

Proof of Concept

Controlled Natural Language Logic For Knowledge Engineering At 1Spatial

Anthony Beck

September 19, 2014

Abstract

1Spatial is in the business of data quality. The software portfolio is aimed at the collection and collation of spatial data and the auditing, integration and enhancement of corporate spatial data assets. The spatial functionality can be deprecated to provide generic data quality enhancements. Much of the consulting work involves building longer-term synergistic relationships with large corporations who recognise the benefits of good quality provenanced data on operational processes. The premise for consultant activities is as follows:

Good quality data with a well understood scope leads to better decisions.

During consultation we need to rapidly determine a number of things:

- the actual nature of the data (a data profiling exercise)
- the expected nature of the data (an ‘as-design’ data modelling exercise)
- the business problems which the data is expected to help solve (a business modelling exercise)
- the business operational processes (an ‘as-built’ data modelling exercise)¹

The ability to express rules and logic in syntactically simple phrases is becoming increasingly important to how we develop an understanding and effectively communicating our clients business needs (expecially in helping to distinguish between their pre-conceived and actual need). To facilitate this process 1Spatial uses *RuleSpeak* to both capture and communicate business rules. RuleSpeak is an example of a Controlled Natural Language (CNL). The point is to build an understanding of how the business systems are expected to function (the ‘as-design’ model) and compare this to how they actually function (the ‘as-built’ model) mediated by the problems the business systems are expected to solve (the business process model). Much of this information is undocumented and needs extracting from domain experts. Domain experts are not knowledge engineers or logicians. CNL helps extract this information in a manner which the domain experts can understand and provide feedback on. Hence, the abstracted model is a closer reflection to reality.

Whilst the RuleSpeak CNL reduces or removes ambiguity it is only understandable by a human reader². The Knowledge Engineering community have developed other CNLs which can be used to create Formal Ontologies.

Ontologies are a way to capture and share people’s knowledge about the world in a way that is processable by computer systems. Ontologies have the potential to serve as a bridge between the human conceptual understanding of the world and the data produced, processed and stored in computer systems. CNL can be used to develop ontologies then we can use to enhance our approaches and improve our services.

This document describes a proof of concept implementation of the ACE/APE CNL which can be used to derive an OWL formal ontology which in turn can be used for automated reasoning. The data deployed is based upon a real case study from United Utilities based on the Open Water policy initiative. Capture of data in this manner will allow 1Spatial to build up a formal knowledge base which could have a major impact on company operations.

This proof-of-concept has not been produced on 1Spatial company time.

¹Not completely happy with this

²I don’t know of a RuleSpeak Natural Language Processing (NLP) parser

Contents

1	Rules, Ontologies and the 1Spatial consultancy model	4
2	Controlled Natural Language systems for Ontology engineering	5
2.1	Candidate systems	5
2.1.1	APE and ACE	6
3	Proof of concept problem - The impact of Open Water and United Utilities	6
3.1	Writing OWL ontologies in ACE	8
3.1.1	Individuals	9
3.1.2	Classes	10
3.1.3	Properties	12
3.1.4	Reasoning	13
3.2	Roo Language Elements	15
3.2.1	Declarations	15
3.2.2	Intersection and Union	17
3.2.3	Ontology Referencing	18
3.2.4	Property Restrictions - An Area of Weakness	19
	Bibliography	19

Distribution list:

- Mike Sanderson
- Matt Beare
- Alan Howie
- Bob Chell
- Stephen Stanley

A version of this document needs sending to UU

- John Daniels
- Lee Mooney (lee.mooney@uuplc.co.uk)

1 Rules, Ontologies and the 1Spatial consultancy model³

The ability to express rules and logic in syntactically simple phrases is becoming increasingly important to how we develop an understanding of clients business needs and then effectively communicating this understanding to our clients. This is particularly important when identifying the difference between a client's pre-conceived need and their actual need (as reflected in the data and the surrounding business processes). To facilitate this 1Spatial use *RuleSpeak* to both capture and communicate business rules. RuleSpeak is an example of a Controlled Natural Language (CNL). Whilst Rulespeak reduces or removes ambiguity it is only understandable by a human reader⁴. There are other CNLs which have been developed in the Knowledge Engineering community which focus on developing Formal Ontologies. This extends the utility of the rules by ensuring they are human *and computer* readable.

Ontologies have been proposed and studied in the last couple of decades as a way to capture and share people's knowledge about the world in a way that is 'computer readable'. Ontologies have the potential to serve as a bridge between the human conceptual understanding of the world and the abstracted view of the world stored in computer systems.

