OWL Distilled

Michael Uschold & Dave McComb
Semantic Arts

Semantic Technology for Intelligence, Defense & Security
November 18th, 2015,
Fairfax, VA

Who Are We

Dave McComb:
• President and Founder of Semantic Arts
• Co-founded the SemTech Industry Conference Series
• Broad and deep expertise in both semantics and enterprise architecture

Michael Uschold
• Senior Ontology Consultant, Semantic Arts
• 25 years experience as ontologist and trainer
• Built commercial ontologies in numerous industries
What is Different?

- Gentle introduction to OWL with minimal technical experience required
- We motivate everything, not just a list of constructs to learn
- We focus on the small percentage of OWL that gets used all the time
- Examples are drawn from commercial ontology work across several industries, toy problems

Who You Are

A few introductions:

- Your name & Affiliation
- Your experience with ontologies & OWL
- What are your goals for this tutorial?
Outline

I   What we need to say?
II  How to say it in OWL?
III Patterns and Pitfalls

Lots of class participation, adjusted to how the timing goes.

Part 1: Getting Started
What we need to say
Let's Jump Right In!

- An “ontology” is a model that represents some subject matter that you care about
- It communicates:
  - What kinds of things there are
  - How they are related to each other
  - In a way that supports automated reasoning
- What subject matter interests you?
- Let's consider healthcare for now

In the beginning, there are Things

- What things matter in health care?
  Shout them out…
- Then ask which ones are most fundamental to this subject?
  - Important and central, vs. minor details
  - Relate to many other things in health care
Things in Finance Industry

- What things matter in the **finance industry**?
  
  *Maybe just consider Asset Management*

- Then ask which ones are most fundamental to this subject?
  
  - Important and central, vs. minor details
  
  - Relate to many other things in finance

---

Corporate Registration

- What things matter in the **area of registration and filings for Corporations**?

- Has anyone started a company?
  
  *You will know something about this area.*

- Which ones are most fundamental to this subject?
  
  - Important and central, vs. minor details
  
  - Relate to many other things in corporate registration and filings
Things in Your Favorite Subject

• Tell us YOUR favorite subject or industry…
• What things matter in your favorite subject or industry?
• Then ask which ones are most fundamental to this subject?
  • Important and central, vs. minor details
  • Relate to many other things in finance

<table>
<thead>
<tr>
<th>Healthcare</th>
<th>Finance</th>
<th>Corporation Registration</th>
<th>Your Favorite Subject</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disease</td>
<td>Stock</td>
<td>Corporation</td>
<td></td>
</tr>
<tr>
<td>Diagnosis</td>
<td>Bond</td>
<td>Shares of stock</td>
<td></td>
</tr>
<tr>
<td>Health insurance policy</td>
<td>Broker</td>
<td>Articles of incorporation</td>
<td></td>
</tr>
<tr>
<td>Regulations</td>
<td>Asset</td>
<td>Director / CEO</td>
<td></td>
</tr>
<tr>
<td>Medicine</td>
<td>NYSE</td>
<td>Jurisdiction</td>
<td></td>
</tr>
<tr>
<td>Hospital</td>
<td>S&amp;P 500</td>
<td>Statute</td>
<td></td>
</tr>
<tr>
<td>Doctor</td>
<td>Legal Entity</td>
<td>Registered Address</td>
<td></td>
</tr>
<tr>
<td>Nurse</td>
<td>Trade</td>
<td>Registered Agent</td>
<td></td>
</tr>
<tr>
<td>Patient</td>
<td>Account balance</td>
<td>Annual report</td>
<td></td>
</tr>
<tr>
<td>Appointment</td>
<td>Fees</td>
<td>Registration fee</td>
<td></td>
</tr>
<tr>
<td>Patient visit</td>
<td>Risk</td>
<td>Official &amp; DBA name</td>
<td></td>
</tr>
<tr>
<td>Credentials</td>
<td>Money</td>
<td>Person</td>
<td></td>
</tr>
</tbody>
</table>
Kinds of Things vs. Individual Things

• Then we notice a distinction between:
  • Kinds of things
  • Individual things (of a certain kind)
    Also called ‘instances’

No Thing is an Island

• Knowing what things there are in a subject is just a start
• Things are not important in their own right, what matters is their relationships with other things
• What makes something central, is that it is related in important ways to many other things.
• The true understanding of a subject is understanding inter-relationships among things.
Inter-Related things in Healthcare

- What are the central things and their relationships in health care?
- Think. Draw them here, or shout them out.
Inter-Related things in Healthcare

Inter-Related things in Finance

• What are the central things and their relationships in finance?
• Think. Draw them here, or shout them out.
• Let’s just consider Asset Management
Corporate Registration in ‘OWL’

Inter-Related things in Your Favorite Topic
Things have a variety of attributes

As we have seen…
• We can say a lot about something by saying what other things it is related to.
• A doctor may be related to a patient by delivering care to them.
• A particular person will be related to a particular trade by being the seller or buyer.

But some things we want to say are a bit different. We want to express:
• Jill’s age, first & last name, and date of birth
• Google’s official name and date of incorporation

• Individuals are linked to “literals” rather than to other individuals
The Story So Far

• There are kinds of things
• There are individual things, of a certain kind
• There are relationships between things
• Things have attributes that link things to literals rather than to other things

NEXT: Generalization and Specialization

Generalization and Specialization

• There are more general things and more specialized things.
• The most common example of this is the hierarchy of different life forms:
  • A human being is a special kind of primate
  • A primate is a special kind of mammal
  • A tree is a special kind of plant
  • Etc.
• “a kind of” relationship between two Kinds
Finding “a kind of” relationships

- Sometimes you have various kinds, and notice that one is a specialization of another.
  - A Patient is a kind of ??
- Sometimes, you have one kind of thing, and you want to add more special kinds.
  - Doctor: shout them out
  - Asset: shout them out
- Sometimes you have a two or more kinds of thing and you realize they are both examples of a more general kind.
  - E.g. Doctor and Nurse. What generalizes both?
  - E.g. Person and Corporation. What generalizes both?

