

THE RABBIT LANGUAGE: DESCRIPTION, SYNTAX AND CONVERSION TO OWL



Document Information

This document has 28 page(s)

This document describes the Rabbit language which is used to author conceptual ontologies, including its syntax, grammar and how to convert the structures to OWL.

The authors are Cathy Dolbear, Glen Hart, Katalin Kovacs, John Goodwin and Sheng Zhou

Ordnance Survey and the OS Symbol are registered trademarks of Ordnance Survey, the national mapping agency of Great Britain.

Table of Contents

1	Introduction	3
2	Related research.....	3
3	Rabbit – motivation and design principles	6
	3.1 Principle 1: Expression	6
	3.2 Principle 2: Grammar	7
	3.3 Principle 3: Interaction between domain expert and knowledge engineer	8
	3.4 Principle 4: Tool support.....	9
	3.5 Principle 5: Domain independence and reuse	10
	3.5.2 Concept level referencing	10
	3.5.3 Concept and sentence inclusion	11
4	References	12
A	Rabbit sentence structure.....	14
	A.1.1 Assumptions made in Rabbit.....	14
	A.1.2 Additional Rules and Notes	15
B	Conversion Examples	16
	B.1 Productive sentences	16
	B.1.1 Declarations	16
	B.1.2 Concepts	17
	B.1.3 Relationships	19
	B.1.4 Axioms.....	19
	B.1.5 Instances.....	20
	B.2 Receptive sentences.....	21
	B.2.1 Concepts	21
	B.2.2 Relationships	21
C	Backus–Naur Form Grammar	24

1 Introduction

This document describes the reasoning for the design of the Rabbit controlled English grammar that is an integral part of building domain ontologies. It is one of five documents outlining the Ordnance Survey ontology authoring method.

There are four further documents that outline the Ordnance Survey method for building ontologies, namely:

- Domain Ontology Development [Hart et al 2006]
- A Methodology for Building Conceptual Domain Ontologies [Kovacs et al 2006]
- A Methodology for Converting Conceptual Domain Ontologies to OWL [Goodwin 2007]
- Modelling Guidelines for Constructing domain ontologies [Hart and Goodwin 2007]

This document describes the first version of the Rabbit controlled natural language - our research and collaboration with other groups researching this area is ongoing, and following human subject testing, the grammar is likely to be updated.

2 Related research

Ever since OWL was conceived there have been concerns that its form makes it inaccessible to all but those with a good understanding of mathematics [Horridge et al 2006]. It is therefore difficult for it to be used by domain experts to author, validate or reuse ontologies. This in turn creates a serious impediment to the adoption of OWL and semantic web technologies in general since there are far too few people with both domain knowledge and the knowledge to be able to use languages such as OWL in a competent and reliable manner. There have been a number of attempts to resolve this issue through the creation of grammars for OWL that attempt to make it more understandable. Such grammars include the Manchester Syntax [Horridge et al 2006] which attempts to replace the abstract symbology of description logic.

For example the statement:

$$\begin{aligned} \text{River} \sqsubseteq & \text{BodyOfWater} \sqcap \exists \text{flowsIn.Channel} \sqcap \exists \text{hasCurrent.Current} \sqcap \\ & \exists \text{hasDirectComponent.Source} \sqcap ((\exists \text{hasDirectComponent.RiverMouth} \sqcap \exists \text{flowsInto}.\text{(Sea} \sqcup \\ & \text{Lake} \sqcup \text{Reservoir)}) \sqcup (\exists \text{hasDirectComponent.Confluence} \sqcap \exists \text{flowsInto.River})) \sqcap \\ & \forall \text{flowsInto}.\text{(River} \sqcup \text{Sea} \sqcup \text{Lake} \sqcup \text{Reservoir}) \end{aligned}$$

is represented in the Manchester Syntax as:

Class: River

subClassOf:

BodyOfWater *and*

flowsIn **some** Chanel *and*

hasCurrent **some** Current *and*

hasDirectComponent **some** Source *and*
((hasDirectComponent **some** RiverMouth *and* flowsInto **some** (Sea *or* Lake *or* Reservoir))
or
(hasDirectComponent **some** Confluence *and* flowsInto **some** (River))
and
flowsInto **only** (River *or* Sea *or* Lake *or* Reservoir)

disjointWith:

RiverMouth

Bay

Pond

Lake

....

Whilst this is significantly more readable than the purely Description Logics representation, the average domain expert will still struggle to understand what it means.

Other approaches are to use constrained forms of English, examples being ACE [Fuchs et al 2005] and Processable English (PENG) [Schwitter et al 2002] both of which provide grammars based on constrained English to represent First Order Logic (FOL) and both have now Description Logic (DL) subsets [Kaljurand and Fuchs 2006] and [Schwitter 2006], the PENG DL version being recently dubbed the “Sydney Syntax”. These do provide significantly more readable representations. For example simple statements such as “France is a country.” can be made. As these grammars are representations of OWL they also allow more complex statements to be made such as “France is a country and Paris is a city.” In our view, this would be easier for the domain expert to digest if it were expressed as two separate statements. The more complex forms also begin to sound a bit unnatural but are still readable: “If X has Y as a topping then X has Y as an ingredient and X is a pizza and Y is a pizza topping and Y is a topping of X.”

All these approaches are limited to representing only what can be stated in DL (or FOL). They will also reflect any optimisations or modelling “tricks” that an ontology engineer might apply and which whilst necessary for either efficient reasoning or accurate modelling (given the OWL language constraints) will obscure the ontology from a domain expert’s point of view. An example of one such “trick” would be as follows: in order to express the sentence “Some Aqueducts contain water” we would create a subclass of the concept Aqueduct, called SomeAqueduct, and make it equivalent to “Aqueduct and contains some Water”. (See Appendix B for details.)