One of the main reasons for the lack of widespread adoption of ontologies is the steep learning curve for authoring them: most people find it too difficult to learn the syntax and formal semantics of ontology languages. If CNL can be used to develop ontologies then we can produce a more formal and rigorous method of modelling data, conceptual and business processes and sharing these models in meaningful ways with our business partners. For 1Spatial this has the added benefit of *accruing a knowledge base*.

From a consultancy perspective 1Spatial is employed because the added business value to the client is greater than the cost of adding the value⁵. **Quality and knowledge is our business.**

During consultation we need to rapidly determine a number of things:

- the actual nature of the data (a data profiling exercise)
- the expected nature of the data (an 'as-design' conceptual modelling exercise)
- the business problems which the data is expected to help solve (a business requirements exercise)
- the business operational processes (an 'as-built' business modelling exercise)⁶

1Spatial has generic skills in data and spatial data quality enhancement but also has expert knowledge in different operational sectors (National Mapping Agencies, Utilities, Defense, Transport etc.). However, we are poor at re-using this knowledge. **Being able to effectively re-use our *knowledge base* provides 1Spatial with a significant competitive advantage.** 1Spatial can capitalise on the knowledge gained from each project more effectively. When working in different organisations which reflect our domains of expertise⁷ there already exists a domain rich model which can be re-used in different business environments (i.e. we already know how *utility A* models water assets.... can this be used to help solve problems for *utility B*?). This knowledge should also be used to identify new market opportunities: for example, the impacts of a new policy initiative (or policy change) can be modelled and inform products and services which can be pitched across the different sectors in a timely manner⁸.

The ability to rapidly generate formal models and build these into a corporate knowledge base represents a significant opportunity for 1Spatial to transform and improve how it operates⁹. This document outlines a proof of concept approach for generating such models.

³Speak to Alan about this

⁴I don't know of a RuleSpeak Natural Language Processing (NLP) parser

⁵this cost is not always calculated financially: improving business value may be mandated by changes in policy terms with metrics that are not calculated financially but where failure to achieve such metrics does incur a financial penalty

⁶Not completely happy with this

⁷each with different but complementary views over the same domain

⁸This is precisely the case for the clean and waste water sector of the utility market where the *Open Water* initiative represents a significant business opportunity for 1Spatial

⁹Alan: would appreciate your perspective on this

2 Controlled Natural Language systems for Ontology engineering

Ontology development requires at least two types of people:

- A knowledge engineer - who can structure the underlying logic that supports the knowledge model
- A domain expert - who has in depth understanding and experience of the phenomena to be modelled

The mathematical nature of description logic has meant that domain experts find them hard to understand. This forms a significant impediment to the creation and adoption of ontologies. Ontology focussed Controlled Natural Language systems constrain language syntax and allow statements to be translated into the Web Ontology Language (OWL) with the aim of achieving both comprehension by domain experts and computational preciseness.

The fundamental principles underlying the use of CNLs for Ontology engineering are (after Denaux (2013)):

- To allow the domain expert, with the aid of a knowledge engineer and tool support, to express their knowledge as easily and simply as possible and in as much detail as necessary.
- To have a well defined grammar and be sufficiently formal to enable those aspects that can be expressed as OWL to be systematically translatable.
- To be comprehensible by domain experts with little or no knowledge of OWL.
- To be independent of any specific domain.

These principles can be achieved - at least in part. One such example is the use of the Rabbit CNL developed by Ordnance Survey (Hart et al. (2008)):

The original intention was for Rabbit to enable domain experts alone to author ontologies. However, practice showed that whilst domain experts could build ontologies, these ontologies often contained many modelling errors not related to the language but the modelling processes. None-the-less Rabbit still enables the domain expert to take the lead and to author ontologies with guidance in modelling techniques from a knowledge engineer. Rabbit also enables other domain experts to verify the ontology.

CNLs should contain a number of language element. Those used to

- express axioms
- introduce (or declare) concepts and relationships
- ADD MORE

2.1 Candidate systems

Two candidate systems were considered:

- ROO and Rabbit (Denaux 2013, Hart et al. (2008))
- APE and ACE (Anon. 2014)

We see Rabbit as complementary to OWL, extending its reach to those who need to author and understand domain ontologies but for whom descriptions logics are difficult to comprehend even when expressed in more user-friendly forms such as the Manchester Syntax.

ROO is a

Rabbit is a

2.1.1 APE and ACE

3 Proof of concept problem - The impact of Open Water and United Utilities

United Utilities (UU) provides both *water* and *wastewater services*. The location of the *termination point* of a United Utilities *wholesale service asset* to a customer is called a *service point*. These service points may be for water supply, sewerage services, metering, trade effluent and any other service within the wholesale service catalogue. A service point is a sub-set of the UU wholesale assets which is uniquely billable and has a geographical location. This locational geography needs to adequately represent billing and operational activities.

