OWL Simplified English: a finite-state language for ontology editing

Richard Power

Department of Computing
Open University
Milton Keynes, UK
r.power@open.ac.uk

Abstract. We describe a controlled fragment of English for editing ontologies in OWL. Although this language substantially overlaps other CNLs that have been proposed for this purpose, it has a number of special features designed to simplify its learning and use. First, the language allows users to start typing in sentences with little or no preliminary effort in building a controlled vocabulary or lexicon. Second, it disallows sentences that *people* interpret as structurally ambiguous. Third, it employs a finite-state grammar, so facilitating fast and reliable implementation of an editing tool. These advantages are gained at the cost of severe restrictions in coverage, which mean that the majority of potential OWL axioms cannot be expressed. However, analysis of axiom patterns from several ontology repositories suggests that these constraints are almost invariably respected by ontology developers, so that in practice the loss of expressivity is rarely noticeable.

Keywords: semantic web, ontology editing, controlled natural language

1 Introduction

A variety of controlled languages for the semantic web have been proposed, prominent examples being ACE (Attempto Controlled English) [7], SOS (Sydney OWL Syntax) [20], and Rabbit [6]. Most of these languages have been used both in systems that generate text from OWL code (*verbalisers*) and in systems that produce OWL code by interpreting text (*editors*). Examples of editing tools are AceWiki [8], ROO [4], RoundTrip Ontology Authoring [3], and FluentEditor [2]. These applications belong to a tradition in CNL research that predates the semantic web, and includes for example Computer Processable English [15] and PENG [18]; the general aim of this research is to design subsets of English that can be unambiguously interpreted in some system of formal logic, while allowing sufficient freedom for human authors to write fluent texts.

In comparing different CNLs for knowledge formation, it is useful to distinguish a number of design requirements and potential trade-offs among them. Schwitter [17] and Kuhn [9] have both distilled these requirements into four groups, which Kuhn calls clearness, naturalness, simplicity and expressivity.

Clearness means that the syntax and semantics of the CNL should be well-defined, and that any well-formed sentence in the CNL should be mapped unambiguously to a formal semantic interpretation. *Naturalness* requires that sentences in the CNL are understandable by people, and includes issues like whether sentences are perceived as belonging to English or some other natural language, and whether people interpret them unambiguously as having the desired meaning. *Simplicity* requires that the CNL is easily described and processed: for people this depends on whether the language is easily learned and applied, for instance when checking that a sentence is correct; for programs it depends on whether a sentence typed in by the user can be efficiently parsed (so confirming that it belongs to the CNL) and interpreted. Finally, *expressivity* concerns how fully the CNL covers the desired problem domain—or in the case of a semantic web editor, its coverage of OWL. As will be obvious, these requirements may conflict, so that different trade-offs are possible between (for example) simplicity and expressivity.

We describe in this paper a CNL for editing ontologies which aims broadly to maximise *simplicity* and *naturalness* at the expense of expressivity; however, we also present empirical evidence that this loss of expressivity is more theoretical than practical, since the proposed language can cover almost all OWL patterns that are actually found in ontology corpora. Unlike ACE and some other languages mentioned above, this CNL is designed solely for convenience in verbalising and editing OWL ontologies, and has no other linguistic or logical pretensions whatever; for this reason we provisionally name it *OWL Simplified English*. In more detail, the design principles informing the CNL are as follows.

1. It should be possible to describe the language very briefly; as a rough guide, the basic rules should fit comfortably on a sheet of A4.
2. Any preliminary work on the lexicon should be minimised: ideally, a user should be able to type in sentences straight away, without having to list content words or entity names.
3. The grammar should disallow sentences that people perceive as structurally ambiguous.
4. The grammar should be finite-state, so that sentences can be parsed and interpreted efficiently by a finite-state transducer.
5. No effort should be made to guarantee that sentences are grammatical according to the conventions of normal English. Provided that the CNL makes it *possible* to write fluent English, adherence to conventional grammar can be left to the human author.

Empirically, OWL Simplified English is based on two findings derived from analysis of ontology corpora [13, 14]. The first is that the names of individuals, classes and properties in an ontology have distinctive features which can be exploited in order to determine where they begin and end. The second is that in practice, complex OWL expressions are invariably right-branching, and so lend themselves to verbalisations that are structurally unambiguous and can be described by a finite-state grammar. These two ideas will be developed in detail in the following sections.

2 Recognising entity names

The following sentence verbalises a fairly complex OWL statement using strategies typical of the CNLs that have been proposed:

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

To interpret this sentence (which is correctly formed in OWL Simplified English), the parser must recognise a number of *entity names*—that is, phrases denoting individuals and atomic classes or properties:[1] thus 'London' and 'the United Kingdom' name individuals, 'city' and 'boroughs' name classes, and 'is capital of' and 'is divided into' name object properties. The other words in the sentence provided a *scaffolding* signalling various axiom and class constructors; they comprise the copular 'is', the indefinite article 'a', the conjunction 'and', the relative pronoun 'that', and the quantifying phrase 'at least 30'. Abstracting from the entity names, we could represent the sentence pattern like this:

> [Individual] is a [class] that [has-property] [Individual] and [has-property] at least 30 [class].

In general, a CNL for verbalising OWL ontologies requires only a handful of function words to provide this scaffolding. As well as the words already illustrated, most CNLs use 'or' for union, 'not' for complement, 'exactly' for exact cardinality, 'at most' for maximum cardinality, and the determiners 'no' and 'every' in sentences expressing *DisjointClasses* and *SubClassOf*. By contrast, the potential vocabulary for entity names is vast, ranging from the commonplace to the highly technical, and will include many words that can belong to more than one part of speech (e.g., 'rank', which can be noun, adjective or verb). To achieve the goals of OWL Simplified English, we must find formation rules for entity names that allow the parser to determine where a name begins and ends, and whether it denotes an individual, a class or a property; moreover, the rules should allow users sufficient freedom to construct appropriate names, while not demanding a great deal of preliminary effort in specifying a controlled vocabulary. It is not at all obvious that this combination of desiderata can be met.

