OWL Simplified English*

Richard Power
Open University, UK

* A finite-state language for ontology editing
Semantic Web Authoring Tool
(EPSRC 2009-2012)

Open University (Department of Computing)
  Richard Power
  Sandra Williams
  Allan Third
  Tu Anh Nguyen

Manchester University (School of Computer Science)
  Robert Stevens
  Alan Rector
  Fennie Liang

Sussex University (Department of Informatics)
  Donia Scott
Objectives

Theoretical
Clarify relationship of formal languages (OWL) to natural languages (English)

Practical
Develop tools for viewing and editing OWL ontologies in natural language
SWAT Natural Language Tools

Analyse your OWL ontology, build a lexicon from it, or convert it into English sentences or definition paragraphs.

Select the ontology file

File format should be OWL/XML (.xml) or OWL/RDF (.owl).

Select the output format

- Alphabetical English glossary (class, individual and property definitions)
- English sentences (one sentence per OWL axiom)
- Prolog terms (translate OWL to Prolog)
- Lexicon (lexical entries for class, individual and property names)
- Axiom patterns (frequency counts)
- OWL/XML, with verbalisations as "SWATDescription" annotations.

Explanations of these outputs are given here.
<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adult (class)</td>
<td>Elderlies, and drivers are adults.</td>
<td>No adult is a young.</td>
</tr>
<tr>
<td>Animal (class)</td>
<td>An animal eats a thing. If X has as pet Y then Y is an animal. If X eats Y then X is an animal.</td>
<td>The following are animals: tigers, sheep and people, and so on (6 items in total).</td>
</tr>
<tr>
<td>Animal Lover (class)</td>
<td>An animal lover is defined as a person that has as pet at least three things.</td>
<td></td>
</tr>
<tr>
<td>Bicycle (class)</td>
<td>A bicycle is a vehicle.</td>
<td></td>
</tr>
<tr>
<td>Bone (class)</td>
<td>No information.</td>
<td></td>
</tr>
<tr>
<td>Brain (class)</td>
<td>No information.</td>
<td></td>
</tr>
<tr>
<td>Broadsheet (class)</td>
<td>A broadsheet is a newspaper.</td>
<td></td>
</tr>
</tbody>
</table>
# OntologyIRI http://www.swatproject.org/ontologies/testing

% ClassAssertion
John is a bigamist.

% ObjectPropertyAssertion
John is married to Mary.
John is married to Jane.

% DataPropertyAssertion
John is aged 35.
John is nicknamed "Mr Polygamous".

% SubClassOf
A husband is a man.
Every wife is a woman.

% EquivalentClasses
A bigamist is defined as a man that is married to at least two people.
A husband is defined as a married man.

% DisjointClasses
No husband is a wife.

% ObjectPropertyDomain
Anything that is married to something is a person.

A|An|The|Every|No|Anything|PROPER–NAME
Unspecified(null,null)
Download editing tool

http://mcs.open.ac.uk/rp3242/editor/

Requires Java runtime environment
Outline

• Motivation
• Demonstration
• Language
• Coverage
• Conclusion
## Previous work

**Controlled Natural Languages**

- Attempto Controlled English (ACE)
- Sydney OWL Syntax (SOS)
- Rabbit

**Ontology Editing Tools**

- ACE Wiki
- ROO (Rabbit to OWL Ontology construction)
- RoundTrip Ontology Authoring
OWL Simplified English

- Very simple rules for forming sentences
- Little or no effort required to build lexicon
- Disallows structurally ambiguous sentences
- Can be interpreted by finite-state transducer
- Coverage limited in theory, adequate in practice
Editing tool

- User edits text as in predictive authoring
- Patterns for a complete sentence are offered as in WYSIWYM
- Patterns contain anchors for entity names (individual, class, property) for which options are computed from the current text
- Efficient implementation is much easier if the grammar is finite-state
Outline

• Motivation
• Demonstration
• Language
• Coverage
• Conclusion
A city is a geographical location.
A country is a geographical location.
Every city is located in a country.

London is a city.
The United Kingdom is a country.

The Tate Modern is an art gallery that is located in London.
The Tate
A city is a geographical location.
A country is a geographical location.
Every city is located in a country.

London is a city.
The United Kingdom is a country.

% The Tate Modern is an art gallery that is located in London.
The Tate Modern

is a [class].
is defined as a [class].
[has-property] [Individual].
[has-data-property] [literal].
[has-property] a [class].
[has-property] only [class].
[has-property] exactly [integer] [class].
[has-property] at least [integer] [class].
[has-property] at most [integer] [class].
A city is a geographical location.
A country is a geographical location.
Every city is located in a country.