Service Connectivity



Figure 1: Service Point connectivity at United Utilities

The Address Management System (AMS) fulfills parts of this need. AMS is a bespoke data repository holding *Service Point Addresses* for water, wastewater and electricity (now deprecated) domains. AMS is not an address gazetteer.

AMS records are related to a *service point*:

- In most cases these service points are at an addressable property (or a *standard address*) such as a house or office. These have an *address_type* of *postal* (PO).
 - There are also some legacy records that refer to postal addresses outside the UU operational area. These have an *address_type* of *foreign* (FO).

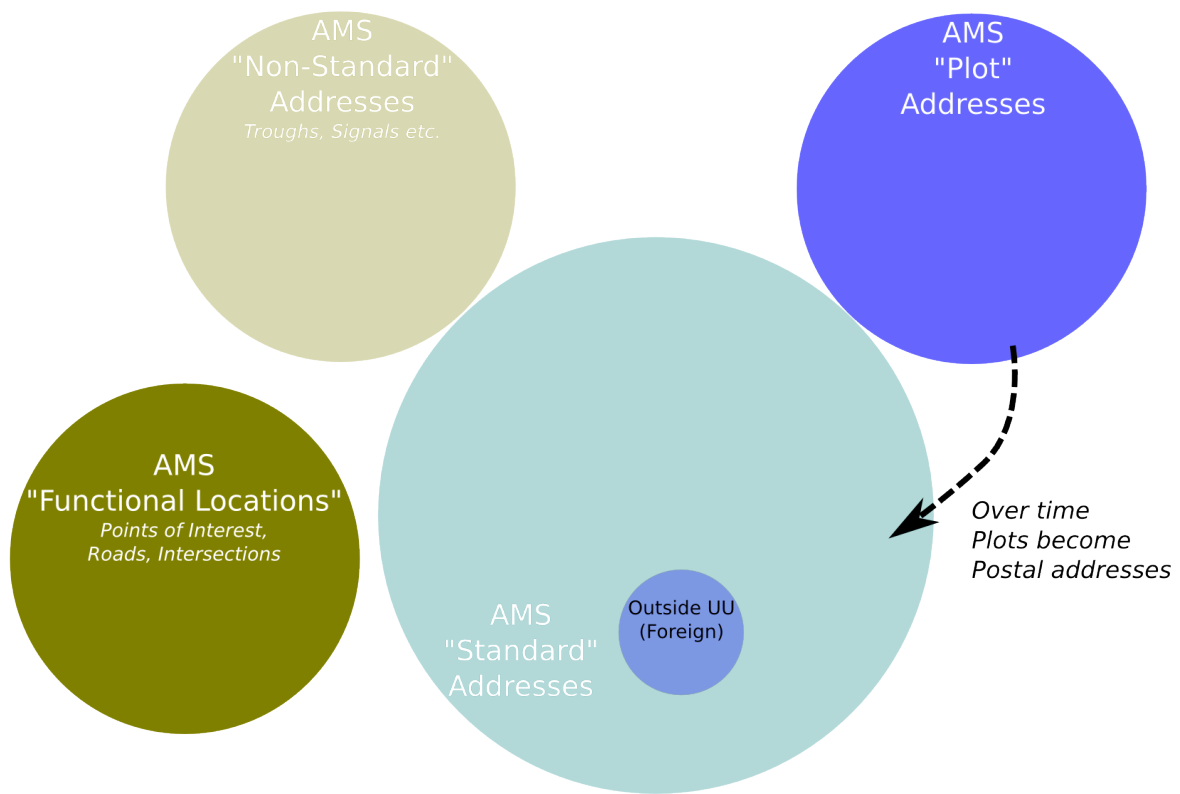


Figure 2: The UU Address Management System

- Some AMS records are related to potential future service points which will be constructed. These represent *plot addresses*. These have an *address_type* of *plot* (PL).
- The AMS record might refer to a service point feeding a cattle trough or a railway signalling box. These represent *non-standard addresses*. These have an *address_type* of *locational* (LO).

At present UU only provides *wholesale* services. When the [Open Water initiative](#) is enacted UU will be split into a *wholesale* and *retail* business. The wholesale part of UU will sell services to a small number of eligible third parties (including the retail part of UU) who will act as retail agents. These eligible retail agents will sell services to individual consumers. The wholesale business will provide transparent systems to all eligible retail businesses. Key to this will be the interoperable exchange of *service point* information. This implies that after Open Water has been enacted there will be both *Retail Service Points* and *Wholesale Service Points*. UU has recognised the important and potentially disruptive impact of Open Water. This recognition underpins the Service Point Address (SPA) project:

In the coming competitive environment, the importance of the service point address will increase. On 2nd January 2014, the OpenWater Group published documentation on how it sees the competitive market operating. A fundamental requirement for market operation is “for registration and switching, a centrally held record of premises, service points, and associated meters and market participants”. The ability of suppliers to interact and automatically match to this data will be critical in running an efficient business.