Lastly it is worth mentioning approaches that take OWL and then produce English language representations such as the DL to Natural Language converter that has been developed as part of SWOOP [Kalyanpur et al]. Whilst these do not allow a domain expert to write in an English-like syntax. they do at least enable the domain expert to confirm that what was written is true. So for example:

```
Class (MediumPizza partial
Pizza
restriction(hasTopping maxCardinality(5))
restriction(hasTopping minCardinality(3))))
```

Would be converted to the perfectly readable: "MediumPizza is a Pizza which has between 3 – 5 topping".

However it can also produce outputs that are quite difficult to understand [Kalyanpur, et al]

```
Ontology: Food – http://www.w3.org/2001/sw/WebOnt/guide-src/food.owl
Class (ShellfishCourse partial
restriction(hasDrink allValuesFrom(restriction(hasBody value (Full)))) restriction(hasDrink
allValuesFrom( restriction(hasFlavor allValuesFrom( oneOf(Strong Moderate))))))
)
)
```

Which results in:

ShellfishCourse is a Meal Course that (if has drink) always has drink Potable Liquid that has Full body and which either has Moderate or Strong flavor

Note: The phrase 'Potable Liquid' in the above output is obtained from the range of the property hasDrink replacing the default keyword 'Thing'.

This is because it can only reflect the terminology and modelling approaches used by the author and which can obscure the meaning or make the representation unnatural.

All these initiatives tend to be developed by the Description Logics community and often do not involve the intended end-users. Moreover, they are a merely a way of making DL more readable and even the good ones can still be a bit stilted or allow over-complexity. Our belief is that the authoring process is one where the domain expert is central and therefore any representational language needs to be integrated into an understanding of the authoring and modelling process. We also believe that the development of any syntax must start from the needs of the domain expert rather than the language constraints of OWL.

In summary, we see the advantages of Rabbit over other methods and DL syntaxes as being:

- Greater clarity of expression due to the involvement of the domain expert both in the development of Rabbit, its authoring and use.
- The ability to hide certain modelling complexities and optimisations.
- The encouragement, through its grammar, to produce short sentences that are more easily interpretable.

3 Rabbit – motivation and design principles

Our research [Hart et al] has been focused on developing a language that overcomes some of the limitations described above: namely, it should be easily readable and writable by domain experts; easy for them to digest, and allow them to express what they need to in order to describe their domain, rather than be limited just to OWL constructs. We have named this language Rabbit, after Rabbit in Winnie the Pooh, who was really cleverer than Owl. To this end, we have involved domain experts from the outset in the core language design decisions. It is a language in its own right, rather than merely a syntactic veneer on OWL. This means that it contains constructs such as “usually” (meaning an unquantified but significant majority but not all) that are needed by the domain expert, even though they cannot be expressed in OWL. These constructs are included to ensure that domain knowledge is not lost, even if it cannot be fully exploited.

The fundamental principles underlying the design of Rabbit are:

1. To allow the domain expert to express their knowledge as easily and simply as possible and as in much detail as necessary.
2. To have a well defined grammar and be sufficiently formal to enable those aspects that can be expressed as OWL to be systematically translatable;
3. To recognise that the domain expert alone cannot produce an ontology and that a knowledge engineer is also necessary;
4. To be used in conjunction with tools which help to enforce an authoring method but not to the point where Rabbit is only readable through tools;
5. To be independent of any specific domain and encourage reuse of knowledge in new domains.

3.1 Principle 1: Expression

The first principle means that Rabbit encourages the use of simple short statements and discourages complex expressions (this means that several Rabbit statements may together result in a complex OWL expression). Rabbit also focuses on keeping the sentences natural sounding (for example using “A River flows into a Sea.” rather than the OWL-like “All Rivers flow into some Seas”). To make it easier for the domain expert to produce, several defaults are assumed. As well as the abovementioned sentence-position dependent understanding of the indefinite article “a”, the conjunction “or” is assumed to be an exclusive or, and all concepts are assumed to be disjoint, unless specified otherwise. As domain experts tend to think in terms of the closed world assumption, these defaults are more natural for them. However, it means that the open world assumptions of OWL have to be explicitly closed down when such Rabbit sentences are converted to OWL. This leads to relatively simple expressions in Rabbit having complex expression in OWL. For example the Rabbit statement:

A River flows into a Sea or a Lake or a River or a Reservoir.

would result in the following OWL:

River -> flowsInto some (Sea or Lake or River or Reservoir) and not (flowsInto some Sea and flowsInto some Lake) and not (flowsInto some Lake and flowsInto some River) and not (flowsInto some River and flowsInto some Reservoir) and not (flowsInto some Reservoir and flowsInto some Sea) and not (flowsInto some Sea and flowsInto some River) and not (flowsInto some Lake and flowsInto some Reservoir)

Use of some constructs such as “usually” can be represented in Rabbit but cannot be represented in OWL. They are included in Rabbit to ensure that the domain can be captured as accurately as possible and to ensure that a better quantification of information loss can be made. Therefore a statement such as:

A River Channel usually contains Water.

would be imperfectly translated as

RiverChannel contains Some Water.

The latter statement is of course inaccurate but the best that can be achieved given that OWL cannot represent probabilities whether quantified or, as in this case, unquantified.

3.2 Principle 2: Grammar

Concepts in Rabbit may comprise single or a number of linked words such as “River” or “River Stretch”. Homonyms (words that have the same spelling but different meaning) are differentiated with the addition of a disambiguation term following the concept name in brackets: Pool (River). This is converted to OWL as Pool_River and an rdf:label of “Pool”.