For evidence on how entity names are formed in practice, we can refer to studies on the structure of identifiers and labels in ontology corpora [10, 13], which show that individuals, classes and properties are named by distinctive part-of-speech sequences. *Individual* identifiers are made up mostly of proper nouns, common nouns and numbers; where the opening word is not a proper noun, the definite article 'the' is often implicit. *Class* identifiers are composed mostly of common nouns and adjectives, although they may also contain numbers and proper nouns (e.g., '1912 Rolls Royce'). *Property* identifiers often open with a verb or auxiliary ('is', 'has') in the present tense; their other constituents are mostly common nouns, participles, and prepositions. In all three types of

[1] The qualification 'atomic' here distinguishes elementary classes/properties from ones that are constructed, for instance through a restriction or intersection functor.

identifier, the function words listed above as scaffolding are very rare—indeed, the only function words that are at all common are auxiliaries (especially 'has') and prepositions, both found mostly in property identifiers.

On the basis of these findings, suppose that we constrain entity names as follows:

1. Some listed function words including 'a', 'every', 'and', 'that', must not be used at all. These words can therefore serve as signals that an entity name has just ended, or is about to begin.
2. Individual names must begin either with a proper noun or the definite article 'the', and may not contain verbs or auxiliaries.
3. Property names must begin either with 'is' or 'has' (or their plurals), or with a verb in the present tense, and may not include proper nouns, numbers or strings (these would signal the onset of an individual name or a literal).
4. Property names opening with an auxiliary must contain at least one further word.
5. Class names may not contain verbs or auxiliaries (which would signal the onset of a property name).

Consider the application of these rules to the following sentences, in which listed function words are shown in italics:

> *Every* capital city *is an* urban area.
> Hyde Park *is* located in London.

Assume that we have no knowledge of the content words 'capital', 'city', etc., so that the first sentence might as well be 'Every xxxxxxx xxxx is an xxxxx xxxx'. We can tell immediately that 'capital' opens a class name, because it is preceded by 'every'. On reaching 'is' we know that the class name is over, but cannot yet tell whether 'is' serves as scaffolding or as the opening of a property name. The next word 'an' rules out a property name (which would require at least one further word after 'is'), and foreshadows another class name, which opens with 'urban' and continues up to the full stop.

The second sentence, which is effectively 'Xxxx Xxxx is xxxxxxx in Xxxxxx', provides little scaffolding, but we can still interpret it if we are allowed to assume that any unlisted word opening with a capital letter is a proper noun.[2] In this case 'Hyde' must open an individual name; this continues up to 'is', which as before has two possible interpretations, but this time the next word is unlisted, and so we may infer that 'is' opens a property name. After adding 'located' and 'in' to this property name we arrive at another proper noun ('London'), which in this context can only signal the opening of another individual name.[3]

[2] Note that this criterion would not work for languages such as German in which both common and proper nouns start with a capital letter. This raises the general issue of whether superficial methods, like those proposed here for English, could be found for other languages, so permitting for example an OWL Simplified German—a point that we have not yet investigated.

[3] Note that 'located', although a verb, cannot open a property name because it is not in the present tense.

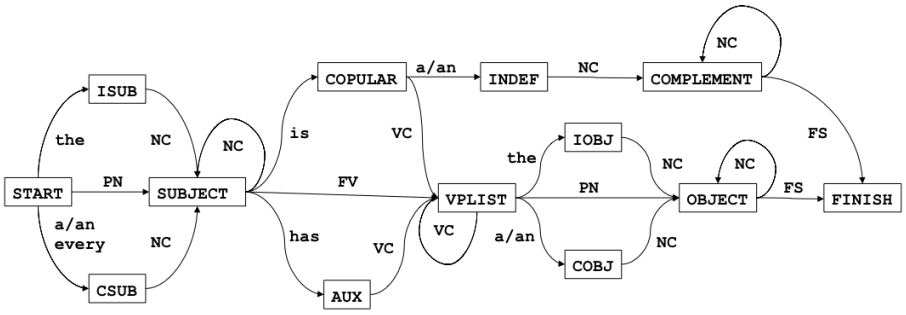

Fig. 1. Finite-state network for recognising basic clause patterns. PN = Proper Noun; NC = Noun-phrase Continuation; VC = Verb-phrase Continuation; FV = Finite Verb; FS = Full-Stop.

These examples are encouraging because they indicate how sentences might be interpreted *with minimal knowledge of content words.* Any words used as verbs in the present tense must be listed,[4] but otherwise the parser can rely on lists of common function words, supplemented by inferences based on typography through which content words can be classified as follows:

Category	**Description**
Verb	Non-auxiliary verb in the present tense, listed by the user
Number	sequence of digits, possibly including decimal point (10, 2.5)
String	sequence of characters inside double-quotes ("J. Doe")
Proper Noun	any other word beginning with a capital letter (London)
Noun/Other	any other word beginning with a lower-case letter

With this classification it is straightforward to define a finite-state transducer for basic clause patterns which can distinguish individual, class and property names without a lexicon of content words, and so construct an OWL expression from the input sentence. A simplified version of the grammar is shown in figure 1, where 'verb-phrase continuation' signifies only Noun/Other, while 'noun-phrase continuation' also includes Number, String, Proper Noun, and the definite article. The full grammar covers a wider range of sentences, and includes actions, defined on each arc, which build the OWL output. These are illustrated by the following table, which shows the state transitions in interpreting the sentence 'London is a capital city'.