Extremely important in the enterprise

Find/Make “a kind of” relationships

<table>
<thead>
<tr>
<th>Healthcare</th>
<th>Finance</th>
<th>Corporation Registration</th>
<th>Your Favorite Subject</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disease</td>
<td>Stock</td>
<td>Corporation</td>
<td></td>
</tr>
<tr>
<td>Diagnosis</td>
<td>Bond</td>
<td>Shares of stock</td>
<td></td>
</tr>
<tr>
<td>Health insurance policy</td>
<td>Broker</td>
<td>Articles of incorporation</td>
<td></td>
</tr>
<tr>
<td>Regulations</td>
<td>Asset</td>
<td>Director / CEO</td>
<td></td>
</tr>
<tr>
<td>Medicine</td>
<td>NYSE</td>
<td>Jurisdiction</td>
<td></td>
</tr>
<tr>
<td>Hospital</td>
<td>S&amp;P 500</td>
<td>Statute</td>
<td></td>
</tr>
<tr>
<td>Doctor</td>
<td>Legal Entity</td>
<td>Registered Address</td>
<td></td>
</tr>
<tr>
<td>Nurse</td>
<td>Trade</td>
<td>Registered Agent</td>
<td></td>
</tr>
<tr>
<td>Patient</td>
<td>Account balance</td>
<td>Annual report</td>
<td></td>
</tr>
<tr>
<td>Appointment</td>
<td>Fees</td>
<td>Registration fee</td>
<td></td>
</tr>
<tr>
<td>Patient visit</td>
<td>Risk</td>
<td>Official &amp; DBA name</td>
<td></td>
</tr>
<tr>
<td>Credentials</td>
<td>Money</td>
<td>Person</td>
<td></td>
</tr>
</tbody>
</table>

Find some “A Kind Of” relationships between these “Kinds” of things.
Identify some important new Kinds that are more general or more special than the ones here.
Some “A Kind Of” Relationships

Drawing Conclusions

- Too often computers are just plain stupid; they don’t know the most obvious things.
- Anyone got a favorite story to share?
- You select M for gender on an online health form, and then it asks if you are pregnant!
- Actually computers are starting to be a bit smarter about this.
- BUT: usually hard-coded rules which are brittle
Drawing Conclusions

- We want a more general way to encode 'knowledge' in the computer so that it is not so dumb.
- Armed with some knowledge and some general rules to apply and a general purpose reasoner, the computer can draw useful conclusions that common sense tells us is so.
- Can you think of some examples of useful conclusions that you would want in the subject areas we have been discussing?

Two Kinds of Conclusions to Draw

NOTE: Not need hard-coded rules.

Very simple, and Very Important
Benefits of Drawing Conclusions

1. Conclude new information, so the computer not seem so dumb, might even seem smart!
   1. fill in the blanks that are obvious to anyone
   2. fill in the blanks that logically follow, but are not necessarily that obvious

2. Detect and explain logical inconsistencies

Logical Inconsistencies

• For example: suppose someone accidentally says that some individual is both a Corporation and a Person
Logical Inconsistencies

- Everyone knows that something cannot be BOTH a warm-blooded physical entity and also a non-physical social object.
- The computer should be able to know this too and cry foul if someone tries it.

Non-Overlapping Kinds

- Some kinds of things are not overlapping; You cannot have one individual be of both kinds
- What are some other examples?

<table>
<thead>
<tr>
<th>Healthcare</th>
<th>Finance</th>
<th>Corporation Registration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disease</td>
<td>Stock</td>
<td>Corporation</td>
</tr>
<tr>
<td>Diagnosis</td>
<td>Bond</td>
<td>Shares of stock</td>
</tr>
<tr>
<td>Health insurance policy</td>
<td>Broker</td>
<td>Articles of incorporation</td>
</tr>
<tr>
<td>Regulations</td>
<td>Asset</td>
<td>Director / CEO</td>
</tr>
<tr>
<td>Medication</td>
<td>MSE</td>
<td>Jurisdiction</td>
</tr>
<tr>
<td>Hospital</td>
<td>MOF</td>
<td>Stanton</td>
</tr>
<tr>
<td>Doctor</td>
<td>Legal</td>
<td>Registered Address</td>
</tr>
<tr>
<td>Nurse</td>
<td>Trusts</td>
<td>Registered Agent</td>
</tr>
<tr>
<td>Patient</td>
<td>Account balance</td>
<td>Annual report</td>
</tr>
<tr>
<td>Appointment</td>
<td>Fees</td>
<td>Registration fee</td>
</tr>
<tr>
<td>Patient visit</td>
<td>Risk</td>
<td>Official &amp; DEA name</td>
</tr>
<tr>
<td>Creditcard</td>
<td>Money</td>
<td>Person</td>
</tr>
</tbody>
</table>

NOTE: You might disagree! All the more important to be explicit and leave as little as possible to guesswork.
Recall

• An “ontology” is a model that represents some subject matter that you care about
• It communicates:
  • What kinds of things there are
  • How they are related to each other
  • In a way that supports automated reasoning
• We have just looked at a few subject areas and how things are related to each other.
• That’s what building an ontology is all about
• We also identified the kinds of things that we must say to describe a subject.

