Grouping axioms for more coherent ontology descriptions

Grouping Axioms for More Coherent Ontology Descriptions

Sandra Williams
The Open University
Milton Keynes, United Kingdom
s.h.williams@open.ac.uk

Richard Power
The Open University
Milton Keynes, United Kingdom
r.power@open.ac.uk

Abstract

Ontologies and datasets for the Semantic Web are encoded in OWL formalisms that are not easily comprehended by people. To make ontologies accessible to human domain experts, several research groups have developed ontology verbalisers using Natural Language Generation. In practice ontologies are usually composed of simple axioms, so that realising them separately is relatively easy; there remains however the problem of producing texts that are coherent and efficient. We describe in this paper some methods for producing sentences that aggregate over sets of axioms that share the same logical structure. Because these methods are based on logical structure rather than domain-specific concepts or language-specific syntax, they are generic both as regards domain and language.

1 Introduction

When the Semantic Web becomes established, people will want to build their own knowledge bases (i.e., ontologies, or TBox axioms, and data, or ABox axioms[1]). Building these requires a high level of expertise and is time-consuming, even with the help of graphical interface tools such as Protégé (Knublauch et al., 2004). Fortunately, natural language engineers have provided a solution to at least part of the problem: verbalisers, e.g., the OWL ACE verbaliser (Kaljurand and Fuchs, 2007).

Ontology verbalisers are NLG systems that generate controlled natural language from Semantic Web languages, see Smart (2008). Typically they generate one sentence per axiom: for example, from the axiom[2] $Cat \sqsubseteq Animal$ the OWL ACE verbaliser (Kaljurand and Fuchs, 2007) generates 'Every cat is an animal'. The result is not a coherent text, however, but a disorganised list, often including inefficient repetitions such as:

> Every cat is an animal.
> Every dog is an animal.
> Every horse is an animal.
> Every rabbit is an animal.

An obvious first step towards improved efficiency and coherence would be to replace such lists with a single aggregated sentence:

> The following are kinds of animals: cats, dogs, horses and rabbits.

In this paper, we show how all axiom patterns in $\mathcal{EL}$++, a DL commonly used in the Semantic Web, can be aggregated *without further domain knowledge*, and describe a prototype system that performs such aggregations. Our method aggregates axioms *while they are still in logical form*, i.e., as part of sentence planning but before converting to a linguistic representation and realising as English sentences. This approach is somewhat different from that proposed by other researchers who convert ontology axioms to linguistic structures before aggregating (Hielkema, 2009; Galanis et al., 2009; Dongilli, 2008). We present results from testing our algorithm on over fifty ontologies from the Tones repository[3].

2 Analysis of Axiom Groupings

In this section we analyse which kinds of axioms might be grouped together. Power (2010) anal-

[1] Description Logic (DL) underlies the Web Ontology Language OWL. DL distinguishes statements about classes (TBox) from those about individuals (ABox). OWL covers both kinds of statements, which in OWL terminology are called 'axioms'.

[2] For brevity we use logic notation rather than e.g., OWL Functional Syntax: `subClassOf(class(ns:cat) class(ns:animal))` where `ns` is any valid namespace. The operator $\sqsubseteq$ denotes the subclass relation, $\sqcap$ denotes class intersection, and $\exists P.C$ the class of individuals bearing the relation P to one or more members of class C.

[3] http://owl.cs.manchester.ac.uk/

No.	Logic	OWL	%
1	$A \sqsubseteq B$	subClassOf(A B)	51
2	$A \sqsubseteq \exists P.B$	subClassOf(A someValuesFrom(P B))	33
3	$[a, b] \in P$	propertyAssertion(P a b)	8
4	$a \in A$	classAssertion(A a)	4

Table 1: The four most common axiom patterns.

ysed axiom patterns present in the same fifty ontologies. In spite of the richness of OWL, the surprising result was that *only four* relatively simple patterns dominated, accounting for 96% of all patterns found in more than 35,000 axioms. Overall there were few unique patterns, typically only 10 to 20, and up to 34 in an unusually complex ontology. Table 1 lists the common patterns in logic notation and OWL Functional Syntax, and also gives the frequencies across the fifty knowledge bases. Examples of English paraphrases for them are:

1. Every Siamese is a cat.
2. Every cat has as body part a tail.
3. Mary owns Kitty.
4. Kitty is a Siamese.

When two or more axioms conform to a pattern:

$$\begin{array}{l} A \sqsubseteq B \\ A \sqsubseteq C \\ B \sqsubseteq C \\ C \sqsubseteq D \end{array}$$

there are two techniques with which to aggregate them: merging and chaining. If the right-hand sides are identical we can merge the left-hand sides, and vice versa:[4]

$$\begin{array}{l} [A, B] \sqsubseteq C \\ A \sqsubseteq [B, C] \end{array}$$

Alternatively, where the right-hand side of an axiom is identical to the left-hand side of another axiom, we can 'chain' them:

$$A \sqsubseteq B \sqsubseteq C \sqsubseteq D$$

Merging compresses the information into a more efficient text, as shown in the introduction, while chaining orders the information to facilitate inference — for example, 'Every A is a B and every B is a C' makes it easier for readers to draw the inference that every A is a C.