[4] The reason verbs must be listed is that otherwise they could be construed as nouns continuing the preceding class name. Thus in the sentence 'Every building works manager reports to a regional director', both 'works' and 'reports' are potentially verbs opening a property name, and the program cannot correctly recognise 'building works manager' as a class name, and 'reports to' as a property name, unless 'reports' is a listed verb and 'works' is not.

State 1	Word	State 2	OWL expression
START	London	SUBJECT	`CA(?,I(London))`
SUBJECT	is	COPULAR	`CA(?,I(London))`
COPULAR	a	INDEF	`CA(C(?),I(London))`
INDEF	capital	COMPLEMENT	`CA(C(capital),I(London))`
COMPLEMENT	city	COMPLEMENT	`CA(C(capital_city),I(London))`
COMPLEMENT	.	FINISH	`CA(C(capital_city),I(London))`

Starting from the state START, the transducer reads the words of the sentence (including the final full-stop) one at a time, from left to right, traversing an arc every time a word is consumed. At each state, the current word will match the conditions on at most one arc, so that progress through the network requires no backtracking. If there are no outgoing arcs that match the current word, interpretation fails; if the current word is a full-stop and there is an arc leading to FINISH, interpretation succeeds.

OWL expressions are built progressively using actions defined on each arc. For instance, on consuming 'London' (classified as a proper name) at the initial state START, it can be inferred that the axiom will have the functor *ClassAssertion*, and that its second argument will be an individual with a name beginning 'London'.[5] This is not necessarily the complete individual name (perhaps the subject is 'London Bridge'), but on consuming 'is' we know that the name is complete, and await developments: perhaps 'is' opens a property name, and perhaps it merely denotes class membership. The next word 'a', which cannot occur in a property name, confirms the latter possibility, and prepares us to receive a class name. Two noun-phrase continuations follow, both of type Noun/Other; finally, the full-stop indicates that both the class name and the whole axiom are complete.

As will be obvious, the grammar in figure 1 will accept and interpret many sentences that are blatantly ungrammatical in English. To give just one example, the rule for constructing class names does not guarantee that anything remotely resembling a noun phrase will result: a nonsensical sequence of noun-phrase continuations such as 'of of the of' will be accepted, allowing such sentences as 'An of of the of is a the the'. Of course we could tighten the grammar to ensure, for example, that class names could not begin with a preposition or the definite article, but we see no advantage in doing so except for helping the user to correct accidental slips. This has to be set against several advantages of our permissive policy: (1) the transition network is simpler (and faster); (2) the rules of the language can be described more briefly; and (3) we allow slang or technical phrases that violate normal English (e.g., 'an in your face person', where the noun group opens with a preposition).

[5] We abbreviate `ClassAssertion` to `CA` and `Individual(#London))` to `I(London)` (etc.) so that the table fits on the page.

OWL Functor	CNL Pattern	Example
`SubClassOf(C1 C2)`	A [C1] is a [C2]	A city is a place
`ClassAssertion(C I)`	[I] is a [C]	London is a city
`ObjectPropertyAssertion(OP I1 I2)`	[I1] [OP] [I2]	London contains Hyde Park
`EquivalentClasses(C1 C2)`	A [C1] is any [C2]	A human is any person
`DisjointClasses(C1 C2)`	No [C1] is a [C2]	No village is a city
`ObjectSomeValuesFrom(OP C)`	[OP] a [C]	contains a park
`ObjectHasValue(OP I)`	[OP] [I]	contains Hyde Park
`DataPropertyAssertion(DP I L)`	[I] [DP] [L]	Hyde Park dates from 1536
`DataHasValue(DP L)`	[DP] [L]	dates from 1536
`ObjectAllValuesFrom(OP C)`	[OP] only [C]	contains only parks
`ObjectExactCardinality(N OP C)`	[OP] exactly [N] [C]	contains exactly 10 parks
`ObjectMinCardinality(N OP C)`	[OP] at least [N] [C]	contains at least 10 parks
`ObjectMaxCardinality(N OP C)`	[OP] at most [N] [C]	contains at most 10 parks

Table 1. OWL functors covered by the CNL. C = Class; I = Individual; OP = Object property; DP = Data property; L = Literal; N = Cardinality.

3 Basic clause patterns

We list in table 1 the linguistic patterns for expressing common axiom and class constructors in OWL Simplified English. For the most part these conform to other CNLs [19], the main exception being the pattern for *EquivalentClasses*, where we suggest using 'any' in the predicate (e.g., 'A pet-owner is any person that owns a pet') as a means of articulating both directions of the equivalence.[6] Another exception is that we allow the indefinite article as well as the quantifier 'every' at the start of the pattern for *SubClassOf*; this derives from an an evaluation of the SWAT Tools verbaliser [21], in which ontology developers declared a preference for the indefinite article at least in some contexts.

To cover *DataPropertyAssertion* and *DataHasValue* the CNL needs names for data properties and literals. At present we assume that literals are named by single tokens belonging to the word categories Number and String (see last section), and that object and data properties are distinguished only by their context (i.e., whether they are followed by a literal). The only other complication comes from universal and numerical quantifiers in restrictions (e.g., 'only parks', 'exactly 10 parks'), which require a singular/plural distinction on class and property names—a potential problem for our approach in which the details of English syntax are disregarded. Our provisional solution is to pluralise by applying standard morphological rules (including common exceptions) to the *head* word of a class or property name, defined as the final word of a class name,[7] or the opening verb/auxiliary in a property name. It is left to the user to avoid names for which this simple rule yields odd results.

