
  - OWLObjectProperty
  - OWLDataProperty

MANCHESTER 1824

The University of Manchester

OWLProperty

- Properties can have associated domains and ranges
- There is also a property hierarchy
  - Super properties
  - Property equivalences
  - Disjoint Properties (OWL1.1)
- In OWL1.1, we also have the option of using property expressions (e.g. property chains).
- Again, as with classes, the property objects are lightweight and all assertions about properties are made in the context of an Ontology.
  - E.g functional properties

MANCHESTER 1824

The University of Manchester

OWLObjectProperty

- Represents an Object Property that can be used to relate two individuals
- Object properties can have additional characteristics
  - Transitivity
  - Inverses

MANCHESTER 1824

The University of Manchester

OWLDataProperty

- Represents an Object Property that can be used to relate two individuals
- Data properties can also have additional characteristics
  - Functional

MANCHESTER 1824

The University of Manchester

OWLDescription

- OWLDescription represents an OWL class expression
- Atomic classes
- Boolean expressions
  - Intersection (and)
  - Union (or)
  - Complement (not)
- Restrictions
  - Explicit quantification (some, all)
  - Cardinality restrictions (atmost, atleast)

MANCHESTER 1824

The University of Manchester

OWLAxiom

- An ontology contains a collection of OWLAxioms
- Each axiom represents some fact that is explicitly asserted in the ontology
- There are a number of different kinds of axiom
  - Annotation Axioms
  - Declaration Axioms
  - Import Axioms
  - Logical Axioms

MANCHESTER 1824

The University of Manchester

Annotation Axioms

- An OWLAnnotationAxiom is used to associate arbitrary pieces of information with an object in the ontology
  - Labels or natural language strings
  - Dublin core style metadata, e.g. author or creator information
- Annotation Axioms have no logical significance
  - They do not affect the underlying semantics of the ontology

MANCHESTER 1824

The University of Manchester

Declaration Axioms

- An OWLDeclarationAxiom simple declares or introduces an entity into an ontology
- Declaration axioms have no logical significance
- Such axioms can be useful for integrity checks, e.g. tools may require that every entity used in the ontology is explicitly declared
  - This can help detect issues or problems with mis-spelling

MANCHESTER 1824

The University of Manchester

Import Axioms

- An OWLImportsDeclaration annotates an ontology in order to describe the ontologies it imports.
  - In the OWL specification, owl:import is a kind of owl:OntologyProperty which is treated in a special way.
- As discussed earlier, we need to be careful about the way in which we treat ontology import structures.

MANCHESTER 1824

The University of Manchester

Logical Axioms

- The subclasses of OWLLogicalAxiom represent the logical assertions contained in the ontology
  - Supers (of classes and properties)
  - Equivalences (of classes and properties)
  - Property Characteristics
    - Functionality, transitivity etc.
  - Facts about particular individuals
    - Types
    - Relationships
    - Values

MANCHESTER 1824

The University of Manchester

Equality

- Equality on objects in the API is defined structurally.
- In general, two objects are equal if they have the same structure
- E.g. for boolean class descriptions, the operands are compared as a *Set*.
  - If the two sets of operands are equal, the descriptions are equal.
- Note that this is syntactic. Logical equivalences that may follow from axioms or assertions in the ontology do not impact on the equality of the objects.

MANCHESTER 1824

The University of Manchester

Visitors

- The API makes extensive use of the Visitor Pattern
  - *Represents an operation to be performed on the elements of an object structure. Visitor lets you define a new operation without changing the classes of the elements on which it operates*

    Design Patterns (Gamm et al)
- The use of the Visitor pattern allows us to define many different operations over the data structure without "polluting" it with application specifics.
- Visitor is good when the data structure is static, but costly if the structure changes
  - As our data structures are based on the language, the structure is likely to remain static here.