To describe a subject we need to say:

• There are individual things
• There are kinds of things (and that some do not overlap)
• Individuals are of a certain kind
• There are more special and more general kinds of things
• There are relationships between things
• Things have attributes (links to literals)
• How to draw conclusions
  • To add new information
  • To help detect and debug inconsistencies
Then What?

- So you have your ontology describing some subject matter.
- Good for communicating meaning to others
- Most of the time you will also want to use it as the basis for representing and storing data.

Ontology as a Vocabulary for Data

- The ontology provides a vocabulary for creating individuals and making statements about how they are related to other individuals and to literals.
  - John Doe received care from Doctor Jill
  - John Doe received care from Nurse Jane
  - John Doe’s patient visit was on date 12 Jan 2013
Preview: Metadata vs. Data

- **Metadata**: the Ontology which defines the subject matter vocabulary
  - Doctor, Nurse, PatientVisit date, received care from
- **Data**: Use that vocabulary to say things about specific individuals.
- **OWL** is used for both aspects

Next: How to say things in OWL.
Part 2: How to Say it in OWL?
Simple Ontology in Healthcare

Saying Things: Assertions

• In English we talk about assertions:
  Sentences saying that something is so
• Basic sentence structure:
  Subject – Predicate – Object
  • John Doe received care from Doctor Jill
  • Jill Smith’s age is 32
  • John Doe’s patient visit was on date 12Jan2013

• An Ontology is a set of Assertions
• Each Assertion is represented as a Triple
An Ontology is a Set of Triples

- **Subject**: that which something is said about e.g. Jill Smith
- **Predicate**: the kind of thing being said, a way two things can be related e.g. careprovider, age
- **Object**: the individual or literal that is linked to the Subject (via the predicate) e.g. 32

Assertion Represented as a Triple

- `odl:_DrJillSmith` `odl:age` “32”

Two Kinds of Triple

- An Individual is linked to another individual e.g. A particular `patient visit` is linked to the `person` receiving care

- An Individual is linked to a literal value e.g. A particular `person` has a `first name`

- These are examples of Data in a given subject.
- Next: Metadata which describes the subject
Metadata Triples

- Linking an individual to its type
- Linking a kind of thing to its more general kind.
- Linking a kind of thing to its OWL construct
- Linking a kind of relationship to its OWL construct

<table>
<thead>
<tr>
<th>Informal Terminology</th>
<th>Formal OWL Terminology</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build a model of some subject matter</td>
<td>Build an ontology</td>
</tr>
<tr>
<td>Thing: used very loosely to mean any concept or notion</td>
<td>Thing will refer to its OWL interpretation, the most general Kind.</td>
</tr>
<tr>
<td>Things</td>
<td>OWL Individuals</td>
</tr>
<tr>
<td>Kinds of things</td>
<td>OWL Classes</td>
</tr>
<tr>
<td>More general/special kinds of things</td>
<td>Sub Classes and Super Classes</td>
</tr>
</tbody>
</table>
From Informal to Formal

<table>
<thead>
<tr>
<th>Informal Terminology</th>
<th>Formal OWL Terminology</th>
</tr>
</thead>
<tbody>
<tr>
<td>Something is of a certain kind</td>
<td>An individual is a member of a class, or that an individual's type is a class.</td>
</tr>
<tr>
<td>A way two things can be related to each other</td>
<td>OWL Object Property</td>
</tr>
<tr>
<td>An attribute; a way something can be linked to a literal</td>
<td>OWL Datatype Property</td>
</tr>
<tr>
<td>Say something</td>
<td>Assert an axiom, or a triple</td>
</tr>
<tr>
<td>Draw a conclusion to get more info</td>
<td>Perform inference &amp; add triples</td>
</tr>
</tbody>
</table>

Lets Revisit Those Triples

- Linking an individual to its Class
- Specify a Class as a subclass of another Class
- Every Class is an instance of owl:Class
- Every property that links to a literal is an instance of owl:DatatypeProperty
Before we jump into Healthcare
A Few Preliminaries

• How to refer to things
• Notational conventions and namespaces
• Some common vocabularies
• OWL Syntax

Notational Conventions: Identifiers

• We need unique identifiers to refer to things so we can say things about them
• They are called IRIs, meaning Internationalized Resource Identifier
• An IRI is just an international version of a URI with lots more characters
• A URL is a URI that points to a web page
  • “L” is for Location
  • “I” is for Identifier
Notational Conventions: Namespaces

- OWL adopts XML’s namespace convention
- Namespace prefixes are handy abbreviations
- Expanding an Abbreviation
  - `rdf:type`
  - `rdf` is called a Namespace Prefix
  - “`rdf:`” expands to `http://www.w3.org/1999/02/22-rdf-syntax-ns#`
  - Add “type” at the end to get the complete identifier
    - `http://www.w3.org/1999/02/22-rdf-syntax-ns#type`
- Each Namespace corresponds to a Vocabulary, typically with its own namespace prefix

Some Common Vocabularies

<table>
<thead>
<tr>
<th>Vocabulary</th>
<th>Prefix</th>
<th>Long Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>OWL 2 Schema</td>
<td>owl</td>
<td><code>http://www.w3.org/2002/07/owl#</code></td>
</tr>
<tr>
<td>RDF Concepts</td>
<td>rdf</td>
<td><code>http://www.w3.org/1999/02/22-rdf-syntax-ns#</code></td>
</tr>
<tr>
<td>RDF Schema</td>
<td>rdfs</td>
<td><code>http://www.w3.org/2000/01/rdf-schema#</code></td>
</tr>
<tr>
<td>XML Schema</td>
<td>xsd</td>
<td><code>http://www.w3.org/2001/XMLSchema#</code></td>
</tr>
</tbody>
</table>
OWL Syntax