[6] We are planning an empirical study on which of the proposed formulations best expresses equivalence.

[7] Except for a name like 'student of art', in which case it will be the word preceding the preposition.

4 Coordination, relative clauses and negation

We now turn to the class constructors for intersection and union, which are typically expressed through coordinated noun phrases or verb phrases, or through relative clauses introduced by 'that'. As we will see, the resulting sentences may become complex and prone to structural ambiguity; to avoid this, OWL Simplified English strongly constrains the permitted sentence patterns.

The first constraint, a very sweeping one, is that constructed classes of any kind may be expressed *only in the predicate*:[8] we disallow sentences such as 'Every person that owns a pet is a pet-owner' where the subject is complex. In terms of the underlying logic, this constraint means that the left-hand side of an axiom constructed with '$\sqsubseteq$', '$\in$' or '$\equiv$' must be atomic: an axiom like $C_1 \sqcap \exists P.C_2 \sqsubseteq C_3$ is excluded.[9] We impose this constraint because it enormously simplifies the language at small practical cost, since axioms with a complex subject term occur very rarely in practice; in a corpus study of around half a million axioms, we found that 99.8% of them had atomic subject terms and only 0.2% had complex subject terms [14]. Perhaps this is not surprising, given that the purpose of description logic is to *describe* things—indeed, the graphical editor Protégé *assumes* that subject terms will be atomic and makes no provision for complex subjects at all [16].

4.1 Intersection

In agreement with all other CNLs that we know of, OWL Simplified English expresses *ObjectIntersectionOf* by the pattern '[X] and [Y]' when '[X]' and '[Y]' are both noun-phrases or both verb-phrases, and by the pattern '[X] that [Y]' when '[X]' is a noun-phrase and '[Y]' is a verb-phrase. This policy achieves the requirement (mentioned earlier) of simplicity, but runs the risk of violating *naturalness* since even fairly simple combinations of intersection and restriction may yield sentences that are structurally ambiguous:

OWL pattern	**English verbalisation**
$C \sqsubseteq \exists P.(C \sqcap C)$	Every dog lives in a house and a kennel
$C \sqsubseteq \exists P.(C \sqcap (\exists P.C \sqcap \exists P.C))$	Every dog lives in a kennel that is located in a garden and is painted a shade of pink

For the first of these sentences, the OWL pattern should mean 'Every dog lives in something that is both a house and a kennel'. For the second, ACE disambiguation rules would imply that it is the dogs, not the kennels, that are painted pink.

[8] In most cases, the predicate of a sentence in OWL Simplified English corresponds to the second argument of the underlying OWL axiom. This holds for *SubClassOf* and *EquivalentClasses*, for example. Two exceptions are *ClassAssertion* and *ObjectPropertyAssertion*, for which the argument verbalised as the subject (in all proposed CNLs) is the second, and the other argument(s) are verbalised as the predicate.

[9] Note that a developer could overcome this limitation by introducing a new class $C_4 \equiv C_1 \sqcap \exists P.C_2$ and then asserting $C_4 \sqsubseteq C_3$.

To eliminate these and other cases of structural ambiguity, OWL Simplified English allows only three strategies for constructing complex sentences: noun-phrase lists, verb-phrase lists, and verb-phrase chains. These are defined as follows:

1. A *noun-phrase list* has the form '[C] and [C] and [C] ...' where each '[C]' is a class name.
2. A *verb-phrase list* has the form '[P] a [C] and [P] a [C] and ...', or the equivalent using other quantifiers such as 'only' and 'exactly two' (or using individuals or literals instead of classes).
3. A *verb-phrase chain* has the form '[P] a [C] that [P] a [C] that ...', or the equivalent using other quantifiers or other property values (individuals or literals).[10]

Examples of the three strategies are as follows:

> (1) London is a city and a capital and a tourist attraction. (*Noun-phrase list*)
> (2) London is capital of the UK and has as population 15000000. (*Verb-phrase list*)
> (3) London is capital of a country that is governed by a man that lives in Downing Street. (*Verb-phrase chain*)

We believe that each of these constructions is free from structural ambiguity, and moreover that they can be *combined* in the same sentence provided that they come in the specified order: noun-phrase list precedes verb-phrase list, verb-phrase list precedes verb-phrase chain.

> London is a city that has as population 15000000 and is capital of a country that is governed by a man that lives in Downing Street.

Here and in the previous examples a parsing algorithm might find alternative analyses of a phrase like 'a city and a capital and a tourist attraction' depending on whether the bracketing was '[a city and a capital] and a tourist attraction' or 'a city and [a capital and a tourist attraction]', but these potential ambiguities are innocuous rather than 'nocuous' [24] owing to the associativity of intersection.[11]

[10] Note that the defining property of a verb-phrase chain is not the occurrence of the word 'that', but the embedding of (at least) one restriction within another retriction. Thus '[C] that [P] a [C] and [P] a [C]' is a one-term noun-phrase list followed by a verb-phrase list, despite the occurrence of 'that', and denotes the constructed class $C \sqcap \exists P.C \sqcap \exists P.C$, which does not contain an embedded restriction. Instead, '[C] that [P] a [C] that [P] a [C]' denotes a constructed class $C \sqcap \exists P.(C \sqcap \exists P.C)$ that contains an embedded restriction (verbalised as a relative clause within a relative clause), and thus ends in a chain.