MANCHESTER 1824

The University of Manchester

Visitors

- Visitors are primarily used when rendering or serializing ontologies into different concrete formats.
- There are Visitors for each of the main classes
  - OWLEntityVisitor
  - OWLAxiomVisitor
  - OWLDescriptionVisitor
- Visitors are also used to enact changes to the Ontology
  - OWLOntologyChangeVisitor

MANCHESTER 1824

The University of Manchester

The times they are a changing…

- The data structures provide read-only access to the information held in the ontology
- Changes to the ontology are effected through the use of the Command Pattern.
  - *Encapsulate a request as an object, thereby letting you parameterize clients with different requests, queue or log requests, and support undoable operations*

Design Patterns (Gamma et al)

- Objects explicitly represent the changes
- The change objects are passed to a visitor which then enacts the changes

MANCHESTER 1824

The University of Manchester

Changes as Objects

- Undo operations
- Metadata representing the provenance of the change
- Explicit dependencies between changes
- Representing change sets
  - History
  - Checkpoints
- Use of the Strategy Pattern to enact changes
  - *Define a family of algorithms, encapsulate each one, and make them interchangeable. Strategy lets the algorithm vary independently from clients that use it.*

Design Patterns (Gamma et al)

MANCHESTER 1824

The University of Manchester

Changes

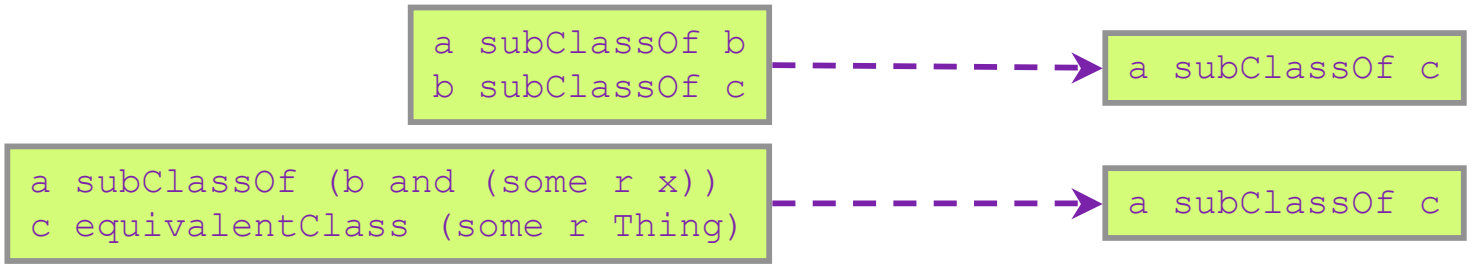
- The API takes an "axiom-centric" view
- There are a limited number of change objects
  - Add an Axiom
  - Remove an Axiom
  - Set the Ontology URI
- Trade off between simplicity and power
  - Change from original API, which had a number of different change objects encapsulating different changes.
  - Change object describes what happened, e.g. add/remove
  - Wrapped axiom describes the change
- Changes then enacted by a Change Visitor.

MANCHESTER 1824

The University of Manchester

Inference

- As we've already discussed, OWL's semantics allows the possibility to perform inference or reasoning over an ontology
- A reasoner may be able to determine additional facts that follow from the information in the ontology
- How best do we expose this information?
  - Add it into the structure?
  - Provide alternative interfaces?