Simple statements about concepts are written in the singular as this mimics the way domain experts often talk about concepts. For example “A River flows into a Sea.” Rabbit has a few predefined relationships (OWL properties) such as “is a kind of” to introduce super and sub-class relationship. Mostly authors will define their own relationships such as “flows into”. Relationships may be modified using phrases such as “only”, “usually”, “at least” and “does not”. For example, “A Braided River Stretch flows in at least 2 Channels.” Rabbit enables lists, and “or” and “and” where “or” is treated as mutually exclusive and “and” as inclusive.

Simple statements such <noun phrase><verb phrase><noun phrase> such as “A River flows into a Sea.” is converted to:

River -> flowsInto Some Sea (expressed using the “Manchester Syntax”).

Should it be that rivers can only flow into seas then the Rabbit would be:

A River only flows into a Sea.

and the OWL:

River -> flowsInto some Sea and flowsInto only Sea.

A defined concept is one that uniquely specifies the concept such that all things that have the defining properties of that concept are that concept. For example if the defining property of mammals is that they suckle their young then all animals that suckle their young must be mammals. Conversely, an additional property of mammals is that they are warm blooded. But whereas all mammals must be warm-blooded other animals such as birds may also be warm-blooded and so warm-bloodedness is not a defining property of mammals. Defined concepts are described using the general form: <concept> is uniquely defined as: <sentence>; <sentence>; ... <sentence>.

Whereas elsewhere all Rabbit sentences are terminated by a full stop “.” When specifying the defining properties all but the last sentence are terminated by semi-colons as one would do in English when constructing a list.

An example being:

A Source is uniquely defined as:
a kind of Spring or Wetland;
feeds a River or a Stream.

And this translates into OWL as:

Source \equiv (Spring or Wetland) and feeds some (River or Stream).

Note that in Rabbit the “Ors” bind together Spring and Wetland so the restriction that they must flow into a river or stream applies to both. If instead only the wetland that had to flow into a river or stream then it would be necessary to make this explicit using brackets. For more details about the allowed constructs and their conversion to OWL, see appendix B.

3.3 Principle 3: Interaction between domain expert and knowledge engineer

An important aspect worthy of a little more discussion is the interaction between the domain expert, the knowledge engineer and the translation process in “compiling” Rabbit into OWL. First, we accept that domain experts will require the assistance of a knowledge engineer to most accurately represent the domain. However, we also feel it is important that the domain expert is in control of the process. Translating Rabbit to OWL will have three significant effects on the modelling process. First, the process is likely to expose flaws in modelling which will need correction in Rabbit. Second, it gives the potential to include optimisations in the OWL version (for reasoner efficiency) which need not be reflected in the Rabbit representation (lest they obscure the clarity of the ontology). Third, the knowledge engineer will point out

additional information that is needed for the logical ontology, to make the domain expert's knowledge more explicit. These will need to be confirmed by the domain expert but may not necessarily be explicitly represented in Rabbit but expressed as "modelling tricks" in OWL. An example would be the representation of transitive properties (see discussion below).

Next, this brief and very incomplete tour of the Rabbit grammar will touch upon transitive relationships. These are interesting, because if we took "has part" as an example, we could say that "A car has part an Engine" and that "An engine has part a Piston" and so on. Normally if we asked what were the parts of a car, then we would expect a list of the major components. However, if "has part" is defined in OWL as transitive, then any self-respecting reasoner would return all parts. This is probably not what most people would expect. This behaviour has been recognised as an issue and solutions have been proposed. One solution is to define the "hasPart" property as transitive and then to define a subproperty "hasDirectPart" which is not transitive. This then enables a choice to be made at query time as to whether all parts or just "direct parts" are selected. Rabbit has adopted this convention in that all transitive relationships in Rabbit translate to OWL where a property and subproperty pair are defined. Future references in Rabbit will then always result in the non-transitive version being used. This ensures that the typical expectations of non-transitive reasoning are fulfilled but still enabling transitive reasoning when required. Introducing this trick within OWL potentially introduces a problem in that the most frequently used form: hasDirectPart is not a term introduced by the domain expert. We overcome this by using the annotation property to label the term "has part" and we annotate the transitive super-property as "has all parts".

3.4 Principle 4: Tool support

We are currently working with the University of Leeds to develop a software tool to assist the domain expert with authoring ontologies according to our method, and producing Rabbit sentences to describe the concepts in their domain. The constrained nature of the Rabbit grammar means that we can avoid the complexity of full natural language sentence parsing. It also allows Rabbit structures to be automatically generated by the software. For example, declaration sentences: "Concepts are: River, Stream, Lake" or "Relationships are: flows into, part of, connects to". During the process of automatically converting the Rabbit sentences to OWL, we want the software tool to assign the Rabbit sentences as annotation properties of the relevant concepts, thus acting as documentation of the OWL.

We also differentiate between productive and receptive sentences in Rabbit. While we expect the domain expert to author "productive" sentences fairly easily themselves, "receptive" sentences are used as a feedback mechanism by the software tool or the knowledge engineer, to check that the knowledge has been correctly captured. This is used particularly for the more complex expressions in OWL. For example, while the domain expert may state: "flows into is irreflexive", when they use it in the ontology, for example stating "A River flows into a Sea.", the receptive sentence would be displayed as a prompt to them, saying: "a River cannot flow into itself" – thus explaining the meaning of irreflexivity. However, while tools can support the development of a Rabbit ontology in this way, Rabbit is designed such that it is independent of any one particular tool.