[11] Since our aim is to avoid sentences that might be misinterpreted by human readers, we not interested here in whether a sentence is ambiguous relative to some specific grammar, but in whether people might assign it different meanings. Research in psycholinguistics has revealed two strategies called *minimal attachment* and *late*

4.2 Union

Using 'and' (or 'that') and 'or' in the same sentence almost always leads to ambiguity, as in these examples:[12]

John is a lawyer or an artist and a pet-owner.
John is a lawyer or an artist that owns a pet.

In one reading, John is definitely a pet-owner, as in 'John is (1) a lawyer or an artist, and (2) a pet-owner. In the other reading he might not be a pet-owner ('John is (1) a lawyer, or (2) an artist and a pet-owner'). As just shown, we can disambiguate these examples with careful punctuation, but this will become cumbersome or break down altogether for more complex sentences.

To avoid such ambiguities by simpler means, OWL Simplified English disallows usage of 'and'/'that' and 'or' in the same sentence. At present, just three patterns with 'or' are allowed: noun-phrase lists, verb-phrase lists, and restrictions over a noun-phrase list.[13] Here are examples of each:

$C \sqsubseteq C \sqcup C$	A married person is a husband or wife
$C \sqsubseteq \exists P.C \sqcup \exists P.C$	A student attends a school or attends a college
$C \sqsubseteq \exists P.(C \sqcup C)$	A student attends a school or college

4.3 Complement

For now we envisage just three cases in which negation will be allowed in a predicate: negating a simple class; negating a simple restriction,[14] and negating the second term of a simple intersection:

$C \sqsubseteq \neg C$	A whale is not a fish
$C \sqsubseteq \neg\exists P.C$	A child does not attend a university
$C \sqsubseteq C \sqcap \neg C$	A consonant is a letter that is not a vowel

As will probably be obvious, expressing the complement of an intersection or union almost always leads to structural ambiguity.[15]

---

closure [1] which are relevant to some of the examples discussed here; however, for the most part we have relied on intuition. Ideally, we would like to test at least some doubtful patterns empirically, as has been done in the study of nocuous and innocuous ambiguity cited here.

[12] Of course the underlying OWL axiom has a precise unambiguous argument structure; the ambiguity resides in the sentence verbalising this axiom, which could also be construed as the verbalisation of a different non-equivalent axiom.

[13] Chains are prohibited since 'or' cannot be used with 'that'.

[14] By a 'simple' restriction we mean one whose value is an atomic entity as opposed to a constructed class.

[15] Forming the complement of a class by prefixing 'non-' also leads to ambiguity for a multiword class name (e.g., 'non-blue whale').

5 Empirical results on coverage

We have outlined a CNL in which structural ambiguity (at least of the 'nocuous' kind) is avoided through severe constraints on the formation of complex predicates using 'and', 'or' and 'that'. In this section we summarise the theoretical and practical consequences of these constraints.[16] In a nutshell, we will argue for two claims:

1. The proposed CNL can express *only a tiny fraction* of the statements that could in theory be constructed using restriction, intersection and union.
2. In spite of this, the proposed CNL can express *almost all* the complex axioms that occur in practice.

To assess the first claim, we need to find some method of enumerating all non-equivalent classes that can be constructed using different combinations of restriction, intersection, and union, and then counting how many can be expressed linguistically under ther constraints defined above. Obviously the set of constructed classes is infinite, so the only practical approach is to count all patterns up to a given level of complexity.

To measure the complexity δ of a constructed class, we can count the number of class constructors: thus C (an atomic class) will have δ=0, $\exists P.C$ will have δ=1, and $C \sqcap C \sqcap C$ will have δ=2. The only complication here is that in OWL, *ObjectIntersectionOf* and *ObjectUnionOf* may have more than two arguments, so that $C \sqcap C \sqcap C$ would normally be encoded using a single functor with three arguments. However, since additional arguments imply additional complexity, we will assume that only two arguments are allowed, so that in such a case two functors would be needed. A class expression can then be represented as a binary tree in which the terminal nodes are atomic entities (classes, properties, etc.), and the non-terminal nodes are restriction, intersection or union functors; the complexity is then given by the number of non-terminal nodes. It is then straightforward to write a program that can generate all binary trees of a given complexity.

As an exercise, let us generate all trees of complexity δ=2, using only two functors, intersection ($\sqcap$) and existential restriction ($\exists$) — by far the most common combination in practice. Using P for any property and C for any atomic class, the full set of class expressions is shown in table 2. Of these patterns, we would argue that three can be disregarded. First, pattern 1 cannot be verbalised unless the inner restriction is recast into a form that can be expressed by a noun phrase;[17] to do this we have introduced 'something that', which is equivalent to expressing the more complex pattern $\exists P.(\top \sqcap \exists P.C)$ (which would be covered

[16] These consequences will be explored more fully in a separate paper.

[17] We are assuming here that restrictions such as $\exists P.C$ are realised by verb phrases in which the object expresses the class C. This policy, which is followed in all the OWL CNLs that we are aware of, runs into difficulty when C is replaced by a restriction, yielding $\exists P.(\exists P.C)$, since for syntactic reasons a verb phrase cannot have another verb phrase as its object.

	Logical pattern	Verbalisation
*1	$\exists P.(\exists P.C)$	Vs (something that) Vs an N
2	$\exists P.(C \sqcap C)$	Vs an N and an N
*3	$(\exists P.C) \sqcap C$	is something that Vs an N and an N
4	$C \sqcap (\exists P.C)$	is an N that Vs an N
5	$C \sqcap (C \sqcap C)$	is an N and an N and an N
*6	$(C \sqcap C) \sqcap C$	is an N and an N and an N

Table 2. All class patterns of complexity δ=2 that can be constructed using intersection and existential restriction. The proposed verbalisation patterns assume that atomic properties are expressed by verbs (V) and atomic classes by nouns (N). Patterns marked with an asterisk can be disregarded (see text).

anyway in the list for complexity 3). Next, since intersection is commutative, patterns 3 and 4 are equivalent, and so are patterns 5 and 6; in such cases we need consider only the form in which the arguments are optimally ordered with simple preceding complex (so eliminating patterns 3 and 6). We are therefore left with three non-equivalent patterns for δ=2, of which two (patterns 4 and 5) can be expressed according to the constraints of our CNL, and one (pattern 2) cannot. Thus if these three patterns were equally common in practice, one-third of the relevant OWL axioms could not be verbalised.