The current addressing system suffers from poor integration with existing systems including Alto and IWM, poor data quality and is based on an end of life, custom application.

Service point address (SPA) will provide a single version of the truth for property addresses and service point locations within the strategic UUGIS (LAM) application and integrate to other key enterprise applications (IWM, Alto). Our critical success will be ensuring that when dealing with a property which receives a UU service, all of our teams are presented with consistent, accurate information about the property address and location.

Other utility companies in the water domain will be responding to this policy initiative.

3.1 Writing OWL ontologies in ACE

To begin, here are some simple statements that are usually found in ontologies. The ACE representation is contrasted with a “formal language” representation that is normally used in such ontologies. Our work intends to show that natural language, if carefully controlled, provides a more readable and writable ontology representation, without a loss in generality and expressive power.

ACE	Formal language
Every man is a human.	$\text{man} \subseteq \text{human}$
Every human is a male or is a female.	$\text{human} \subseteq \text{male} \cup \text{female}$
John is a student and Mary is a student.	$\{\text{John}, \text{Mary}\} \subseteq \text{student}$
No dog is a cat.	$\text{dog} \subseteq \neg \text{cat}$
Every driver owns a car.	$\text{driver} \subseteq \exists \text{ own car}$
Everything that a goat eats is some grass.	$\text{goat} \subseteq \forall \text{ eat grass}$
John likes Mary.	$\langle \text{John}, \text{Mary} \rangle \in \text{like}$
Everybody who loves somebody likes him/her.	$\text{love}(X, Y) \implies \text{like}(X, Y)$

ACE	Formal language
-----	-----------------

Table 1: example statements represented in both ACE and as formal language

Not all ACE constructions map nicely to OWL. Therefore, we describe only a subset of ACE. This subset does not contain adjectives, adverbs and prepositional phrases; intransitive and ditransitive verbs; queries and imperative constructions; modality and sentence subordination; and negation as failure.

The following OWL constructions are currently not supported by the ACE-to-OWL mapping.

- All meta elements, such as imports, versioning, annotations, etc.
- IRIs and namespaces. One cannot refer to concepts already (partly) defined in other ontologies.
- Data-valued properties are only partly supported, e.g. one can use only integers and strings as data values, and cannot define the domain and range of data-valued properties.

Note that some OWL elements will never be supported, because they are just syntactic sugar and have furthermore no clear ACE counterparts.

The following text is structured to separately discuss the three main building blocks of ontologies as used in OWL: individuals, classes and properties.

3.1.1 Individuals

One can talk about individuals, classify them and describe relations between them. For example,

- There is a man. The man is John.
- His age is 32 and he likes Mary.
- Mary loves herself.
- Bill sees something.
- John is not Bill.

The following graph visualizes the model described by the story, with individuals (denoted by ellipses), their belonging to classes (boxes with labels), properties (solid arrows with labels), equality (dashed line) and difference (red line). Data-values are in a box with dotted border. In case the name of an individual is known then it is written into the ellipses.

OWL does not make a unique name assumption, i.e. an individual can have different names and one has to explicitly state the equivalence or difference of two differently named individuals. Therefore, whether Mary is John or is Bill or is different from both of them, is still open, we only know that John is not Bill. The fact that John belongs to the class *man* is not explicit in the picture, but can be derived by simple reasoning. Note also that John, Mary, Bill and the thing that Bill sees, all belong to the topmost class *owl:Thing*.

One can also classify the individuals by complex class descriptions, e.g.

- John is not a woman. (i.e. *John* belongs to the complement of the class *woman*)
- John is a programmer or is a manager. (i.e. *John* belongs to the union of classes *programmer* and *manager*)