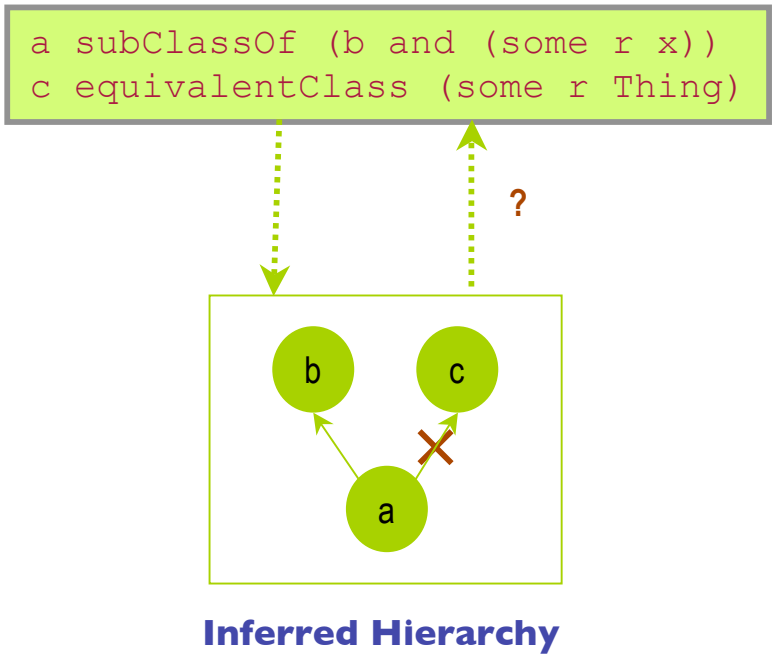


MANCHESTER 1824

The University of Manchester

Inference

- This is particularly important when users get involved
  - What do we do when the user tries to **remove** some inferred information?
- Separating the inferred information from the asserted information doesn't necessarily answer that question
  - But does mean that we have enough to be able to make a choice.
- Offering separate inference interfaces also allows applications to be clear about the services they may (or may not) supply.



MANCHESTER 1824

The University of Manchester

Inference

- The API includes a collection of interfaces that provide access to inference services operating over the model
- OWLClassReasoner
  - Hierarchy information, class satisfiability
- OWLConsistencyChecker
  - Ontology consistency
- OWLIndividualReasoner
  - Reasoning about individuals
- OWLPropertyReasoner
  - Reasoning about properties

MANCHESTER 1824

The University of Manchester

Reasoner Implementations

- Generic DIG Reasoners
  - Via DIG 1.1
- Pellet
  - Pure Java implementation
  - Implements OWL API reasoner interfaces
- FaCT++
  - C++ Implementation
  - Java wrapper
  - OWLAPI wrapper implementating OWL API interfaces
- Debuggers etc. then sit on whichever reasoner you have.
- Syntactic/Told Reasoners
  - Should these be here?

MANCHESTER 1824

The University of Manchester

Putting things together

- So far, what we've seen are a collection of interfaces providing access to the various aspects of functionality that we can associate with OWL ontologies
- How do we bring this all together in order to actually *use* the code and make use of a particular implementation?

MANCHESTER 1824

The University of Manchester

Managing Ontologies

- The model data structures provide representations of the basic building blocks.
- Management and creation of ontologies is controlled by an OWLOntologyManager
  - This replaces OWLConnection/OWLManager in the original implementation
- The Manager is responsible for keeping track of the ontologies and concrete formats for storage of the ontologies.
- Handles Ontology changes

MANCHESTER 1824

The University of Manchester

Manager desiderata

- This is still somewhat experimental.
- Flexibility
  - Allowing loading/storage from a variety of form
- Decoupling of formats
- Mix/match implementations?

MANCHESTER 1824

The University of Manchester

One Ring to Bind Them

- The apibinding module provides a utility class OWLManager that brings together common parsers and renderers along with the in-memory implementation
- We can use this as a point of entry into the implementation.
  - Provides basic infrastructure that takes care of standard
  - If we're doing "standard" things, then the basic OWLManager setup is enough.
  - We can extend this, by, for example adding new concrete syntaxes (along with parsers and renderers that will handle those formats).
    - See examples later