To ascertain the frequency of these patterns in practice, we have collected a corpus of around 550 ontologies from several repositories, containing some 500,000 axioms in all.[18] By far the majority of these axioms have simple predicates (complexities less than 2). Counting only axioms of complexity 2 using intersection and existential restriction, the corpus contains 897 predicates distributed as follows: 892 have the form $C \sqcap (\exists P.C)$, 5 have the form $\exists P.(C \sqcap C)$, and none at all have the form $C \sqcap (C \sqcap C)$. Thus instead of some 300 unverbalisable axioms, as we might have feared, we obtain only five.[19]

With increasing complexity, as one would expect, the proportion of unverbalised axioms rises sharply. Thus for δ=3 there are six non-equivalent patterns of which two are not verbalised (33%); for δ=4 the proportion rises to 60% (6/10); for δ=5, 65% (13/20); for δ=6, 81% (30/37); for δ=7, 86% (66/77); for δ=8, 93% (138/149); and so forth. Note that these figures apply only to classes constructed from restriction and intersection; if we add union and/or complement, the proportion of unverbalised axioms at a given complexity level naturally rises still further.

[18] Our corpus is taken from three sources: first, the University of Manchester TONES repository [22]; second, the Ontology Design Patterns corpus [11]; and finally a collection of about 160 ontologies downloaded from Swoogle [5]. It embraces a wide range of sizes, domains, and authoring styles.

[19] Here and elsewhere in this section, 'unverbalisable' means that the axiom in question cannot be verbalised *by our grammar*; obviously any axiom could be verbalised by *some* grammar.

	Predicate pattern	Frequency
1	$C \sqcap P.C$	1100
2	$P.(C \sqcap P.C)$	473
3	$P.(C \sqcup C)$	434
4	$C \sqcap (P.C \sqcap P.C)$	248
5	$P.(C \sqcap (P.(C \sqcap P.C)))$	164
6	$P.(C \sqcup (C \sqcup C))$	105
7	$C \sqcap (P.C \sqcap (P.C \sqcap P.C))$	78
8	$P.C \sqcap P.C$	59
9	$C \sqcap (P.C \sqcap (P.C \sqcap (P.C \sqcap P.C)))$	47
10	$P.(C \sqcup (C \sqcup (C \sqcup C)))$	43
11	$C \sqcap (P.(C \sqcap P.C))$	39
12	$P.(C \sqcup (C \sqcup (C \sqcup (C \sqcup C))))$	35
13	$P.(P.C)$	33
14	$P.\neg C$	28
15	$P.(C \sqcup (C \sqcup (C \sqcup (C \sqcup (C \sqcup C)))))$	26
16	$P.C \sqcup P.C$	25
17	$P.C \sqcap (P.C \sqcap P.C)$	20
18	$C \sqcap (P.C \sqcap (P.C \sqcap (P.C \sqcap (P.C \sqcap (P.C \sqcap P.C)))))$	20
19	$\neg P.C$	19
20	$P.(C \sqcup (C \sqcup (C \sqcup (C \sqcup (C \sqcup (C \sqcup C))))))$	17

Table 3. Frequencies of predicate patterns including intersection, union and complement. Restrictions are abstracted, so that $P.C$ covers existential, universal and numerical restrictions together with object and data values. Frequencies are based on a corpus of 3648 axioms of complexity $\delta \geq 2$.

By restricting the CNL to non-ambiguous sentence patterns, we thus incur a potentially disastrous loss of coverage; our second claim is that in practice this loss is negligible. We cannot defend this claim in detail here, but as suggestive evidence table 3 lists the twenty most frequent predicate patterns in our corpus, including some with high complexity values (e.g., pattern 18 has δ=12): nevertheless, with one exception, *every single predicate* in the list can be verbalised by one of our permitted sentence patterns. Looking through the list, one finds that again and again ontology authors opt for the familiar forms of restriction lists and chains, while avoiding the combination of intersection and union in the same predicate, almost as if they were composing axioms with the intention of avoiding ambiguous verbalisations. The only exception is the pattern $P.(P.C)$ which, as already mentioned, cannot be verbalised for syntactic reasons unless it is reformulated as $P.(\top \sqcap P.C)$, which corresponds to pattern 2 in the table and can therefore be verbalised unambiguously.