London is a city.
The United Kingdom is a country.

% The Tate Modern is an art gallery that is located in London.
The Tate Modern is a [class].

Complete sentence
ClassAssertion(NamedIndividual(#The_Tate_Modern),Class(#[class]))
% Comments message

A city is a geographical location.
A country is a geographical location.
Every city is located in a country.

London is a city.
The United Kingdom is a country.

% The Tate Modern is an art gallery that is located in London.
The Tate Modern is a [class].

Complete sentence
ClassAssertion(NamedIndividual(#The_Tate_Modern),Class(#[class]))
A city is a geographical location.
A country is a geographical location.
Every city is located in a country.

London is a city.
The United Kingdom is a country.

% The Tate Modern is an art gallery that is located in London.
The Tate Modern is an art gallery.

Complete sentence
ClassAssertion(NamedIndividual(#The_Tate_Modern),Class(#art_gallery))
A city is a geographical location.
A country is a geographical location.
A city is located in a country.

London is a city.
The United Kingdom is a country.

The Tate Modern is an art gallery that is located in London.
The Tate Modern is an art gallery
A **city** is a geographical location.
A **country** is a geographical location.
A **city** is located in a **country**.

**London** is a **city**.
**The United Kingdom** is a **country**.

% The Tate Modern is an art gallery that is located in London.
**The Tate Modern** is an **art gallery** that [has-property] [Individual].

Complete sentence
ClassAssertion(NamedIndividual(#The_Tate_Modern),ObjectIntersectionOf(Class(#art_gallery),ObjectHasValue(ObjectProperty([has-property]),NamedIndividual(#[Individual])))))
% Comments message

A city is a geographical location.
A country is a geographical location.
A city is located in a country.

London is a city.
The United Kingdom is a country.

% The Tate Modern is an art gallery that is located in London.
The Tate Modern is an art gallery that has–property [Individual].

Complete sentence
ClassAssertion(NamedIndividual(#The_Tate_Modern),ObjectIntersectionOf(Class(#art_gallery),ObjectHasValue(ObjectProperty (#[has–property]),NamedIndividual(#[Individual]))))
A city is a geographical location.
A country is a geographical location.
A city is located in a country.

London is a city.
The United Kingdom is a country.

% The Tate Modern is an art gallery that is located in London.
The Tate Modern is an art gallery that is located in [Individual].

Complete sentence
ClassAssertion(NamedIndividual(#The_Tate_Modern),ObjectIntersectionOf(Class(#art_gallery),ObjectHasValue(ObjectProperty(#is_located_in),NamedIndividual(#[Individual])))
A city is a geographical location.
A country is a geographical location.
A city is located in a country.

London is a city.
The United Kingdom is a country.

The Tate Modern is an art gallery that is located in London.
Outline

• Motivation
• Demonstration
• Language
• Coverage
• Conclusion
Axiom in OSE and OFS

London is a city that is capital of the United Kingdom and is divided into at least 30 boroughs.

```
ClassAssertion(Class(#London),
   ObjectIntersectionOf(Class(#city),
      ObjectHasValue(ObjectProperty(#capitalOf),
         NamedIndividual(#UK))
   ObjectMinCardinality(30,
      ObjectProperty(#dividedInto),
      Class(#borough))))
```
Restricted words

London is a city that is capital of the United Kingdom and is divided into at least 30 boroughs.

**ENTITY NAMES**
- Individual name
- Class name
- Property name

Some words are used only as scaffolding, and cannot be included in an entity name.
## Word categories

<table>
<thead>
<tr>
<th>Category</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntactic sugar</td>
<td>every, no, a/an, and, or, that, ...</td>
</tr>
<tr>
<td>Number</td>
<td>two, 365, 3.14, ...</td>
</tr>
<tr>
<td>String</td>
<td>“Pride and Prejudice”, “XY123”, ...</td>
</tr>
<tr>
<td>Verb (present)</td>
<td>is, has, takes, participates, ...</td>
</tr>
<tr>
<td>Proper noun</td>
<td>John, X23, London, ...</td>
</tr>
<tr>
<td>Preposition</td>
<td>of, by, in, from, ...</td>
</tr>
<tr>
<td>Noun/other</td>
<td>person, taken, yellow, slowly, ...</td>
</tr>
</tbody>
</table>
### How words are categorised