3.5 Principle 5: Domain independence and reuse

Rabbit has been developed over the course of authoring an ontology in the domain of hydrology and has been further tested with the buildings and administrative geography domains. We believe that it now contains most of the necessary constructs to author ontologies in any domain, including expressions beyond OWL or OWL 1.1.

Rabbit supports two methods for reusing the content of other Rabbit ontologies:

- Concept level referencing;
- Concept and sentence inclusion.

Irrespective of the method used references to other ontologies are introduced in the fashion of document references using the structure:

Use references:

<ref> from <URL>;

.

.

<ref> from <URL>.

E.g.

Use references:

OSHydro from <http://www.ordnancesurvey.co.uk/ontology/Hydrology.rbt>;

OSPlaces from <http://www.ordnancesurvey.co.uk/ontology/Places.rbt>;

OSBuildings from <http://www.ordnancesurvey.co.uk/ontology/Buildings.rbt>.

The <ref> name is used as the short hand to refer to the ontology elsewhere in the ontology where it is enclosed in square brackets e.g. [OSPlaces].

3.5.2 Concept level referencing

Concept level referencing enables the authored ontology to reference individual concepts. In doing so it limits the reference to that concept only. As an example if the referenced ontology contains the concepts a, b and c that have the following relationships:

a is part of b.

b is part of c.

If the authored ontology references concept a it sees the relationship a is part of b but treats the concept b as an undefined stub concept, with no reference to c.

Referenced concepts are introduced by using the Rabbit sentence to declare a concept but with the addition of the ontology reference e.g.

Pub is a concept [OSPlaces].

If the concept name clashes with a name already used within the ontology it is possible to give the referenced concept a local name:

Refer to Pub [OSPlaces] as Public House.

Lastly the definition can be extended simply by adding additional sentences. For example the definition of Pub can be extended by adding the sentence:

A Pub has a Taxable Value.

3.5.3 Concept and sentence inclusion

Whereas the previous methods work by reference – the concepts are accessed live as they are needed – this method physically includes concepts and sentences relating to the concept definitions into the authored ontology. This has three advantages. Firstly, if the reference ontology changes, it does not affect the current ontology and its links to the reused concepts. Secondly, it makes the domain expert check every sentence as it is copied to make sure it is really relevant to the current domain, so that the ontology being built doesn't end up containing information outside its scope, or worse, inaccurate from the current perspective. Thirdly, it enables sentences within a concepts definition that are not necessary to the authored ontology to be omitted.

A reused sentence is indicated using the reference label [<ref>] to precede the sentence.

[OSHydro] A River flows into a Sea or a Lake or a River.

This implies that every concept and relationship in this sentence comes from the OSHydro ontology. Note that it is the sentence, rather than the concept that is being reused and copied. So you never need to reference a concept alone, but a whole sentence.

A more complete example might be where it is recognised that a concept from one ontology is the same as that in the authored ontology although not all the properties are appropriate for this domain and this domain wishes to add more as well. For example if a topographic ontology defines a building this may well be reused in another ontology say a taxation ontology.

Topo Ontology:

A Building is a concept.

A Building is uniquely defined as:

A Building is a kind of Roofed Permanent Structure;

A Building provides Shelter.

A Building has a Height.

A Building has a Footprint.

From the perspective of the Valuation Office that taxes buildings they recognise when authoring their ontology that the topographic definition is reusable by them although they are not interested in building heights and want to say something about the taxable value. So they would write in their ontology:

Use references:

OSBuildings from <http://www.ordnancesurvey.co.uk/ontology/Buildings.rbt>.

[OSBuildings] A Building is a concept.

A Building is uniquely defined as:

[OSBuildings] A Building is a kind of Roofed Permanent Structure;

[OSBuildings] A Building provides Shelter.

[OSBuildings] A Building has a Footprint.

A Building has a Taxable Value.

After the new Rabbit ontology has been built using part of a previous ontology, it can then be compiled into OWL as one file. This is a different approach to previous work whereby it is the OWL ontology that is reused via importing the entire ontology (or segmenting the ontology to create a smaller module that has to be imported in its entirety) to create a second OWL ontology. By forcing the domain expert to select each sentence that really needs to be reused, we avoid the need for inaccurate ontology segmentation or bulk importing of axioms unnecessarily. We would like to see tool support in the future for reuse of Rabbit, assisting domain experts to make these reuse and merging decisions.

4 References

[Fuchs et al 2005] Fuchs, N.E., S. Höfler, K. Kaljurand, F. Rinaldi, and G. Schneider. Attempto Controlled English: A Knowledge Representation Language Readable by Humans and Machines. In Norbert Eisinger and Jan Ma'luszzyński, editors, Reasoning Web, First International Summer School 2005, Msida, Malta, July 25 29, 2005, Revised Lectures, number 3564 in Lecture Notes in Computer Science. Springer, 2005

[Goodwin 2007] Goodwin, J. A Methodology for converting conceptual domain ontologies to OWL Ordnance Survey Technical Report IRI-0005 2007

[Hart and Goodwin 2007] Hart G. and Goodwin, J. Modelling Guidelines for constructing domain ontologies Ordnance Survey Technical Report IRI-0006 2007

[Hart et al 2006] Glen Hart, Catherine Dolbear, Katalin Kovacs, John Goodwin, Sheng Zhou, Using Structured English to Author a Topographic Hydrology Ontology, GISRUK 2007.

[Horridge et al 2006] Horridge M., Drummond, N, Goodwin, J, Rector, A, Stevens, R, Wang, H, The Manchester OWL Syntax. OWL Experiences and Directions Workshop, Athens, Georgia November 2006

[Kaljurand and Fuchs 2006] Kaljurand K. and N. E. Fuchs. Mapping Attempto Controlled English to OWL DL. In 3rd European Semantic Web Conference. Demo and Poster Session, Budva, Montenegro, June 12th 2006.