MANCHESTER 1824

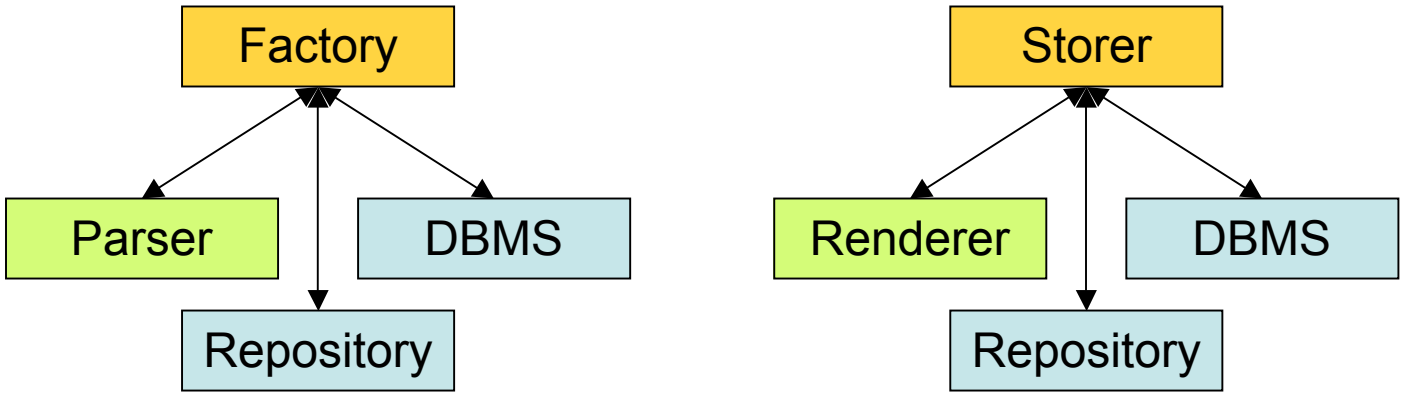
The University of Manchester

URI Mapping

- A URIMapper allows us to maintain a separation between logical URIs and physical URIs
- Useful for
  - Offline working
  - Caching ontologies
  - Solving the name/location problems identified with imports

Factory and Storer

- OWLOntologyFactory and OWLOntologyStorer are used for loading and saving ontologies
- They provide an extra level of abstraction above parsers and serialisers/renderers
  - Allows the potential for retrieving ontologies from data sources other than files/streams, e.g. DBMS/repository/RMI etc.



MANCHESTER 1824

The University of Manchester

Parsing and Rendering

- Parsers and Renderers allow creation of ontologies from concrete serialisations and storage of ontologies using those serialisations.
- A Parser returns information about the format that the ontology was parsed in.

MANCHESTER 1824

The University of Manchester

Loading an Ontology

- Map the logical URI to a physical URI
- Query available Factories for one that can handle the URI
- For the basic factories in our implementation, the selected Factory will then use a parser to parse the given URI
  - Selection of parsers could be via MIME-types or extension mappings
  - Currently simple approach
  - Could do something other than parsing….

MANCHESTER 1824

The University of Manchester

Ontology Formats

- The OWLOntologyFormat class represents a format used for concrete serialisation
  - E.g OWL RDF/XML
- The format may also contain information about the particular serialisation
  - E.g. namespace declarations
  - Ordering
  - Structural information
  - Helps in addressing problems with round-tripping
- If an ontology was parsed, the Manager maintains information about the original format of the ontology

Manager

- The default Manager provides support for a number of basic formats
- Parsing:
  - RDF/XML; OWL XML; KRSS; OBO; Functional Syntax
  - ParserFactoryRegistry knows about available parsers.
- Rendering
  - RDF/XML; OWL XML; Functional Syntax
  - Most renderers effectively involve writing a Visitor that walks the data structure
  - See later example
- You *could* do it all yourself with bespoke parser factories etc….

MANCHESTER 1824

The University of Manchester

Recap

- Basic interfaces providing data structures
  - Visitors
- Manipulation via Change objects
- Inference via parallel interfaces
- I/O through Factory and Storer classes, wrapping Parsers and Renderers
- Manager object pulling together standard components.

MANCHESTER 1824

The University of Manchester

Programming to the OWL API: Examples

Sean Bechhofer

University of Manchester

sean.bechhofer@manchester.ac.uk

MANCHESTER 1824

The University of Manchester

Nuts and Bolts

- OWL API code is available from sourceforge:

  http://sourceforge.net/projects/owlapi

- Latest versions are in the SVN repository.
- There are two versions of the API
  - The latest version is updated to use features like generics and support extensions to OWL.
  - We are focusing largely on the updated version
  - The basic philosophy in both is the same

MANCHESTER 1824

The University of Manchester

Modules

- The distribution is split into a number of different modules
  - Each module has some identified functionality.
  - There may be dependencies between modules.
- The distribution uses Apache's maven to control the build process
  - Known to work with maven version 2.0.5