<table>
<thead>
<tr>
<th>Category</th>
<th>Example</th>
<th>List Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntactic sugar</td>
<td><code>a/an, and,...</code></td>
<td>Listed in program</td>
</tr>
<tr>
<td>Number</td>
<td><code>two, 365, ...</code></td>
<td>Number words, digits</td>
</tr>
<tr>
<td>String</td>
<td>“XY123”, ...</td>
<td>Double quotes</td>
</tr>
<tr>
<td>Verb (present)</td>
<td><code>takes, ...</code></td>
<td>Listed by USER</td>
</tr>
<tr>
<td>Proper noun</td>
<td><code>John, ...</code></td>
<td>Upper-case letter</td>
</tr>
<tr>
<td>Preposition</td>
<td><code>of, by, ...</code></td>
<td>Listed in program</td>
</tr>
<tr>
<td>Noun/other</td>
<td><code>person, ...</code></td>
<td>Lower-case letter</td>
</tr>
</tbody>
</table>
## Entity names

<table>
<thead>
<tr>
<th>Entity</th>
<th>Opening</th>
<th>Continuation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Individual</strong></td>
<td>Proper noun, ‘the’</td>
<td>Proper noun, ‘the’, Number, String, Preposition, Noun/other</td>
</tr>
<tr>
<td><strong>Class</strong></td>
<td>Proper name, ‘the’,</td>
<td>Proper name, ‘the’, Number, String, Preposition, Noun/other</td>
</tr>
<tr>
<td></td>
<td>Number, String, Preposition, Noun/other</td>
<td></td>
</tr>
<tr>
<td><strong>Property</strong></td>
<td>‘is’, ‘has’, Verb (present)</td>
<td>Noun/other, Preposition</td>
</tr>
<tr>
<td><strong>Literal</strong></td>
<td>Number, String</td>
<td></td>
</tr>
</tbody>
</table>
Reason for these rules

The author is not required to define names for individuals, classes and properties in advance, so the system must infer when they start and end.

The queen is a woman that lives in Buckingham Palace and is married to a Greek that is named “Phillip”.

- Individual
- Class
- Property
- Literal
Tony Blair is married to a lawyer.
Tony Blair is married to a lawyer.

ClassAssertion(NamedIndividual(#Tony),null)
Tony Blair is married to a lawyer.

ClassAssertion(NamedIndividual(#Tony_Blair),null)
Tony Blair is married to a lawyer.

ClassAssertion(NamedIndividual(#Tony_Blair),null)
Tony Blair is married to a lawyer.

ClassAssertion(NamedIndividual(#Tony_Blair), UnspecifiedRestriction(ObjectProperty(#is_married),null))
Tony Blair is married to a lawyer.

ClassAssertion(NamedIndividual(#Tony_Blair), UnspecifiedRestriction(ObjectProperty(#is_married_to),null))
Tony Blair is married to a lawyer.

ClassAssertion(NamedIndividual(#Tony_Blair), ObjectSomeValuesFrom(ObjectProperty(#is_married_to), Class()))
Tony Blair is married to a lawyer.

ClassAssertion(NamedIndividual(#Tony_Blair),
ObjectSomeValuesFrom(ObjectProperty(#is_married_to),Class(#lawyer)))
Tony Blair is married to a lawyer.

ClassAssertion(NamedIndividual(#Tony_Blair), ObjectSomeValuesFrom(ObjectProperty(#is_married_to), Class(#lawyer))))
Basic sentence patterns

- A city is a geographical location.
- A country is a geographical location.
- A city is located in a country.

- London is a city.
- The United Kingdom is a country.
A city is a geographical location.
A country is a geographical location.
A city is located in a country.

London is a city.
The United Kingdom is a country.

% The Tate Modern is an art gallery that is located in London.
The Tate Modern is an art gallery
Sentence structure

Sentence = Subject  Predicate

Subject = [Individual]
Subject = A|Every|No [Class]

Predicate = is NPLList  that VPLList  that VPChain

NPLList = a [Class] and a [Class] ...
VPLList = [Props] a [Class] and [Props] ...
VPChain = [Props] a [Class] that [Props] ...
Outline

• Motivation
• Demonstration
• Language
• Coverage
• Conclusion
## Complex axiom patterns