- There are a variety of standard syntaxes for OWL
  - **Turtle**: easy to read, good for understanding underlying details
  - **Manchester Syntax**: easiest to read, hides some underlying details
  - **RDF/XML**: hard to read, hides nothing and adds some stuff that makes it even harder to read
  - And other less commonly used ones

We will use both Turtle and Manchester Syntax.
We also use our own Visual Syntax

Where We are Going

- In this section, we will learn how to use the various OWL constructs to build a portion of a healthcare ontology
- We will use more formal terminology
- We will touch on the 30-40% of OWL2 that you will use 90% of the time
Typical Healthcare Data

• A Visual OWL syntax
• Exports to text file in RDF/XML

Exercise: Identify where certain OWL constructs are being used.

Look at the previous slide and then:
• List all the owl:Thing(s)
  i.e. member of the class owl:Thing
• List all the object properties
• List all the datatype properties
• Are there any owl:Class(es)?
### Example Assertions as Triples

<table>
<thead>
<tr>
<th>Subject (Individual)</th>
<th>Predicate (Property)</th>
<th>Object (Individual/Literal)</th>
</tr>
</thead>
<tbody>
<tr>
<td>odl:_JohnDoeDoctorVisit12Jan</td>
<td>odl:careRecipient</td>
<td>odl:_PatientJohnDoe</td>
</tr>
<tr>
<td>odl:_DrJillSmith</td>
<td>odl:age</td>
<td>32</td>
</tr>
<tr>
<td>odl:_PatientJohnDoe</td>
<td>odl:firstName</td>
<td>&quot;John&quot;^^xsd:string</td>
</tr>
<tr>
<td>odl:_NurseJaneWilson</td>
<td>odl:lastName</td>
<td>&quot;Wilson&quot;^^xsd:string</td>
</tr>
<tr>
<td>odl:_DrJillSmith</td>
<td>rdfs:label</td>
<td>&quot;Dr. Jill Smith&quot;^^xsd:string</td>
</tr>
<tr>
<td>odl:_NurseJaneWilson</td>
<td>rdf:type</td>
<td>owl:NamedIndividual</td>
</tr>
</tbody>
</table>

---

### Example Triples: Things to Note

- Turtle clearly shows the triples structure
- Every Individual has an IRI as unique identifier
- IRIs may not be human readable, use labels for that
- Every literal is of a certain datatype, which may or may not be explicit. (e.g. string, integer, date)
Our Example in Turtle Syntax (1/3)

```
################################################################
#####  Object Properties  #####
###  http://ontologies.owldistilled.com/tutorial#careProvider
odl:careProvider rdf:type owl:ObjectProperty .
###  http://ontologies.owldistilled.com/tutorial#careRecipient
#####  Data Properties  #####
###  http://ontologies.owldistilled.com/tutorial#age
odl:age rdf:type owl:DatatypeProperty .
###  http://ontologies.owldistilled.com/tutorial#date
odl:date rdf:type owl:DatatypeProperty .
###  http://ontologies.owldistilled.com/tutorial#firstName
odl:firstName rdf:type owl:DatatypeProperty .
###  http://ontologies.owldistilled.com/tutorial#lastName
odl:lastName rdf:type owl:DatatypeProperty .
```

Our Example in Turtle Syntax (2/3)

```
#####  Individuals  #####
###  http://ontologies.owldistilled.com/tutorial#_JohnDoeDoctorVisit12Jan
odl:_JohnDoeDoctorVisit12Jan rdf:type owl:NamedIndividual ,
    owl:Thing ;
    rdfs:label  "John Doe's doctor visit on 12 Jan" ;
    odl:date  "12-Jan-2013"^^xsd:dateTime ;
    odl:careProvider odl:_DrJillSmith ,
        odl:NurseJaneWilson ;
###  http://ontologies.owldistilled.com/tutorial#_DrJillSmith
odl:_DrJillSmith rdf:type owl:NamedIndividual ,
    owl:Thing ;
    rdfs:label  "Dr. Jill Smith" ;
    odl:age  32 ;
    odl:firstName "Jill"^^xsd:string ;
    odl:lastName  "Smith"^^xsd:string .
```
### Our Example in Turtle Syntax (3/3)

#### Individuals

#### http://ontologies.owldistilled.com/tutorial#_NurseJaneWilson

```
odl:_NurseJaneWilson rdf:type owl:NamedIndividual ,
    owl:Thing ;
    rdfs:label  "Nurse Jane Wilson" ;
    odl:firstName "Jill"^^xsd:string ;
    odl:lastName  "Smith"^^xsd:string .
```

#### http://ontologies.owldistilled.com/tutorial#_PatientJohnDoe

```
odl:_PatientJohnDoe rdf:type owl:NamedIndividual ,
    owl:Thing ;
    rdfs:label  "Patient John Doe" ;
    odl:age  57 ;
    odl:lastName  "Doe"^^xsd:string ;
    odl:firstName  "John"^^xsd:string .
```

---

### What “Kind” of Individuals have we?

**Our Four Individuals**

- odl:_JohnDoeDoctorVisit12Jan
- odl:_NurseJaneWilson
- odl:_DrJillSmith
- odl:_PatientJohnDoe

**What is OWLese for “Kind”?**

Have we said anything at all about that?