[Kalyanpur, et al] Kalyanpur A., C. Halaschek-Wiener,.V. Kolovski, J. Hendler, Effective NL paraphrasing of ontologies on the semantic web (Technical Report). URL <http://www.mindswap.org/papers/nlpowl.pdf>

[Kovacs et al 2006] A Methodology for Building Conceptual Domain Ontologies Katalin Kovacs, Catherine Dolbear, Glen Hart, John Goodwin and Hayley Mizen Ordnance Survey Research Labs Technical Report IRI-0002 2006

[Schwitter et al 2006] Schwitter R., M. Tilbrook, Let's Talk in Description Logic via Controlled Natural Language, in: Proceedings of the Third International Workshop on Logic and Engineering of Natural Language Semantics (LENLS2006) in Conjunction with the 20th Annual Conference of the Japanese Society for Artificial Intelligence, Tokyo, Japan, June 5-6, pp. 193-207, 2006.)

[Schwitter. 2002] Schwitter, R. English as a formal Specification Language. Proceedings of the Thirteenth International Workshop on Database and Expert Systems Application (DEXA 2002), pp. 228-232

A Rabbit sentence structure

This assumes that the relationships and concepts have all been declared first, (or a GUI is used to prompt the domain expert to identify which parts of the sentence are concepts and which relationships) so there is no need to split up parts of relationship or concept terms. The complex phrases are therefore based on what are already known to be concepts or relationships, and can be iterative.

Sentences are either productive: that the domain expert is meant to understand easily enough to author themselves, or receptive: sentences that the domain expert would not be expected to author themselves, but would be produced automatically by a software tool, or used by the knowledge engineer as a feedback mechanism to check that the domain expert has understood the exact meaning of what they are stating. For example, the domain expert might state the productive sentence: “Flows into is irreflexive.” and then the receptive sentence would be used to check that they had really understood the meaning of irreflexivity: “A river cannot flow into itself.”.

Please note that some sentences in appendix B do not contain accurate content – they have been chosen to each demonstrate one Rabbit structure clearly, whereas in reality ontology sentences will often need to combine several structures. For example, *all values from* may be combined with *exclusive or*.

Authors may place free text comments into Rabbit using the notation (Note: <comment>). Such annotation may be used to add metadata such as author, provenance etc.

A.1.1 Assumptions made in Rabbit

- Plural concepts add an ‘s’ as default (otherwise the appropriate sentence must be written)
- Plural relationships – remove an ‘s’ as default (otherwise the appropriate sentence must be written)
- Plurals – phrases written in the singular imply the plural, except for materials (e.g. water, rock) (if it applies to one, it applies to all).
- “Or” – in the conceptual ontology, the use of the term “or” in a collection means an exclusive or (this or that, but not both) unless stated otherwise.
- Disjoint – all named concepts in the conceptual ontology are disjoint unless told otherwise.
- Sentences describing the same concept are implicitly concatenated together with AND in the OWL translation.
- The indefinite article will be “an” in cases where the noun phrase begins with a vowel
- Properties are by default object properties, otherwise they must be specified as datatype properties using the structure: “has age is a relationship that take a value as an object”.

- Transitive properties are assigned a non-transitive subproperty in OWL, auto-labelled “direct<name-of-property>” or for “has “ properties “hasDirect<name-of-property>”. This does not need to be expressed in the Rabbit, but is a modelling default in OWL. The “direct” subproperty is used as the default internally to OWL.
- AND takes precedence over OR by default. For example, “A Windmill is associated with Grinding Grain and Crushing Grain, or Draining Water or Generating Electricity.” means A Windmill is associated with Grinding Grain and Crushing Grain, or is associated with Draining Water, or is associated with Generating Electricity. If we need to indicate that “A has relation to B and (C or D)” these would be written as two separate Rabbit sentences: “A has relation to B” and “A has relation to C or D”

A.1.2 Additional Rules and Notes

- No punctuation should be used, with the following exceptions:
 - A full stop must be used to denote the end of a sentence unless the sentence is all but the last clause when specifying a defined concept or set of references.
 - Commas are used in the sentence structure “Tributary is a concept, plural tributaries.” And in lists or collections: “a sea, a river, or a lake”
 - Semi colon and colon are used in definition or reference sentence structure, for example “A source is uniquely defined as: A source is a kind of spring or a wetland; A source feeds a river or a stream.”
 - Round Brackets are used only to for disambiguation of terms, for example Spring (Season) and Spring (Water) and for disambiguation of and/or conjunctions.
 - Square brackets are only used for denoting references (provenance of sentences reused from other ontologies).
- No adverbs other than “usually” are allowed, for example “temporarily” or “mostly”. Alternatives should be sought which move the information into the relationship. For example, instead of saying “A field is mostly bounded by a hedge or a fence.” We would say “A field is partially bounded by a hedge or a fence.” and “partially bounded is a relationship.” The reason for this is not only that we couldn’t express the adverbs in OWL anyway, but also, we want to make it clearer exactly what we mean by mostly.
- Prepositions are not separate Rabbit constructs and should be modelled within relationships. For example, instead of saying “An Underground Car Park is associated with Parking Cars Under Ground.” We would say “An Underground Car Park is associated with Parking Cars.” And “An Underground Car Park is located Under Ground.