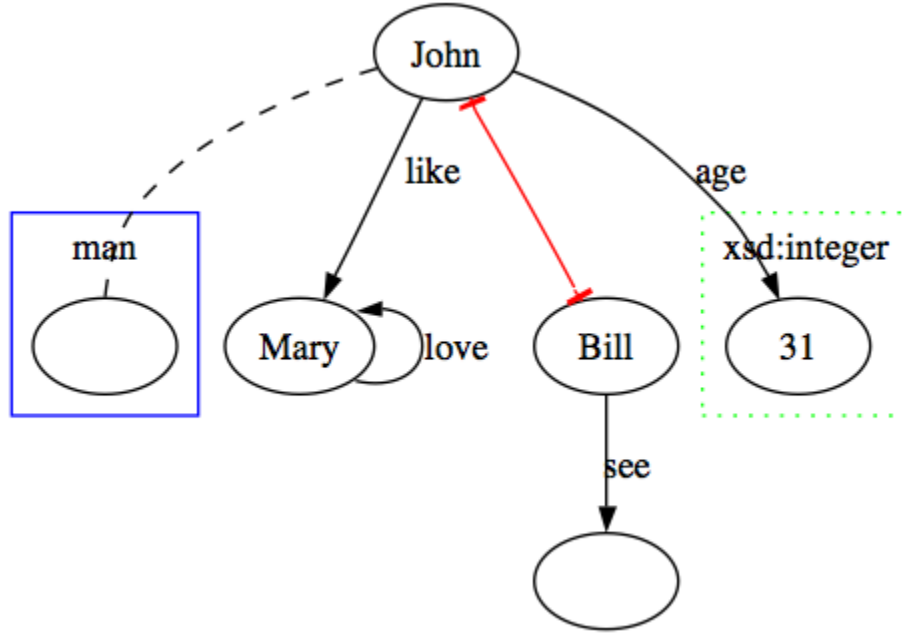


Figure 3: Picture. Representation of individuals as a Venn diagram.

3.1.2 Classes

One can describe classes by specifying relations to other classes. Such relations can be a simple inheritance from a named class (“Every man is a human.”) or more complex property restrictions (“Every driver owns a car.”). In OWL, the classes do not need names, they can be abstract descriptions, e.g. “(Every) man who owns a dog ...”. Such descriptions can be made arbitrarily complex by negation (*no*, *is not*, *does not*), disjunction (*or*, *or that*) and conjunction (*and*, *that*, *and that*).

Both the *every*-sentences (“Every man who has a driving-license drives a car.”) and the *if-then* sentences (“If a man has a driving-license then he drives a car.”) can be used. The *if-then* sentences are more flexible in the way the verbs can refer to existing nouns via anaphoric references (pronouns, variables, and definite noun phrases). However, it is often the case that OWL cannot express this kind of argument sharing between verbs. Therefore, it is a good idea to only use *every*-sentences and avoid all kinds of anaphoric references to nouns occurring in the same sentence.

- Every man is a human. ($\text{man} \subseteq \text{human}$)
- If there is a man then the man is a human. ($\text{man} \subseteq \text{human}$)
- Every professor is a scientist and is a teacher. ($\text{professor} \subseteq \text{scientist} \cap \text{teacher}$)
- Everybody who is not a child is an adult. ($\text{owl:Thing} \cap \neg \text{child} \subseteq \text{adult}$)

Equality of classes has to be spelled out, there is no shorthand construction like OWL’s *EquivalentClasses*.

- Every carnivore is a meat-eater and every meat-eater is a carnivore.

Disjointness of classes can be expressed via ACE’s negation which maps to OWL’s *ComplementOf*.

- Every man is not a woman. ($\text{man} \subseteq \neg \text{woman}$)
- No man is a woman. ($\text{man} \subseteq \neg \text{woman}$)

- Nothing is a man and is a woman. ($\text{owl:Thing} \subseteq \neg (\text{man} \cap \text{woman})$)

Transitive verbs and adjectives introduce the *SomeValuesFrom* restriction.

- Every man who knows a publisher writes a book. ($\text{man} \cap \exists \text{ know publisher} \subseteq \exists \text{ write book}$)
- Every man likes a dog or likes a cat. ($\text{man} \subseteq (\exists \text{ like dog}) \cup (\exists \text{ like cat})$)
- If there is a woman then she does not like a snake. ($\text{woman} \subseteq \neg \exists \text{ like snake}$)

OWL's *AllValuesFrom* constraint e.g. that carnivores eat only meat (if they eat at all), can be expressed by double negation. Let's say we want to express that carnivores and meat-eaters are equivalent concepts (i.e. that $\text{carnivore} = \forall \text{ eat meat}$).

- No carnivore eats something that is not a meat.
- Everybody that doesn't eat something that is not a meat is a carnivore.

There is also a shorter ACES syntax for some patterns of *AllValuesFrom* (i.e. where *AllValuesFrom* is on the right side of the subsumption sign).

- Everything that a carnivore eats is a meat. ($\exists \text{ inv(eat) carnivore} \subseteq \text{meat}$)
- Everything that is eaten by a carnivore is a meat. # the same using passive

Classes can also be described in terms of a property and its cardinality. (Note that the plural of 'thing' will be mapped to *owl:Thing*.)

- Every woman likes more than 2 things. ($\text{woman} \subseteq \geq 3 \text{ like owl:Thing}$)
- Every millionaire owns at least 2 cars. ($\text{millionaire} \subseteq \geq 2 \text{ own car}$)

Reflexive pronouns ('itself', 'himself', 'herself') can be used to describe (ir)reflexivity.