6 Designing an editing tool

The sentence formation rules in OWL Simplified English ensure that complex sentences are *right-branching*, since the first constituent of a coordinated unit is

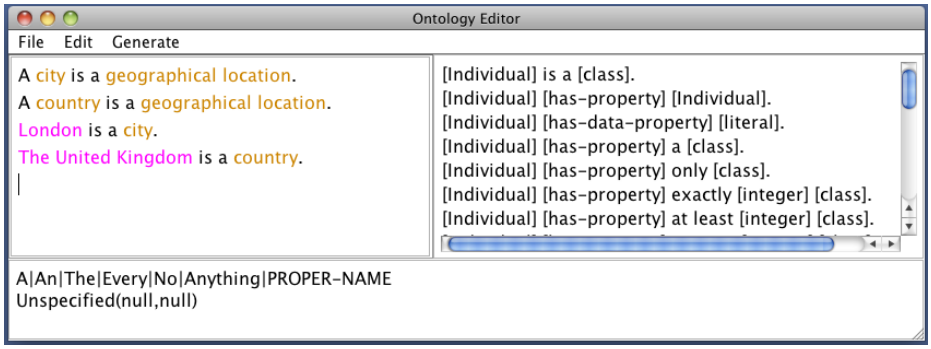

Fig. 2. Snapshot of editing tool

always a simple noun-phrase or verb-phrase expressing an atomic class or simple restriction. As well as minimizing structural ambiguity, this constraint allows the grammar to be cast in the form of a finite-state transducer (as in figure 1) which reads sentences word by word from left to right, while progressively building an interpretation in OWL. Such a grammar brings an obvious advantage in efficiency—processing time is linear with sentence length; it also facilitates implementation of an editing tool that gives feedback and guidance as the user composes each sentence: handcrafted information can simply be associated with each state of the network.

We are now prototyping an editing tool that exploits these advantages in order to combine menu-guided sentence construction, as in WYSIWYM [12], with free text entry. The appearance of the interface is shown in figure 2. Four sentences have been composed in the editing pane at the top left; the cursor is now placed at the start of the fifth line, ready to start a new sentence. As can be seen, entity types are distinguished through a colour code based on the conventions of Protégé: violet for individuals, orange for classes, and blue for properties.[20] In the top right pane are shown all the minimal patterns for complete sentences expressing the various axiom types, with entity names represented through placeholders, as in WYSIWYM applications. The pane at the bottom indicates the words that may be typed in next, along with the interpretation of the current (empty) sentence, shown in OWL Functional Syntax.

To add a new sentence, the user can either select a sentence pattern from the options on the right, or simply start typing, perhaps entering the words 'The Tate'. Recognising a (possibly complete) individual name, the editor will colour this text violet, and the options will be modified to show possible *continuations*—for instance, they will include 'is a [class].'. The feedback in the lower pane will also change, including an OWL expression for which the functor is now known:

```
ClassAssertion(NamedIndividual(#The_Tate_),null)
```

[20] In versions of this paper formatted for printing in black and white, we will show individual names in **bold**, class names in *italics*, and property names underlined.

After completing the name by typing 'Modern', the user might click on the option 'is a [class].', whereupon this string will be copied into the editing pane, so yielding the outline of a complete sentence:

The Tate Modern is a *[class].*

As in a WYSIWYM application, clicking on the place-holder will then yield a list of substitution options in the right pane, based in this case on the class names already used in the text:

The Tate Modern is a [class] .	city country geographical location

Assuming that none of these options fits the bill, the user can simply type in a new class name, replacing the selected substring as in any text editor.

The Tate Modern is a *gallery.*
```
ClassAssertion(NamedIndividual(#The_Tate_Modern),Class(#gallery))
```

A minimal sentence is now complete. To extend the sentence, the user deletes the full stop and types a space: minimal continuations including 'and a [class].', 'or a [class].', and 'that [has-property] [Individual].' then appear in the options pane. If the user selects the last of these, overwrites the place-holder '[has-property]' with the new property name 'is located in', and clicks on '[Individual]', a more complex sentence takes shape:

The Tate Modern is a *gallery* that is located in [Individual] .	London The Tate Modern The United Kingdom

Clicking on 'London' completes the sentence, interpreted as a class assertion with a constructed class as predicate.

The Tate Modern is a *gallery* that is located in **London**.
```
ClassAssertion(NamedIndividual(#The_Tate_Modern),
   ObjectIntersectionOf(Class(#gallery),
      ObjectHasValue(ObjectProperty(#is_located_in),
         NamedIndividual(#London))))
```

If desired, the sentence could be extended further by the same method, yielding perhaps 'The Tate Modern is a gallery that is located in London and contains at least three portraits that are painted by Strindberg.'. This continuation assumes that 'contains' has been listed as a verb; otherwise the finite-state transducer will fail at this point, with the substring after 'and' displayed in red. Verbs are listed through metadata statements marked by the initial character '#':

VERB contain contains

Apart from guiding the editing of single sentences, the prototype has most of the functionality that one would expect from an editing tool: ontologies can be saved, either as text files or in OWL/XML format; ontologies already in OWL/XML format can be imported and converted to text in OWL Simplified English;[21] texts can be regenerated from their OWL interpretations (this will help clear up any errors in morphology, such as 'a animal'); *alternative* versions of the text can also be generated, such as a glossary that groups together the statements about each individual or class, as in the SWAT verbaliser [23].

7 Conclusion

We have described work in progress on developing a CNL for specifying OWL ontologies, and an accompanying editing tool. The hypothesis underlying this work is that ontology editing can be supported using a simple finite-state grammar in which complex sentences are always right-branching. We have argued that such a language brings several potential advantages: effort in specifying a vocabulary or lexicon is minimized; structurally ambiguous sentences are avoided; and sentences can be parsed by a finite-state transducer which efficiently provides feedback on the interpretation assigned so far, the words that may be typed in next, and the linguistic patterns by which a sentence may be completed (or extended when potentially complete). These advantages are gained only by accepting severely reduced coverage of the OWL statements that are possible in principle; however, analysis of several hundred existing ontologies indicates that in practice, developers overwhelmingly favour right-branching expressions that can be verbalised by our grammar. As will be obvious, a crucial question not yet addressed is whether the proposed CNL and editing tool prove effective in user trials, especially for domain experts with limited knowledge of OWL.