Don’t you read too much into terms

Because computers can’t do that.
What Properties do we have?

- odl:careRecipient
- odl:careProvider
- odl:firstName
- odl:lastName
- odl:age

- Have we said anything about them?
- objectProperties vs. dataProperties

NEXT: Metadata
Introducing Metadata

- Metadata tells you about the data.
- Metadata in OWL:
  - gives meaning to the data
  - supports inference
- We don’t have explicit metadata in our example
- You can create and load triples without any metadata
- Not having metadata:
  - Can be useful for getting moving quickly
  - Bad idea for the long term

Metadata in OWL

- Which of the following OWL constructs do you think are used to specify metadata?

<table>
<thead>
<tr>
<th>What to express</th>
<th>OWL construct used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specific thing</td>
<td>owl:Thing</td>
</tr>
<tr>
<td>Kind of thing</td>
<td>owl:Class</td>
</tr>
<tr>
<td>What Kind of thing something is</td>
<td>rdf:type</td>
</tr>
<tr>
<td>Generalization / Specialization between kinds of things</td>
<td>owl:subClassOf</td>
</tr>
<tr>
<td>Relationships between things</td>
<td>owl:ObjectProperty</td>
</tr>
<tr>
<td>Attributes describing things with literals</td>
<td>owl:DatatypeProperty</td>
</tr>
</tbody>
</table>

Most of OWL is for specifying metadata.
Metadata in OWL

- `owl:Class & rdf:type` for representing the kind of thing an individual is: e.g. Person, PatientVisit
- `subClassOf` relationship to indicate that one class is a specialization of another class: e.g. Patient is `subClassOf` Person, PatientVisit is a `subClassOf` Event
- Properties: for characterizing the nature of the relationship between:
  - two individuals (e.g. careRecipient)
  - an individual and a literal (e.g. firstName)

Healthcare Class Hierarchy

- `odl:Person` as the root class
- Subclass of `odl:Event`
- `odl:Patient` subclass of `odl:PatientVisit`
- Classes are Rectangles
- Arrow means `subClassOf`
- IRI
- No `subClassOf` here
- Alternative way to show `subClassOf`
Healthcare Properties

**odl:careRecipient** [F]

- **Domain:** odl:PatientVisit
- **Range:** odl:Person

Links a PatientVisit to the Person receiving care

**Functional**

**odl:careProvider**

- **Domain:** odl:PatientVisit
- **Range:** odl:Person

Links a PatientVisit to the Person providing care

**Comment**

- What kinds of things you can link.

Object Properties

Datatype Properties

Annotations in OWL

**Labels**

- Easy to read text for humans

**Comments**

- Natural Language text to indicate the meaning of the class or property
- Very good practice
- Helps others understand the ontology
- Helps YOU remember what you were thinking
Annotations in OWL

Annotations are also represented using Triples:

- **Subject:** odl: NurseJaneWilson
  
  **Predicate:** rdfs:label

  **Object:** "Nurse Jane Wilson".

The predicates we have seen:

- rdfs:label
- rdfs:comment

- These are called “Annotation Properties”
  A 3\textsuperscript{rd} kind of property besides object & datatype

- For HUMANs only. Ignored by the inference engine.

Property Characteristics

- Properties are different from each other in a variety of specific ways.
  
  E.g. to be “functional”, is to have no more than one value

- We saw another one earlier, the “jump over” characteristic

- Does anyone know what that is called?
Applicability of Properties

• What kinds of things can you link?
• We want our properties to make sense
  • It makes no sense to say the careRecipient for a PatientVisit is a rock.
  • It makes no sense to say a rock has a last name
• There are two aspects:
  • What kind of thing must the Subject be in a triple using a particular property?
  • What kind of thing must the Object be in a triple using a particular property?

Applicability of Properties

• Applicability with respect to Subject
  • A property only applies to things of type C (for some class C)
  • OWL terminology: The domain of a property is C

• Applicability with respect to Object
  • The range of possible values that a property can have must all be of type C
  • OWL terminology: The range of a property is C
Domain and Range

Saying that “the domain of careRecipient is PatientVisit”
Means that “the Subject of every triple using careRecipient is of type PatientVisit”

Saying that “the range of careRecipient is Person”
Means that “the Object of every triple using careRecipient is of type Person”

It turns out this has some surprising consequences.

Semantics of Properties

Ways to indicate what a property means:
• Comments
• Indicate its characteristics (e.g. functional)
• Indicate its domain and range

WHY?
• to remove ambiguity
• make it easier for both people and machines to understand and use the ontology
Semantics of Classes

Ways to indicate what a class means:
- Comments
- What are its super classes
- Much more coming in a while.

WHY? Same Reason:
- to remove ambiguity
- make it easier for both people and machines to understand and use the ontology

Where are we?

- We are building parts of a healthcare ontology
- We looked at data: Individuals connected
  - to other individuals
  - to literals
- We looked at some metadata:
  - Classes, subClassOf
  - Properties
- Is there something missing?
- What class is _NurseJaneWilson an instance of?
- We need to connect the the individuals to their classes?
Our Example in Turtle Syntax

##### Individuals #####

odl:_NurseJaneWilson rdf:type owl:NamedIndividual
odl:_NurseJaneWilson rdf:type owl:Thing

For now, we just need to know now that they are ‘system’ classes that we do not specify. We want to use the classes we created to say what kind of individuals we have.

What Classes to our Individuals Belong to?

Our Four Individuals

- odl:_JohnDoeDoctorVisit12Jan
- odl:_NurseJaneWilson
- odl:_DrJillSmith
- odl:_PatientJohnDoe

We gave them sensible names, but the computer will not make sense of that. We have to be explicit.
Individuals and their Types

- We assigned the most specific classes
- Can you think why this might not always be what you want?
- Later we will see.