MANCHESTER 1824

The University of Manchester

Modules

- api
  - Core interfaces covering ontology data structures, parsing and rendering, inference etc.
- impl
  - Basic "reference" implementation
- xxxparser
  - Parser for concrete syntax xxx (e.g. rdfxml, krss, owlxml, obo etc)
- xxxrenderer
  - Renderer for concrete syntax xxx (e.g. rdfxml, krss, owlxml etc)
- debugging
  - Inference debugger

MANCHESTER 1824

The University of Manchester

Modules

- dig1_1
  - Classes supporting interaction with DIG reasoners
- rdfapi
  - RDF parser (from KAON)
- apibinding
  - Provides a point of convenience for creating an OWLOntologyManager with commonly required features (e.g. rdfxmlparser).

MANCHESTER 1824

The University of Manchester

Examples

- The following examples serve to illustrate various aspects of the API and how we might use it.
  1. Producing a basic hierarchy
  2. Adding Closure Axioms
  3. Rendering Alternative Concrete Syntaxes
  4. Black Box Debugging
- We won't go into *all* the details of how these are done
  - Source for the examples will be available on line.

MANCHESTER 1824

The University of Manchester

1. Hierarchy Example

- Our first example will read an ontology, use a reasoner to calculate the inferred subsumption hierarchy, and then display this hierarchy.

MANCHESTER 1824

The University of Manchester

Details

1. Creating/reading ontologies
2. Basic reasoner interfaces
3. Instantiating a reasoner
4. Printing out the results

- Uses the basic apibinding
  - Standard parsers.
- Interrogation of the data structures
  - Visitor for simple display of entities.

MANCHESTER 1824

The University of Manchester

2. Closure Axioms

- OWL allows us to provide definitions of classes using composite expressions to assert necessary and sufficient conditions.
- However, it's not always easy to capture exactly what we want.
- Closure axioms are an example where additional tool support can help to achieve the "right" model.