- Every narcissist likes himself.
- No river flows-into itself.

Classes can also be described in terms of a property and its value that is either an individual or data-value. This corresponds to OWL's *HasValue* construct. For individual-valued properties, one must use a transitive verb (or adjective) with an object that is a proper name or a reference to an individual that has already been introduced.

- Every man knows John. ($\text{man} \subseteq \exists \text{ know } \{\text{John}\}$)
- There is a woman who everybody likes. ($w \in \text{woman}; \text{owl:Thing} \subseteq \exists \text{ like } \{w\}$)
- John likes every dog. ($\text{dog} \subseteq \exists \text{ inv(like) } \{\text{John}\}$)
- Every dog is liked by John. # the same using passive

For data-valued properties, ACE makes use of the copula verb 'is', a genitive construction (Saxon Genitive, possessive pronoun or an *of*-construct), a noun which expresses the property, and finally a number or a string. For example

- Everybody whose age is 32 is a grown-up.
- If there is a man who lives-in Paris then the man's address is "Paris".
- For all water it is false that the temperature of the water is -1.

where the genitive construction is expressed by 'whose', 'man's', 'of the water', the property is expressed by 'age', 'address', 'temperature', and the data-value is '32', "Paris" and '-1'.

3.1.3 Properties

OWL properties correspond to ACE's verbs and transitive adjectives (e.g. 'taller than').

A superproperty for a given property can be defined as:

- Everybody who loves somebody likes him/her.

or in a more verbose manner:

- If there is somebody X and X loves somebody Y then X likes Y.

Disjointness of properties can be declared by using negation.

- Nobody who likes somebody hates him/her.

or more verbosely:

- If somebody X likes somebody Y then it is false that X hates Y.

Property chaining is a complex mathematical concept that does not have a simple verbalization in natural language. E.g. the transitivity of a property is formulated in ACE in the following ways (different levels of syntactic compactness are provided):

- If something X is taller than something Y and Y is taller than something Z then X is taller than Z.
- If something X is taller than something that is taller than something Y then X is taller than Y.

To declare an inverse property of a property, two sentences are needed.

- If something X is taller than something Y then Y is shorter than X.
- If something X is shorter than something Y then Y is taller than X.

The rest of the syntax that OWL provides for describing properties is just syntactic sugar, one can use the already described class relations, cardinality constraints and inverse properties instead.

Properties can have a domain and a range. The domain of a property can be expressed e.g. by

- Everybody who writes something is a human.

The range of a property can be expressed by:

- Everything that somebody writes is a book or is a paper.
- Everything that is written by somebody is a book or is a paper. # the same using passive

Properties can be functional and inverse functional. Functionality means that for a given subject, the object of the property (verb) is always the same. Inverse functionality means that for a given object the subject must always be the same. For functionality, one has to restrict the class *owl:Thing* to have at most one value for a given property (it is OK to have no values at all).

- Everybody orders at most one thing. ($\text{owl:Thing} \subseteq \leq 1 \text{ order owl:Thing}$)

Inverse functional properties are expressed by switching the subject and the object of the verb.

- Everything is something that at most one thing orders. (owl:Thing \subseteq \leq 1 inv(order) owl:Thing)
- Everything is ordered by at most one thing. # using passive

Symmetric property is its own inverse. The various ACE syntaxes to express this are:

- If somebody X loves somebody Y then Y loves X.
- Everybody who is loved by somebody loves him/her.
- Everybody who somebody loves loves him/her.

Equivalent properties are expressed by declaring the two properties to be superproperties of each other.

- Everybody who hates somebody despises him/her.
- Everybody who despises somebody hates him/her.

3.1.4 Reasoning

User acceptance of the reasoning results obtained on the OWL level and perceived on the ACE level provides the basis for proving that the mapping from ACE to OWL is useful. Given the ACE text

- John is a man.
- No man is a woman.
- John is a woman.

the reasoner will find an inconsistency because John has been asserted as a man and men have been described as disjoint from women.

Here is another example.

- Every child is a man or is a woman.
- No child is a man.
- No child is a woman.

This text may seem strange, but it is not inconsistent, as far as OWL is concerned. All we have done, is constructed an empty class *child*, which is perfectly legal (one can say, though, that the class *child* is unsatisfiable). Only when we claim that there are elements (individuals) in this class, for instance by stating

- John sees a child.

do we end up being inconsistent.

Reasoning about property restrictions finds inconsistency in:

- Every man sees a dog.
- Nothing is a dog.
- John is a man.

Reasoning about properties of properties. The following text is consistent.

- No dog likes a cat.
- Fido is a dog.
- It loves a cat.