<table>
<thead>
<tr>
<th>Predicate pattern</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C \cap P.C$</td>
<td>1100</td>
</tr>
<tr>
<td>$P.(C \cap P.C)$</td>
<td>473</td>
</tr>
<tr>
<td>$P.(C \cup C)$</td>
<td>434</td>
</tr>
<tr>
<td>$C \cap (P.C \cap P.C)$</td>
<td>248</td>
</tr>
<tr>
<td>$P.(C \cap (P.(C \cap P.C)))$</td>
<td>164</td>
</tr>
<tr>
<td>$P.(C \cup (C \cap C))$</td>
<td>105</td>
</tr>
<tr>
<td>$C \cap (P.C \cap (P.C \cap P.C))$</td>
<td>78</td>
</tr>
<tr>
<td>$P.C \cap P.C$</td>
<td>59</td>
</tr>
<tr>
<td>$C \cap (P.C \cap (P.C \cap P.C))$</td>
<td>47</td>
</tr>
<tr>
<td>$P.(C \cup (C \cap (C \cap C)))$</td>
<td>43</td>
</tr>
<tr>
<td>$C \cap (P.(C \cap P.C))$</td>
<td>39</td>
</tr>
<tr>
<td>$P.(C \cup (C \cap (C \cap (C \cap C))))$</td>
<td>35</td>
</tr>
<tr>
<td>$P.(P.C)$</td>
<td>33</td>
</tr>
<tr>
<td>$P.\neg C$</td>
<td>28</td>
</tr>
<tr>
<td>$P.(C \cup (C \cap (C \cap (C \cap (C \cap C))))))$</td>
<td>26</td>
</tr>
<tr>
<td>$P.C \cap P.C$</td>
<td>25</td>
</tr>
<tr>
<td>$P.C \cap (P.C \cap P.C)$</td>
<td>20</td>
</tr>
<tr>
<td>$C \cap (P.C \cap (P.C \cap (P.C \cap (P.C \cap P.C))))$</td>
<td>20</td>
</tr>
<tr>
<td>$\neg P.C$</td>
<td>19</td>
</tr>
<tr>
<td>$P.(C \cup (C \cap (C \cap (C \cap (C \cap (C \cap C))))))$</td>
<td>17</td>
</tr>
</tbody>
</table>

### Results from corpus of over 550 ontologies

99.8% of axioms had simple subject term

All the top 20 complex predicate patterns are within the constraints of OWL

### Simplified English

If axiom patterns were created randomly we would expect just 2-3 to lie within our constraints
Three fundamental patterns

• Genus-Differentia (Aristotle)
  – A pet-owner is a person that owns a pet

• Restriction list
  – A pet-owner owns a pet and cleans a cage

• Alternative role-fillers
  – A pet-owner owns a cat or a dog or a canary
Measuring practical coverage

• Enumerate all possible complex class expressions up to a given complexity level
• Apply a criterion to determine which expressions yield ambiguous sentences
• Count the expected frequency of ambiguous sentences if all complex class expressions were equally likely
• Compare with the observed frequency for complex class expressions in an ontology corpus
Structural ambiguity

\[ C \subseteq \exists P. (C \cap C) \]

A N Vs a N and a N.
A child has as parent a mother and a father.

\[ C \subseteq \exists P. (C \cap \exists P. C \cap \exists P. C) \]

A N [Vs a N that Vs an N] and Vs a N.
A N Vs a N that [Vs an N and Vs a N].

A queen appoints a minister that governs a country and wears a crown.
Enumerating complex classes (1)

\[ C \subseteq \exists P.(C \cap C) \]

\[
\exists \\
\exists \\
P \\
\exists \\
C \\
\]

Complexity = 2 (number of non-terminal nodes)
Enumerating complex classes (2)

\[
C \subseteq \exists P. (C \cap \exists P.C \cap \exists P.C)
\]

Complexity = 5 (normalised to binary tree)
The *expected* frequency of complex axioms yielding ambiguous verbalisations was 2416/5523 or 43.7%.

The *obtained* frequency was 84/5523 or 1.5%.
Outline

• Motivation
• Demonstration
• Language
• Coverage
• Conclusion
Conclusions on complexity

• Overwhelmingly ontology authors favour complex class constructions that are not structurally ambiguous when verbalised

• Therefore, if we restrict sentence patterns to avoid structural ambiguity, almost all axioms found in our corpus could be formulated

• Probably many of the remaining axioms could be refactored
Main ideas in OSE

• Editing tool combines predictive authoring and WYSIWYM
• Finite-state controlled language favours efficient implementation and prevention of structural ambiguity
• Language requires minimal lexical input from user (verb list)
• Language allows but does not impose correct English
Will it work

User studies still pending ...
Questions?

... but thanks for your attention