MANCHESTER 1824

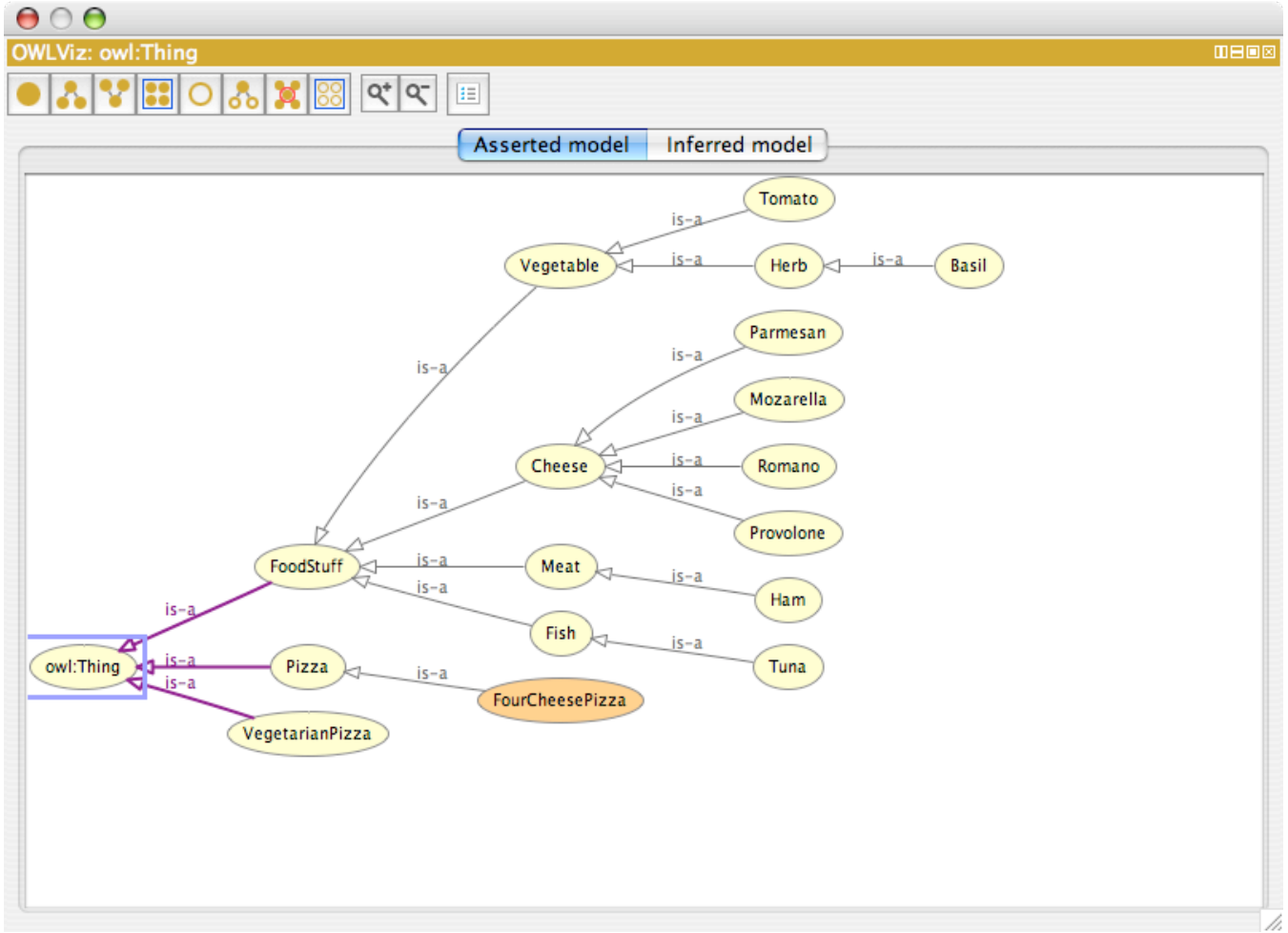
The University of Manchester

Pizzas

- Tomato is a Vegetable
- Basil is a Herb (which is a Vegetable)
- Cheese, Vegetables and Herbs are disjoint from Meat and Fish
- Mozarella, Provolone, Romano and Parmesan are all kinds of Cheese.
- A FourCheesePizza has Tomato, Mozarella, Provolone, Romano and Parmesan and Basil.
- A VegetarianPizza is one that has no Meat or Fish toppings
- Is a FourCheesePizza a VegetarianPizza?