B Conversion Examples

B.1 Productive sentences

Conceptual aspect	Example	Logical structure	Notes
B.1.1 Declarations			
<noun phrase> is a concept.	River Stretch is a concept. Pool (river) is a concept.	Class:RiverStretch rdf:label "River Stretch" Class:Pool_River rdf:label "Pool (River)" rdf:label "Pool"	All concepts have a label with the same string as their concept name. Ones with brackets used for term disambiguation have the commonly used terms as well.
<verb phrase> is a relationship.	Flows into is a relationship. A River flows into a Sea. (where River and Sea are concepts) Has name is a relationship that takes a value as an object.	ObjectProperty: flowsInto DatatypeProperty: hasName	Object Property (default) Datatype property
<noun phrase> is an instance of a <concept >.	England is an instance of a country.	<Country rdf:ID="#england"/>	
A <concept> is a kind of <concept>.	A Bourne is a kind of Stream.	Bourne subClassOf Stream	(Bourne -> Stream in the rest of this notation)
<noun phrase> is a concept, plural <noun phrase plural>.	Tributary is a concept, plural tributaries.	Class:Tributary	No extra OWL needed for the plural. Only

			indicate if not straightforward "s"
A <concept > is uniquely defined as: <sentence> ; <sentence> ; (iterate) <sentence>.	A Source is uniquely defined as: is a kind of Spring or Wetland; feeds a River or a Stream.	Source = Spring or Wetland and feeds Some (River or Stream)	To create a defined class.
Use references: <string1 > from <string 2 >; (iterate) <string3 > from <string4 >.	Use references: OSHydro from http://www.ordnancesurvey.co.uk/Rabbit/Hydrology.rbt ; OSBuildings from http://www.ordnancesurvey.co.uk/Rabbit/Buildings.rbt .	No translation	No translation is needed as it all compiles to one OWL file without imports
Refer to <concept1 > [ref] as <concept2 >.	Refer to Pub [OSHydro] as Public House.	OSHydro:Pub = PublicHouse	
<sentence > [ref]	A River flows into a Sea [OSHydro].	River -> flowsInto Some Sea	As usual
B.1.2 Concepts			
A <concept> <relationship> <concept>. A <concept> <relationship> a <concept>.	A Drain has primary purpose Drainage. A River flows into a Sea.	Drain -> hasPrimaryPurpose Some Drainage River -> flowsInto Some Sea	Some Values From These constructs are all encoded the same in OWL -
A <concept> <relationship at least 2 <concept plural>.	A Braided River Stretch flows in at least 2 Channels.	BraidedRiverStretch -> flowsIn min 2 Channel	

A <concept> <relationship> at most 2 <concept plural>	A Child has at most 2 Parents.	Child -> hasParent max 2 Parent	
A <concept> <relationship> exactly 2 <concept plural>.	A Channel has exactly 2 Banks.	Channel -> hasBank exactly 2 Bank	
A <concept> only <relationship> a <concept>.	A Mill Stream only flows in a Mill Race	MillStream -> flowsIn some MillRace and flowsIn only MillRace	Closure axiom
A <concept> only <relationship> a <concept> or a <concept> or nothing.	A Basin is only connected to a Channel or a Pipe or a Basin or nothing.	Basin ->connectedTo only (Channel or Pipe or Basin)	All values from (NB:connectedTo is a functional property - not demonstrating xor here.)
A <concept> <relationship1> <concept>. <relationship1> is opposite to <relationship2>.	Prevents is opposite to enables. A Waterfall prevents Transport.	Waterfall -> not enables some Transport	
A <concept> does not <relationship plural> <concept>.	A Backwater does not have a Current.	Backwater -> not hasCurrent some Current	
A <concept> can also be a <concept>.	A River can also be a Stream. An Employee can also be a Parent.	All sibling primitive concepts are set to be disjoint, except for River and Stream or Employee and Parent (respectively)	
A <concept> usually <relationship> <concept>.	An Aqueduct usually contains Water.		This cannot be translated to OWL - for now, pattern as for sentence without usually.

A <concept> <relationship> a <concept> or a <concept> or a <concept>.	An Inlet is part of a Lake or a Sea or a River.	Inlet -> partOf some (Lake or Sea or River) and NOT (partOf some Lake and partOf some Sea) and NOT (partOf some Lake and partOf some River) and NOT (partOf some Sea and partOf some River) and partOf only (Lake or Sea or River)	Exclusive OR is the default. There are often other ways to model this e.g. if the relationship is made functional
A <concept><relationship> <concept> that <relationship> <concept>	A Pub has at least 1 Building that has use Boozing.	pub -> hasPart min 1 (building and hasUse boozing)	
A <concept> <relationship> 1 or more of a <concept> or a <concept>.	A Floodplain is adjacent to 1 or more of a River or a Stream.	Floodplain -> adjacentTo Some (River or Stream)	Non exclusive or structure
B.1.3 Relationships			
A <concept1> has a <concept2>	A Whirlpool has a Current.	Whirlpool -> hasCurrent Some Current	
A <concept 1> <relationship> <concept 2>	A Country has part Government Office Region.	Country -> hasPart Some GovernmentOfficeRegion	No indefinite article allowed to split the relationship
A <relationship> is a transitive relationship.	contains is a transitive relationship.	Contains hasSubproperty directlyContains directlyContains is not transitive	And then directlyContains is used in the rest of the OWL
B.1.4 Axioms			