<table>
<thead>
<tr>
<th>Subject</th>
<th>Predicate</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>odl:_JohnDoeDoctorVisit12Jan</td>
<td>rdf:type</td>
<td>odl:PatientVisit</td>
</tr>
<tr>
<td>odl:_PatientJohnDoe</td>
<td>rdf:type</td>
<td>odl:Patient</td>
</tr>
<tr>
<td>odl: NurseJaneWilson</td>
<td>rdf:type</td>
<td>odl:Nurse</td>
</tr>
<tr>
<td>odl:_DrJillSmith</td>
<td>rdf:type</td>
<td>odl:Doctor</td>
</tr>
</tbody>
</table>

Common Terminology

- There are different ways to say in English, what kind of a thing an individual is:
  - an individual’s type is some Class
  - an individual is a member of some Class
  - an individual is an instance of some Class

- All are common and perfectly acceptable

- Next: adding more meaning to Classes
Using Properties to Add Meaning to Classes

• WARNING: this part of OWL is very tricky.

• There is a lot we can say about a PatientVisit that is necessarily true for every PatientVisit
  • It is an Event
  • Every PatientVisit has a Patient
  • Every PatientVisit has a HealthCareProvider

• While not immediately obvious, the last two indicate what properties are used in triples with PatientVisit as the Subject.

 lets make it more obvious

To think in OWL, it often helps to think in Triples.

• Every PatientVisit has at least one Patient
• Every PatientVisit is associated with one Person who is receiving care
• Every PatientVisit has a careRecipient relationship with at least one Person.
• Every PatientVisit is the Subject of at least one triple where
  • the Predicate is odl:careRecipient
  • the Object is of type Person.

Why did I change Patient to Person?

Think of this as Triplized English
Here is a Picture

- Every PatientVisit is the Subject of at least one triple where
  - the predicate is odl:careRecipient
  - the Object is of type Person.

Do same thing for CareProvider

We used a series of semantically equal sentences to get this:
- Every PatientVisit is the Subject of at least one triple where
  - the predicate is odl:careRecipient
  - the Object is of type Person.

By the same process, we also can convert:
“Every PatientVisit has a HealthCareProvider” to:

- Every PatientVisit is the Subject of at least one triple where
  - the predicate is odl:careProvider
  - the Object is of type Person.
We Can say this in OWL Directly

Normal English

Every Patient necessarily:
• is an Event
• has a careProvider (Person)
• has a careRecipient (Person)

Class: odl:PatientVisit

SubClassOf:

odl:Event,

odl:careProvider some odl:Person,

odl:careRecipient some odl:Person

PatientVisit is a subclassOf Event

Classes, Sets and Venn Diagrams

• Classes are Sets
• Subclass means Subset

Event

PatientVisit is a subclassOf Event
**Every PatientVisit has a Care Provider**

Every PatientVisit necessarily:
- is an Event
- has a careProvider (Person)
- has a careRecipient (Person)

**OWL syntax**

\[
\text{Class: odl:PatientVisit} \\
\text{SubClassOf:} \\
\text{odl:Event,} \\
\text{odl:careProvider some odl:Person,} \\
\text{odl:careRecipient some odl:Person}
\]

**Every PatientVisit has a Care Recipient**

Every PatientVisit necessarily:
- is an Event
- has a careProvider (Person)
- has a careRecipient (Person)

**OWL syntax**

\[
\text{Class: odl:PatientVisit} \\
\text{SubClassOf:} \\
\text{odl:Event,} \\
\text{odl:careProvider some odl:Person,} \\
\text{odl:careRecipient some odl:Person}
\]
PatientVisit is SubClass of Three other Classes

Class: odl:PatientVisit

SubClassOf:
    odl:Event,
    odl:careProvider some odl:Person,
    odl:careRecipient some odl:Person

Manchester Syntax

As a Venn Diagram

Event

Every PatientVisit necessarily:
• is an Event (i.e. sub class)
• has a careProvider (Person)
• has a careRecipient (Person)
Clear as mud? Let’s go over that again

- The idea was to say in OWL that every PatientVisit necessarily has a careProvider that is a Person.
- To do that, we create the class expression:
  \[(\text{odl:careProvider some odl:Person})\] which means \text{ThingWithACareProviderThatsAPerson}
- Then we make PatientVisit a subclass of it.
- Because of the way subclass works, this does the trick.
- Giving the class a meaningful (and usually long) name is optional, but a good idea when you are learning how this works.

Does this class expression have a name?

- A class expression using a property in this manner is called an OWL Restriction
- It is used to narrow down, or ‘restrict’ the possible meaning of a class.
- Individuals that have no careProvider that is a Person cannot be a PatientVisit
- This is intuitive enough… **BUT:**
OWL Restrictions: caveat

• Technically, an OWL restriction is a Class that is defined in a particular way.  
  \(<\text{property}\> \text{ some } \langle\text{Class}\rangle\)

• The way it is used, restricts meaning, but it is not very intuitive to refer to the Class itself as a “restriction”.

• For this reason, it can help to use the less ambiguous term “property restriction class” or “property restriction” for short.

Equivalence vs. Subclass

IF an individual is known to be a member of the PatientVisit class, we can already infer the following new information:

• it is an Event  
• it has a careRecipient that is a Person  
• it has a careProvider that is a person  

Can we go the other way around? If all three things are true, does that mean it must be a PatientVisit? Can you think of any counter-examples?