The University of Manchester

An example Model

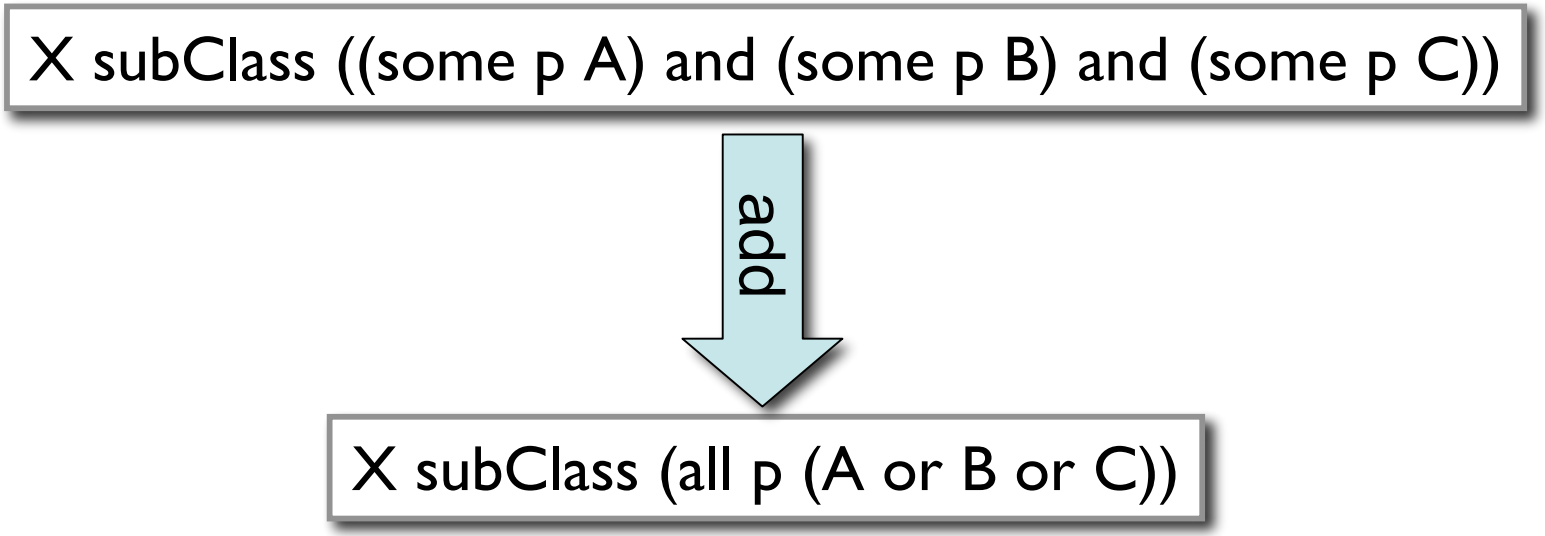


Pizzas

- The problem is that, although we have given conditions that describe what toppings a FourCheesePizza must have, the possible world semantics that OWL has means that we can't preclude situations where a FourCheesePizza has other things on it.

Closure Axioms

- A standard pattern to deal with this situation is to add a "Closure axiom" that states that, in addition to the existential restrictions, there is a universal restriction stating that the only possible fillers are those named.



MANCHESTER 1824

The University of Manchester

Adding Closure axioms

1. Find the axioms that define C
2. Filter subclass axioms
3. Pull out any existential restrictions
4. Create a new axiom with a universal restriction and a filler formed by the union of the definitions
5. Add the axiom to the ontology

- We can do this via a combination of
  - OWLOntology methods
  - Specialised visitor to collect information about axioms
  - Change objects

Working with axiom objects

MANCHESTER 1824

The University of Manchester

3. Concrete Syntax

- There are a variety of concrete syntaxes available for OWL
  - XML/RDF, XML Schema, Concrete Abstract Syntax
- Here we'll show how we can add a serialisers for a new syntax.

MANCHESTER 1824

The University of Manchester

Details

1. Defining renderer for objects
2. Defining renderer for ontology
3. Plugging it into the manager infrastructure

- Definition of some Visitors
- Format object and associated storage helpers
- Registration with the manager.

MANCHESTER 1824

The University of Manchester

4. Black Box Debugging

- Constructing OWL ontologies can be a difficult task.
- Unexpected inferences may be derived
- Debugging can help to track down the reasons for any subsumptions or inconcsistencies.
- The API includes interfaces to support debugging of ontologies
  - Based on work from Maryland's Mindswap Lab.
- We can use this to provide an explanation service

MANCHESTER 1824

The University of Manchester

Details

1. Read ontology
2. Determine Unsatisfiable Classes
3. Provide explanation for the unsatisfiability
   - Minimal sets of axioms causing unsatisfiability
   - Possibly multiple sets
4. Report sets of axioms
   - Rendering again

MANCHESTER 1824

The University of Manchester

Protégé and the API

- Protégé is an ontology editor.
  - One of the most used ontology editing tools (for OWL at least)
- Originally built on top of a frame-based model
- OWL support added through a layered approach.
  - Somewhat problematic
- A reimplementation (Protégé 4) is targeted specifically at OWL and is built on top of the OWL API

MANCHESTER 1824

The University of Manchester

Protégé and the API

- Load/Store Abstractions
  - Allows much cleaner client code.
  - In the past, the tool had to worry about a lot of the details regarding input/output formats
    - E.g. what do you do when the user hits "Save".
- Change
  - Recording history/checkpoints
  - Tracking ontology changes
  - Undo!
  - Event notification
- Can still support a frame-based presentation on top of the axiom-centric model.