But it becomes inconsistent if we add information about the relation between ‘like’ and ‘love’:

- Everything that loves something likes it.

We can also reason about the domain and range of properties.

- Everything that somebody writes is a book.
- Everybody that writes something is a human.
- John who is a man writes a paper.
- Every man is a human.

Inconsistency is caused when we now claim that:

- No paper is a book.

Reasoning about cardinality, e.g. functional properties. Imagine a library policy saying that everybody can order only one item from the library, and that John places two orders.

- Everybody orders at most one thing.
- John orders a book X and orders a book Y.

This text is consistent, namely because the reasoner sees nothing strange about taking X and Y to denote the same item (individual). Only when we state that:

- X is not Y.

would it become inconsistent.

Reasoning about inverse functional properties is similar.

- Everything is ordered by at most one thing.
- Mary orders a book X.
- Bill orders the book X.
- Mary is not Bill.

Reasoning about transitivity and *HasValue* will find the following text inconsistent.

- If something X follows something that follows something Y then X follows Y.
- John follows Mary who follows Bill who follows John.
- No manager follows Mary.
- Bill is a manager.

Reasoning about inverse properties. In ACE, in many cases, one does not have to explicitly assert a named inverse property of a given property. One can simply use a passive verb to create an (unnamed) inverse property. An inconsistent text, containing an inverse property, can be constructed like this:

- John likes Mary.
- Nobody is liked by John.

This statement would be redundant:

- If somebody X likes somebody Y then Y is liked by X.

Reasoning about nominals (i.e. OWL's *OneOf* construct). The following text is inconsistent.

- Every student is somebody who is John or who is Mary.
- Bill is a student who is not John and who is not Mary.

3.2 Roo Language Elements

3.2.1 Declarations

Rabbit allows an author to explicitly declare **concepts**, **relationships** and **instances**.

Concepts are introduced using the form:

is a concept [, plural].

for example:

River is a concept. (plural defaults to Rivers.)

Sheep is a concept, plural Sheep.

Homonyms are dealt with using brackets. For example a Pool can either be a pool of water on land or a pool of water within a river. These are quite different physical phenomena. The former is more common and so we would introduce it as:

Pool is a concept.

whereas the later would be introduced as:

Pool (River) is a concept.

Where it is unclear which was more common both would use the bracketed form.

In Owl these statements translate to: River -> Thing with the annotation "River" as a label. (Our notation here uses -> to indicate subclass.) Homonyms will all share the same annotation `rdf:label` so Pool and Pool (River) will both be annotated as "Pool" but the class names will be Pool and Pool_River respectively.

Relationships and instances are introduced in a similar way:

is a relationship.

is a .

For example:

next to is a relationship.

Derwent Water is a Lake.

3.2.1.1 Concept Descriptions, Definitions and Axioms A concept is described (necessary conditions) or defined (necessary and sufficient conditions) by a collection of axioms relating to that concept. Each axiom is represented by a single Rabbit sentence. The simplest sentence form is where one concept is associated with another through a simple relationship.

For example:

Every is a kind of . (Subsumption.)

Every .

E.g.

Every House is a kind of Building.

Every River contains Water.

Such statements translate to OWL as follows:

House -> Building.

River -> contains some Water.

A number of modifiers exist to enable cardinality and lists to be supported.

For example:

Every River Stretch has part at most two Confluences.

Every River flows into exactly one of River, Lake or Sea.

Concept descriptions comprise a series of axioms relating to the same concept. A definition comprises a group of axioms that make up the necessary and sufficient conditions. The axioms are grouped by an introductory statement and then follow as a list:

A School is defined as:

Every School is a kind of Place;

Every School has purpose Education;

Every School has part a Building that has purpose Education.

Which translates into OWL as:

School Place and (hasPurpose some Education) and (hasPart some (Building and hasPurpose some Education))

Rabbit at present contains a sentence form that cannot be translated into OWL, although its omission does not invalidate the OWL ontology it does make the OWL representation less complete. This is the ability of Rabbit to modify an axiom by adding “usually”. For example:

A Postal Address usually contains a Road Name.

As OWL is unable to express the existential quantifier for elements on the left hand side of an expression, this sentence cannot be converted to OWL. The reason for including it is to enable the domain expert to record frequent but not mandatory relationships, the absence of which would make certain definitions seem strange. Indeed without the use of “Usually” the only things that could be said about a British Postal address are:

A Postal Address contains exactly 1 Post Town Name.

A Postal Address contains exactly 1 Postcode.

3.2.2 Intersection and Union

Rabbit implements intersection in a number of different ways in order to promote the development of short sentences, rather than encourage the authoring of long sentence structures that could become hard to understand. Most common amongst these mechanisms is that all Rabbit sentences that refer to the same concept are converted to OWL as one long conjunction comprising all the Rabbit sentences. Within a sentence, “and” is supported, but in practice has rarely been used. Rabbit also supports “that” which is interpreted exactly the same as “and”.