A <concept> <relationship> <instance>.	A Loch is a kind of Lake. A Loch is located in Scotland. Scotland is an instance of a Region Of Country.	Loch -> Lake and locatedIn has {scotland} where scotland is an instance of RegionOfCountry.	
A <concept> <relationship> <concept - direct object> <preposition> a <concept - indirect object>.	A Confluence connects a River or a Stream or a Canal to a River or a Stream or a Canal.	Confluence-> providesConnection some (Connection and (connectionFrom some (Canal.Water or River or Stream)) and (connectionTo some (Canal.Water or River or Stream)))	Concept-level n-ary relation. Difficulty in automatic translation
A <concept> can only be <concept> or <concept> ...or <concept>.	A Season can only be Winter or Spring or Summer or Autumn.	Season = (Winter or Spring or Summer or Autumn)	This indicates that there can be no other subclasses of Season.
everything that <relationship> that <relationship> some <concept> will also <relationship> some <concept>	Everything that has a part that contains some Water will also contain some Water.	hasPart Some (contains Some Water) -> contains Some Water	General Concept Inclusion Axiom
everything that <relationship> that <relationship> something will also <relationship> that thing	Everything that has a part that contains something will also contain that thing.	hasPart o contains -> contains	Complex Role Inclusion Axiom - only possible in OWL 1.1
B.1.5 Instances			
<concept 1> is an instance of <concept 2>. <concept 1> <relationship>	osgb0001 is an instance of a Real World Object. osgb0001 has a hospital use.	Osgb0001 rdf:type RealWorldObject Osgb0001 hasUse hospital_use hospital_use rdf:type HospitalUse	Using a class as a value

<concept3>. <concept 3> is a concept.	hospital use is a concept.		
--	----------------------------	--	--

B.2 Receptive sentences

B.2.1 Concepts			
A <concept> is anything that is <sentence1> and that <sentence 2> and that ...	A Source is anything that is a Spring or a Wetland and that feeds a River or a Stream.	Source = Spring or Wetland and feeds Some (River or Stream)	Defined class

B.2.2 Relationships			
If <concept 1> <relationship> <concept 2> and <concept 2> <relationship> <concept 3> then <concept 1> <relationship> <concept 3>.	If a Bank is part of a Channel and a Channel is part of a River, then a Bank is part of a River.	<owl:ObjectProperty rdf:about="#partOf"> <rdf:type rdf:resource="#owl;TransitiveProperty"/> </owl:ObjectProperty>	Transitive relationship
If a <concept 1> <relationship> a <concept 2> then the <concept 2> <relationship> the <concept 1>.	If a Floodplain is adjacent to a River, then the River is adjacent to the Floodplain.	<owl:ObjectProperty rdf:about="#adjacentTo"> <rdf:type rdf:resource="#owl;SymmetricProperty"/> </owl:ObjectProperty>	Symmetric relationship
A <concept> <relationship>	A River is part of itself.	<owl:ObjectProperty rdf:about="#partOf"> <rdf:type	Reflexive relationship

itself.		<code>rdf:resource="#owl;ReflexiveProperty"/> </owl:ObjectProperty></code>	
A <concept> does not <relationship> itself.	A River does not flow into itself	<code><owl:ObjectProperty rdf:about="#flowInto"> <rdf:type rdf:resource="#owl;ReflexiveProperty"/> </owl:ObjectProperty></code>	Irreflexive relationship (OWL 1.1)
If 1 <concept> <relationship> another <concept> then the second <concept> cannot <relationship> the first.	If 1 River flows into another River then the second River cannot flow into the first.	<code><owl:ObjectProperty rdf:about="#flowInto"> <rdf:type rdf:resource="#owl;AntisymmetricProperty"/> </owl:ObjectProperty></code>	Anti-symmetric relationship (OWL 1.1)
If a <concept1> <relationship1> a <concept 2> then the <concept 2> <relationship 2> a <concept 1>.	If Water is contained in a river then the River contains Water.	<code><owl:ObjectProperty rdf:about="#contains"> <owl:inverseOf <owl:ObjectProperty rdf:about="#containedIn"/> </owl:inverseOf> </owl:ObjectProperty></code>	Inverse relationship
A <concept> can only <relationship> 1 <concept>.	An Address can only have 1 Postcode.	<code><owl:ObjectProperty rdf:about="#hasPostCode"> <rdf:type rdf:resource="#owl;FunctionalProperty"/> </owl:ObjectProperty></code>	Functional relationship
1 <concept> <relationship> a <concept>.	1 Postcode is assigned to an Address.	<code><owl:ObjectProperty rdf:about="#isAssignedTo"> <rdf:type rdf:resource="#owl;InverseFunctionalProperty"/> </owl:ObjectProperty></code>	Inverse functional relationship
The <relationship> relationship can only have a <concept> as an object.	The has address relationship can only have an Address as an object.	<code><owl:ObjectProperty rdf:about="#hasAddress"> <rdfs:range <owl:Class</code>	Specifying range restriction on the hasAddress

		<pre> rdf:about="#Address"/> </rdfs:range> </owl:ObjectProperty> </pre>	relationship
The <relationship> relationship can only have a <concept> as a subject.	The capital city of relationship can only have a City as a subject.	<pre> <owl:ObjectProperty rdf:about="#capitalCityOf"> <rdfs:domain> <owl:Class rdf:about="#City"/> </rdfs:domain> </owl:ObjectProperty> </pre>	Specifying domain restriction on the isCapitalCityOf relationship
A <concept> cannot be both <relationship 1> and <relationship 2>	A River cannot be both larger than and smaller than a Stream		Disjoint properties

C Backus-Naur Form Grammar

C.1: conventions:

ISO Extended BNF (main set) is used in this appendix.

C, R, I (or with subscripts) denotes the name of a concept, relationship or instance.

V denotes a data type value.

CC denotes a collection.