MANCHESTER 1824

The University of Manchester

SWOOP and the API

- SWOOP is an OWL editor, originally from the University of Maryland.
- It's a "hypermedia inspired" lightweight editor for OWL ontologies
  - Web browser look & feel
  - Inline Editing
  - Integration with a reasoner
- SWOOP uses the OWL API to represent OWL Models.

MANCHESTER 1824

The University of Manchester

SWOOP and the API

"…Swoop uses the OWL API to model ontologies and their associated entities, benefiting from its extensive and clean support for changes. The OWL API separates the representation of changes from the application of changes. Each possible change type has a corresponding Java class in the API, which is subsequently applied to the ontology (essentially, the Command design pattern). These classes allow for the rich representation of changes, including *metadata* about the changes."

MANCHESTER 1824

The University of Manchester

Programming to the OWL API: Wrap Up

Sean Bechhofer

University of Manchester

sean.bechhofer@manchester.ac.uk

MANCHESTER 1824

The University of Manchester

The OWL API

- OWL is a language for representing ontologies.
- Although it has a relationship with other representations like RDF, it has particular characteristics
- An API allows tools to work at an appropriate level of abstraction
  - Working with the language objects
  - Hiding issues of concrete syntax and representation
- Provides clear identification of functionalities
  - Particularly important when we have inference

MANCHESTER 1824

The University of Manchester

Highlights

- Basic data structures representing OWL Ontologies
  - Influenced by the logical perspective
  - Targeted at a particular level of expressivity
    - OWL DL/OWL 1.1
- Parsing & Rendering
  - Support for standard formats (e.g.XML/RDF)
- Manipulation
  - Through explicit change objects
- Inference
  - Explicit Reasoning interfaces
  - Communication with reasoners through “standard” protocols

MANCHESTER 1824

The University of Manchester

Where's it used?

- Pellet
  - OWL reasoner
- SWOOP
  - OWL editor
- Protégé 4
  - OWL editor
- ComparaGrid
- CLEF
- OntoTrack
  - OWL Editor
- DIP Reasoner
- BT
  - SNOMED-CT support
- BioPAX
  - Lisp bindings (!!)

Caveats

- Primarily designed to support manipulation of T-Box/schema level ontologies
  - Large amounts of instance data may cause problems.
- Designed to support OWL (not RDF)
- This isn't industrial production level quality code
  - It's not bad though :-)
- We can't promise to answer all your questions
- We can't promise to fix all your bugs
- But we'll try……

MANCHESTER 1824

The University of Manchester

The Future

- The API continues to be developed, primarily to support the development of Protégé and OWL support within Protégé.
- Additional Components to support UIs and general manipulation
  - Migrating Protégé code into the code base
- Handling Change History
- Alternative implementations
  - DB backed storage?
- Query Interfaces
  - Conjunctive Query Style?

MANCHESTER 1824

The University of Manchester

Other, Related Work

- Jena
  - Provides OWL Ontology interfaces, layered on the RDF structures of Jena
- Protégé API
  - Protégé 3 provided OWL API layered on Protégé model
  - Mixture of frames, RDF and OWL
  - Evolution to support a UI
- KAON2
  - Support for OWL
  - Not open source

MANCHESTER 1824

The University of Manchester

References

- *Cooking the Semantic Web with the OWL API*, ISWC2003
- *Parsing OWL DL: Trees or Triples?*, WWW2004
- *Patching Syntax in OWL Ontologies*, ISWC 2004
- *The Manchester OWL Syntax*, OWLEd 2006
- *Igniting the OWL 1.1 Touch Paper: The OWL API*, OWLEd 2007

MANCHESTER 1824

The University of Manchester

Thanks!

- Thank you for listening
- Thanks also to everybody who has contributed to the API design and codebase
  - Matthew Horridge
  - Phil Lord
  - Angus Roberts
  - Raphael Volz
  - Olaf Noppens
  - Evren Sirin
  - Any others I forgot to mention (sorry!)…

MANCHESTER 1824

The University of Manchester

More Information

http://sourceforge.net/projects/owlapi
http://owlapi.sourceforge.net/
http://www.co-ode.org/
http://owl.cs.manchester.ac.uk/2007/05/api/