References

1. Gerry Altmann. Ambiguity in sentence processing. *Trends in Cognitive Sciences*, 2(4):146–152, 1998.
2. Cognitum. FluentEditor for OWL. `http://www.cognitum.eu/Products/FluentEditor/`, 2012. Last accessed: 5th March 2012.
3. Brian Davis, Ahmad Ali Iqbal, Adam Funk, Valentin Tablan, Kalina Bontcheva, Hamish Cunningham, and Siegfried Handschuh. RoundTrip Ontology Authoring. In *International Semantic Web Conference*, volume 5318 of *Lecture Notes in Computer Science*, pages 50–65. Springer, 2008.
4. Ronald Denaux, Ian Holt, Ilaria Corda, Vania Dimitrova, Catherine Dolbear, and Anthony G. Cohn. ROO: A Tool to Assist Domain Experts with Ontology Construction. In Sean Bechhofer, Manfred Hauswirth, Jorg Hoffmann, and Manolis

[21] Importing an existing ontology will often result in loss of information, since some axioms might lie outside the coverage of the CNL; also, entity names will have to be derived from identifiers or labels which might be unsuitable, for instance because they are not English-based.

Koubarakis, editors, *Proceedings of the 5th European Semantic Web Conference*, 2008.
5. Li Ding, Tim Finin, Anupam Joshi, Rong Pan, R. Scott Cost, Yun Peng, Pavan Reddivari, Vishal Doshi, and Joel Sachs. Swoogle: a search and metadata engine for the semantic web. In *Proceedings of the 13th ACM International Conference on Information and Knowledge Management*, pages 652–659, New York, NY, USA, 2004. ACM.
6. Glen Hart, Martina Johnson, and Catherine Dolbear. Rabbit: Developing a control natural language for authoring ontologies. In Manfred Hauswirth, Manolis Koubarakis, and Sean Bechhofer, editors, *Proceedings of the 5th European Semantic Web Conference*, pages 348–360, 2008.
7. K. Kaljurand and N. Fuchs. Verbalizing OWL in Attempto Controlled English. In *Proceedings of OWL: Experiences and Directions*, Innsbruck, Austria, 2007.
8. Tobias Kuhn. How Controlled English can Improve Semantic Wikis. In Christoph Lange, Sebastian Schaffert, Hala Skaf-Molli, and Max Völkel, editors, *Proceedings of the Fourth Workshop on Semantic Wikis, European Semantic Web Conference 2009*, volume 464 of *CEUR Workshop Proceedings*. CEUR-WS, June 2009.
9. Tobias Kuhn. *Controlled English for Knowledge Representation*. PhD thesis, Faculty of Economics, Business Administration and Information Technology of the University of Zurich, 2010.
10. Chris Mellish and Xiantang Sun. The semantic web as a linguistic resource: Opportunities for natural language generation. *Knowledge-Based Systems*, 19(5):298–303, 2006.
11. Ontology Design Patterns. Repository of ontologies (NEON Project). http://ontologydesignpatterns.org/wiki/, 2010. Last accessed: 21st April 2010.
12. R. Power and D. Scott. Multilingual authoring using feedback texts. In *Proceedings of the 17th International Conference on Computational Linguistics and 36th Annual Meeting of the Association for Computational Linguistics*, pages 1053–1059, Montreal, Canada, 1998.
13. Richard Power. Complexity assumptions in ontology verbalisation. In *48th Annual Meeting of the Association for Computational Linguistics*, 2010.
14. Richard Power and Allan Third. Expressing OWL axioms by English sentences: dubious in theory, feasible in practice. In *Proceedings of the 23rd International Conference on Computational Linguistics*, 2010.
15. Stephen Pulman. Controlled language for knowledge representation. In *Proceedings of the first international workshop on controlled language applications*, Katholieke Universiteit Leuven, Belgium, 1996.
16. Alan Rector, Nick Drummond, Matthew Horridge, Jeremy Rogers, Holger Knublauch, Robert Stevens, Hai Wang, and Chris Wroe. OWL Pizzas: Practical Experience of Teaching OWL-DL: Common Errors and Common Patterns. In *14th International Conference on Knowledge Engineering and Knowledge Management*, pages 63–81, 2004.
17. Rolf Schwitter. Controlled natural languages for knowledge representation. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 1113–1121, 2010.
18. Rolf Schwitter. Processing Coordinated Structures in PENG Light. In *Australasian Conference on Artificial Intelligence*, pages 658–667, 2011.
19. Rolf Schwitter, Kaarel Kaljurand, Anne Cregan, Catherine Dolbear, and Glen Hart. A Comparison of three Controlled Natural Languages for OWL 1.1. In *4th OWL Experiences and Directions Workshop (OWLED 2008 DC)*, Washington, 1–2 April 2008.

20. Rolf Schwitter and Thomas Meyer. Sydney OWL Syntax - towards a Controlled Natural Language Syntax for OWL 1.1. In *OWLED: OWL Experiences and Directions*, 2007.
21. Robert Stevens, James Malone, Sandra Williams, Richard Power, and Allan Third. Automating generation of textual class definitions from OWL to English. *Journal of Biomedical Semantics*, Vol. 2 Suppl 2:S5, 2011.
22. TONES. Repository of ontologies at the University of Manchester. http://owl.cs.manchester.ac.uk/repository/browser, 2010. Last accessed: 21st April 2010.
23. Sandra Williams, Allan Third, and Richard Power. Levels of organisation in ontology verbalisation. In *Proceedings of the 13th European Workshop on Natural Language Generation*, pages 158–163, 2011.
24. Hui Yang, Anne N. De Roeck, Alistair Willis, and Bashar Nuseibeh. A methodology for automatic identification of nocuous ambiguity. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 1218–1226, 2010.