IF an individual is known to:

• be a member of the class Event  
• have a careRecipient that is a Person  
• have a careProvider that is a Person  

THEN we can conclude that the individual is a member of the class, PatientVisit.
**Equivalence**

- Equivalence enables us to do Automatic Categorization (using inference)

---

**Another Equivalence Example**

**PATIENT**

- What do we have to know in order to infer something to be a Patient?
- Suppose something was known to be:
  - A Person
  - A recipient of care on a PatientVisit
- Would that make them a Patient?
- Lets say it does, for our current purposes.
How can we say this in OWL?

• Let try to “tripelize” some English for an arbitrary individual: X
  • X is a Person
    X is a member of the class Person
    odl: X rdf:type odl:Person
  • X is a recipient of care on a PatientVisit
    X careRecipient some PatientVisit
• This is close, but not quite right. Why not?

Introducing Inverse Properties

• Most relationships are directional
  • Joe is hasParent Tommy is very different than Tommy hasParent Joe
  • The relationship going the other direction sometimes has a common name.
    • hasChild or parentOf would both work
  • We need the inverse of careRecipient which goes from the Patient to the PatientVisit
• What should we call it?
  • Choose names so you when you read off the Subject, Predicate and Object it is quasi-English.
    • How about “careRecipientOn”
    • X is careRecipientOn a PatientVisit.
  • We can use a Property Restriction to do this in OWL
Patient: using inverse property

- Note how we have both metadata and data above
- The metadata allows us to take that data and infer John Doe into the class Patient

Inferring John Doe to be a Patient

Get an explanation
Set Intersection

- We defined Patient to be the intersection of two sets:
  - Person
  - (CareRecipientOn some PatientVisit)
    i.e. ThingThatIsACareRecipientOnSomePatientVisit

- Exercise: draw a Venn diagram to show this.

Intersection: More Examples

Class: odl:PerformedProcedure
Annotations: rdfs:comment
  "A procedure that has already been performed."
EquivalentTo: odl:HistoricalEvent and odl:Procedure
Classes Are Sets

- Formally, an OWL Class refers to a set of things.
- `odl:TwoWheeledVehicle` refers to the set of all two-wheeled vehicles.
- `odl:MotorizedVehicle` refers to the set of all motorized vehicles.
- `odl:Motorcycle` refers to the intersection of these two sets.

- Intersection is a common set operation.
- We can also use set Union to create classes.

Class Expressions using Union

- Intersection corresponds to AND
- Union corresponds to OR

You can also make a class using set complement, but it is not that common.
Specifying Meaning

- We just saw a number of ways to specify meaning for Classes
- We saw different expressions
  - Union, Intersection, Property Restrictions
- Next we look at specifying more meaning for properties.

Property Generalization/Specialization

- Sometimes it useful to capture the fact that one property is more general than another property.
- Simple Example: sibling, brother, sister

IF: X sisterOf Y
THEN: X siblingOf Y

IF: X brotherOf Y
THEN: X siblingOf Y

- Can you think of other examples in Healthcare or any other subject area?
Enterprise Example: subPropertyOf

IF: X employedBy Y
THEN: X worksFor Y

IF: X contractorFor Y
THEN: X worksFor Y

• Highly analogous to subClassOf

subPropertyOf vs. subClassOf

subClassOf
IF the more specialized class has X as a member
THEN the more general class has X as a member

subPropertyOf
IF the specialized property relationship holds between X and Y
THEN the more general property relationship holds between X and Y
Property Characteristics

• **Functional**: there is no more than one value
e.g. hasBiologicalMother, careRecipient

• **Transitive**: the “jump over” property
e.g. subClassOf, largerThan, physicalPartOf.
What about: consistentlyDefeatsInAGameOfTennis?

• **Symmetric**: the property goes both ways
e.g. siblingOf, colleagueOf
What about: hasBestFriend?

Where Are We?

• We just had a close look at how to add semantics to classes:
  • Property restrictions
  • Intersection and Union

• We also learned how to add semantics to properties:
  • Sub-property
  • Characteristics: functional, transitive, symmetric

• NEXT: helping the inference engine find errors
Help the Inference Engine Find Errors

- Saying that two classes don't overlap helps find errors.
- In terms of Sets, not overlapping means disjoint

Class: odl:CancerPatient SubClassOf: odl:Patient
Class: odl:Patient SubClassOf: odl:Person
Class: odl:PatientVisit SubClassOf: odl:Event
Class: odl:Person DisjointWith: odl:Event

We saw this example earlier
Part 3: Patterns, Pitfalls & Conclusion

Naming Conventions

- **Classes are upper case** (e.g. Patient)
- **Properties: camelCase** (e.g. careRecipient)
- **Individuals: start with “_”** (e.g. _JillSmith)
- **Property Inverses**
  - hasX/xOf (e.g. hasParent / parentOf)
  - Xes/XdBy (e.g. identifies / identifiedBy)

- Whatever conventions you adopt, use them consistently and uniformly
Distinctionary Pattern

• Define a new class C using two things
  • Say what more general class C is a subclass of
  • Say what distinguishes a member of class C from the more general class.

• What was the example we already saw?

Define Concepts from Primitives

• Keep the number of primitives in your ontology relatively small
• Use equivalence to define other concepts using the set of primitives
Use Necessary Conditions

- Sometimes you don’t know enough to infer membership, so equivalence will not work.
- Use property restrictions on their own using subclass.