C.2 Numerical values:

```
Digit = = '0'|'1'|'2'|'3'|'4'|'5'|'6'|'7'|'8'|'9';
positive-integer = (digit -'0'),{digit};
non-negative-integer = '0' | positive-integer;
integer = ('-',positive-integer)|non-negative-integer;
decimal = integer, '.'digit,{digit};
numerical-value = integer | decimal;
```

C.3 String and string-literal:

```
letter = 'a'|...|'z'|'A'|...|'Z';
character = letter|digit|'_'|'-' | " ' ";
string = letter,{character};
String-literal = ' " ',string,' " ';
```

C.4 Productive sentences:

```
rabbit-sentence
= ['[',external-ontology-alias,']'] subject predicate-verb
object,eos-symbol;

External-ontology-alias = string;

eos-symbol = '.';          (* end-of-sentence-symbol *)

subject = (indefinite-pronoun relative-clause) | ([determiner] sub-
noun-phrase);

indefinite-pronoun = 'someone' | 'anyone' | 'everyone' |
'something'| 'everything'| 'anything';
```

```

relative-clause = 'that' predicate-verb object [end-position-adjunct];

determiner = 'a' | 'an' | 'the' | 'some';

sub-noun-phrase = {adverb} {adjective} {noun} noun [NI-sub-postmodifier];

    NI-sub-postmodifier = prepositional-phrase;

predicate-verb
= lexical-verb-phrase | copular-verb-phrase1 | copular-verb-phrase2;

Lexical-verb-phrase = [negation|mid-position-adjunct]
{verb|phrasal-verb};

    negation = ('do' | 'does') 'not';

    mid-position-adjunct = 'usually' | 'only' | 'may' | 'may only' | ...;

    verb = transitive-verb;

    phrasal-verb = transitive-phrasal-verb;

copular-verb-phrase1
= link-verb [mid-position-adjunct1 | negation] copular-complement-phrase | passive-verb-phrase;

    link-verb = 'is' | 'are' (* 'are' for more flexibility *)
    mid-position-adjunct1 = 'usually' | 'only' ;
    negation = 'not';

    copular-complement-phrase = 'associated with' | . . . ;

copular-verb-phrase2
= (mid-position-adjunct2 'be') | link-verb copular-complement-phrase | passive-verb-phrase;

    Mid-position-adjunct2 = 'may' | 'may only';

object =
[collection-descriptor] object-noun-phrase {' , ' | [ ' , ' ], conjunction object-noun-phrase } [end-collection-descriptor];

Collection-descriptor = 'one or more of';

object-noun-phrase = [obj-pre-head] obj-head [obj-post-head];

    obj-pre-head = [determiner | (cardinality-modifier cardinality-value)];

        determiner = 'a' | 'an' | 'the';

```

```

cardinality-modifier = 'at least'|'at
most'|'exactly';

cardinality-value = non-negative-integer;

obj-head = {adverb} {adjective} {noun} noun [NI-obj-
postmodifier];

NI-obj-postmodifier = prepositional-phrase;

obj-post-head = prepositional-phrase | (prepositional-
phrase relative-clause);

relative-clause = relative-pronoun predicate-verb
object;

relative-pronoun = 'that'|'which'|'who';

predicate-verb

(* object: recursive definition *)

conjunction = 'and' | 'or';

end-collection-descriptor = 'or both' | 'or nothing else'|
'or nothing';

```

C.5 Declarative sentences:

C.5.1 concepts and instances

```

Concept-declaration-sentence
= sub-noun-phrase 'is a concept' [', plural' concept-name-
plural] ['['external-ontology-alias']'], '.';

Concept-name-plural = string;

External-ontology-alias = string;

Defined-concept-sentence
= C 'is uniquely defined as: defined-clause {, " ; " defined-
clause}, '.';

defined-clause = rabbit-sentence - eos-symbol; (* without '.'
*)

Value-partition-sentence = C 'can only be' CC;

Concept-not-disjoint-sentence = C1 'can also be' C2;

Sub-concept-sentence = C1 'is a kind of' | 'is a type of' C2, '.';

Instance-declaration-sentence = I 'is an instance of' C, '.';

```

C.5.2 Relationships

```

relationship-declaration-sentence
  = relation-verb-phrase 'is a relationship' ['that takes a
    value as an object'] ['plural' relationship-name-
    plural], '.';

Relation-domain-sentence = R 'can only have' C|CC 'as a subject.';

Relation-range-sentence = R 'can only have' C|CC 'as an object.';

Transitive-relation-sentence = R 'is transitive.';

Functional-relation-sentence = R 'references a single instance.';

Symmetric-relation-sentence = R 'is symmetric.';

Inverse-functional-relation-sentence
  = R 'is referenced by a single instance.';

Reflexive-relation-sentence = R 'is reflexive.';

Inverse-relation-sentence
  = R1 'is inverse of' | 'is inverse relationship of' R2;

```

C.5.3 External resources

```

External-ontology-declaration-sentence
  = 'Use references:' external-ontology-reference-clause{','; '
  external-ontology-reference-clause}, '.';

External-ontology-reference-clause
  = external-onto-alias 'from' external-onto-URL;

External-onto-alias = string;

External-onto-URL = string-literal;

External-concept-declaration-sentence
  = external-concept-name 'is a concept [' external-ontology-
  alias '].';

External-concept-name = string;

External-ontology-alias = string;

External-relation-declaration-sentence
  = External-Relationship-name 'is a relationship' '['
  external-onto-alias '].';

External-relation-name = string;

External-instance-declaration-sentence
  = External-instance-name 'is an instance' '[' external-onto-
  alias '].';

External-instance-name = string;

```

```
External-resource-alias-declaration-sentence
  = 'Refer to' external-resource-name 'as' local-alias;

External-resource-name = string;

Local-alias = string;

External-rabbit-sentence-inclusion
  = '['external-onto-alias']' external-rabbit-sentence;

External-rabbit-sentence = rabbit-sentence;
```