For example:

Every Almshouse has part a Building that has purpose Housing of Elderly People.

Here we encourage the use of “that” rather than “and” as it both sounds better, and we also believe it will discourage long chains of conjunctions that would be better treated as separate Rabbit sentences.

Another mechanism used to implement intersection is the use of “of”, “for” and “by” (again all semantically equivalent in their OWL translation). They are used in the sentences of the structure:

Every [of | for | by] .

e.g.

Every School has purpose Education of Children.

this translates into OWL as:

School -> hasPurpose some (Education and appliesTo some Child)

Here “of”, “for” and “by” are translated to “and appliesTo” in OWL with appliesTo being a predefined Rabbit relationship.

“Or” is supported in both inclusive and exclusion forms. By default Rabbit treats “or” (and “,”) as exclusive. So:

Every River flows into a Sea or Lake.

Is interpreted in OWL as:

River -> flowsInto some ((River or Lake) and not (River and Lake)).

However, we found that the exclusive or was very rarely used in ontology modelling, and where it did appear, usually indicated some mistake in the content of our model. In the above example, we would more accurately designate “flows into” as a functional property, and hence not need to consider the exclusive or.

Inclusive Or is implemented through the use of the modifier “One or more of”:

Every Mission has purpose one or more of Christian Worship or Charitable Activities.

which in OWL is:

Mission -> hasPurpose some (ChristianWorship or CharitableActivity).

3.2.3 Ontology Referencing

No ontology can be an island unto itself: the point of ontologies is interoperability, and therefore they need mechanisms to include concepts and relationships from other ontologies.

OWL achieves this through the owl:import statement, although since it operates at the level of the whole ontology, it is a fairly crude approach.

The equivalent Rabbit construct is:

Use ontologies: from ; ... from .

e.g.

Use ontologies:

Wildfowl from <http://www.ordnancesurvey.co.uk/ontology/v1/Wildfowl.rbt>

Transport from <http://www.ordnancesurvey.co.uk/ontology/v1/Transport.rbt>

Concepts and relationships taken from these ontologies are then identified using the notation:

[]

for example

Every Duck Pond is a kind of Pond.

Every Duck Pond contains Ducks [Wildfowl].

This indicates that the concept Duck is obtained from the Wildfowl ontology. Since repeatedly referencing Ducks [Wildfowl] can be a bit cumbersome, Rabbit also enables a local name to be applied:

Refer to Duck [Wildfowl] as Duck.

This produces the following OWL:

?Duck -> Thing

Duck))

As this creates a new local concept, it is then possible to extend the definition of the imported concept if required by adding further axioms to the local concept.

3.2.4 Property Restrictions - An Area of Weakness

All CNLs appear to struggle when it comes to implementing property characteristics. such as symmetry or transitivity. Fundamentally this is because these constructs are not well aligned to the way that people normally think about things and hence there are no easily understood natural language analogues. Probably the worst example is transitivity. In Rabbit such characteristics are defined as follows:

The relationship is transitive.

e.g.

The relationship “is part of” is transitive.

However as discussed below, human subject testing has shown that this is very poorly understood by people, as are similar constructions. ACE and SOS have both taken a different approach. They attempt to define such relationships through example. SOS express the transitive relationship as follows:

If X Y and Y Z then X Z.

If X is part of Y and Y is part of Z then X is part of Z.

It may well be that such expressions are more understandable and we are planning to include such sentences in the next phase of human subject testing. However, we strongly suspect that such structures will still present domain experts with significant problems with interpretation, and take much longer to input when faced with authoring a large ontology. We are prepared to admit that there is no good solution within the language and so will have to be managed through training, tool support and guidance by the knowledge engineer.

Bibliography

Anon., 2014. Attempto controlled english. *Wikipedia, the free encyclopedia*. Available at: http://en.wikipedia.org/w/index.php?title=Attempto_Controlled_English&oldid=620928248 [Accessed September 17, 2014].

Denaux, R., 2013. *Intuitive ontology authoring using controlled natural language*. PhD thesis. University of Leeds. Available at: <http://etheses.whiterose.ac.uk/4446/> [Accessed September 17, 2014].

Hart, G., Johnson, M. & Dolbear, C., 2008. Rabbit: Developing a control natural language for authoring ontologies. In S. Bechhofer et al., eds. *The semantic web: Research and applications*. Lecture notes in computer science. Springer Berlin Heidelberg, pp. 348–360. Available at: http://link.springer.com/chapter/10.1007/978-3-540-68234-9_27 [Accessed September 17, 2014].