High Level Disjoint

- Declare classes high up in the class hierarchy to be disjoint.
- Catches a lot of errors
Pitfall: Reading too much into labels and comments

- Labels and comments are very useful to help PEOPLE understand and use an ontology.
- They are useless to the inference engine.
- It is easy to look at a label like “NurseJaneWilson” and assume that the individual is a member of a class called Nurse.
- And then wonder why the inference engine gives a “wrong” result.

Pitfall: Over-constrain Domain & Range

- Say we model start and end dates for events.
- Later on we want to model effective dates for Agreements.
- BUT: effective date seems to be a subproperty of startDate.
- Do you see the problem?
Pitfall: Over-constrain Domain & Range

- The way Domain works, an Agreement would have to be a kind of Event.
- Signing an agreement is an event, but the Agreement itself, probably not.
- Remember siblingOf, brotherOf and sisterOf?
- Domain would be Male Person & Female Person, but still Persons!

Pitfall: Over-constrain Domain & Range

SOLUTION:
- Broaden the domain of startDate to include both Agreement and Event
- Now you can use sub-property
Pitfall: Forcing Class Relationships

- Sometimes classes seem closely related so you naturally want to express that relationship. E.g. Charity and Legal Entity
- But on closer examination, you realize that neither is a subclass of the other
  - Many charities are small enough to not need any legal documentation
  - Most Legal Entities are not charities.
- They DO overlap, but there is no useful and easy way to say that.

Pitfall: Forcing Class Relationships

Recommendation:
- DO NOT give in to the temptation to force an equivalence or subclass relationship.
- DO Make both classes be subclasses of a common class
- DO See if there is a useful subclass of one that can be made a subclass or superclass of the other (e.g. Legal Charity)
- Otherwise, just indicate the relationship between the two classes in a comment (e.g. 90% of the time there is overlap)
Pitfall: Proliferation of Namespaces

- It is common practice to mint a new namespace for each new ontology module.
- We think this is bad practice.
- It makes it much harder to re-factor the ontology
- Keep same namespace:
  - for closely related subject matter
  - governed by same people

Challenges of Ontology Reuse

- Reuse seems like a great idea
- Reuse other people’s thinking
- Save time

BUT:
- there are many ways to do this
- many pitfalls lurking
- carefully weigh the pros and cons
Different Ways to Reuse

- Import an ontology directly
- Reuse the URI, but do not import
- Use the ontology for inspiration
  - do not import
  - do not use same URIs

Importing

- You get everything, whether you want it or not
- What seems to be the perfect match for what you want, may not work in your situation
- E.g. W3C Media Ontology has properties:
  - date
  - isMemberOf (to put something into a collection)
- But you cannot use them for anything other than a MediaResource.
- Why?
  Hint: One of the pitfalls we just mentioned.
Reuse URIs, but do not import

There are two ways to do this

- Easy way: just refer to the URI; nothing else. RISK: not using the term as it was intended invites misuse and technical debt
- Hard way: take the time to manually extract the axioms that define the URI you want.
  - More reliable
  - supports inference for consistency checking
  - Takes time, but only works if you agree with axioms

Reuse for Inspiration

- Take what you like, augment as necessary
- Leave the rest behind
- Mint your own URI
  - good karma to acknowledge source.
- Adapt and customize to your needs.
- More work, but is the right fit
Issues that Arise

• Domains and Ranges often overly constrained
• Inconsistent conventions
  • naming
  • definitions and comments
• OWL2 compatibility – e.g. xsd:date is not supported
• Ontology evolution: if the borrowed ontology evolves, is work to keep it up to date

Recommendations

• Be clear on expected advantages of reuse
• Take into account how long it takes to search around for existing ontologies, and choose among the options.
• Reuse a minimum of ontologies to simplify ongoing evolution.
• Check very carefully before you decide to import an external ontology.
More Recommendations

• Some URIs are widely used and you can safely reuse them w/o bringing in the axioms
• If you want to run inference to check consistently of the ontology, you probably want to bring in the axioms.
• If you cannot find exact match, use other ontologies for inspiration, and make changes.

OWL Distilled

• You just learned 41 of the 142 of the constructs on the OWL 2 Quick Reference Guide
• That is just 29%
• 90% of the time, that is all you need.

We are the Twenty-Nine Percenters!
Another say 1 1%: Commonly Used

- Object and Datatype Property Restrictions
  - Cardinality: min, max, exactly
    - lastName exactly 1
  - Has value: e.g. governedBy value Google
    The set of all things governed by Google
    Includes all formal & informal Google organizations
- Import one ontology into another
- Classes: Complement & Enumeration
- Annotations:
  - Several more common ones
  - Make your own
  - Can be arranged in a hierarchy

The 60%: Advanced OWL Topics
Infrequently used

- Property Chains
- More Property Characteristics
- Negative assertions & disjoint properties
- Blank Nodes & Anonymous Individuals
- Keys
The 60%: Advanced OWL Topics Infrequently used

- Datatypes
  - Built-in: many more defined by XML
  - User defined datatypes
    - Teenager: range of integers [13-19]
    - Many more variations
  - Restrictions using user defined datatypes

OWL: The Open World

- The Most Advanced OWL Topic
  (not a construct, but an underlying paradigm)
- Most Systems:
  - If it cannot find an answer, it just assumes NO
- OWL and the Open World
  - If it cannot find an answer, it assumes nothing
  - It might be true, but not enough information yet
  - Distinguishes between: NO and DON'T KNOW
- Causes a lot of confusion
- Inferences will surprise you
  - Integrity vs. Inference
If You Want More…

- For a Written Version of this Talk
- Just leave us your card

Thank You

Michael Uschold &
Dave McComb
Semantic Arts
www.semanticarts.com