[4]We regard expressions like $A \sqsubseteq [B, C]$ and $A \sqsubseteq B \sqsubseteq C$ as shorthand forms allowing us to compress several axioms into one formula. For merges one could also *refactor* the set of axioms into a new axiom: thus for example $A \sqsubseteq [B, C]$ could be expressed as $A \sqsubseteq (B \sqcap C)$, or $[A, B] \sqsubseteq C$ as $(A \sqcup B) \sqsubseteq C$. This formulation would have the advantage of staying within the normal notation and semantics of DL; however, it is applicable only to merges, not to chains.

	1.	2.	3.	4.
1.	L,R,C	×,R?,×	×,×,×	L?,×,C
2.		L,R,×	×,×,×	×,×,C
3.			L,R,×	×,R?,×
4.				L,R,×

Table 2: Aggregating common axioms: 1. $A \sqsubseteq B$, 2. $A \sqsubseteq \exists P.B$, 3. $[a, b] \in P$, 4. $a \in A$

Table 2 summarises our conclusions on whether each pair of the four common patterns can be merged or chained. Each cell contains three entries, indicating the possibility of left-hand-side merge (L), right-hand-side merge (R), and chaining (C). As can be seen, some merges or chains are possible across different patterns, but the safest aggregations are those grouping axioms with the same pattern (down the diagonal), and it is these on which we focus here.

3 Merging Similar Patterns

Function	Merge Patterns
$f_1(A)$	$f_1([A_1, A_2, A_3, \ldots])$
$f_2(A, B)$	$f_2([A_1, A_2, A_3, \ldots], B)$
	$f_2(A, [B_1, B_2, B_3, \ldots])$
$f_3(A, B, C)$	$f_3([A_1, A_2, A_3, \ldots], B, C)$
	$f_3(A, [B_1, B_2, B_3, \ldots], C)$
	$f_3(A, B, [C_1, C_2, C_3, \ldots])$

Table 3: Generic merging rules.

If we represent ABox and TBox axioms as Prolog terms (or equivalently in OWL Functional Syntax), they take the form of functions with a number of arguments — for example `subClassOf(A,B)`, where `subClassOf` is the functor, `A` is the first argument and `B` is the second argument. We can then formulate generic aggregation rules for merging one-, two- and three-argument axioms, as shown in table 3.

In general, we combine axioms for which the functor is the same and only one argument differs. We do not aggregate axiom functions with more than three arguments. The merged constituents must be different expressions with the same logical form.

4 Implemention

This section describes a Prolog application which performs a simple verbalisation including aggregation. It combines a generic grammar for realising logical forms with a domain-specific lexicon derived from identifiers and labels within the input ontology.

Input to the application is an OWL/XML file.[5] Axioms that conform to $\mathcal{EL}$++ DL are selected and converted into Prolog format. A draft lexicon is then built automatically from the identifier names and labels, on the assumption that classes are lexicalised by noun groups, properties by verb groups with valency two, and individuals by proper nouns.

Our aggregation rules are applied to axioms with the same logical form. The first step picks out all the logical patterns present in the input ontology by abstracting from atomic terms. The next step searches for all axioms matching each of the patterns present. Then within each pattern-set, the algorithm searches for axioms that differ by only one argument, grouping axioms together in the ways suggested in table 3. It exhaustively lists every possible grouping and builds a new, aggregated axiom placing the values for the merged argument in a list, e.g., consider the axioms:

```
subClassOf(class(cat), class(feline)).
subClassOf(class(cat), class(mammal)).
subClassOf(class(dog), class(mammal)).
subClassOf(class(mouse), class(mammal)).
```

Identical first arguments $\Longrightarrow$

```
subClassOf(class(cat),
           [class(feline),
            class(mammal)]).
'Every cat is a feline and a mammal.'
```

Identical second arguments $\Longrightarrow$

```
subClassOf([class(cat), class(dog),
class(mouse)], class(mammal)).
'The following are kinds of mammal:
      cats, dogs and mice.'
```

For all axioms with an identical first argument, `class(cat)`, the algorithm places the second arguments in a list, `[class(feline), class(mammal)]`, and builds a new axiom with the first argument and the merged second argument. From this, our realiser generates the sentence 'Every cat is a feline and a mammal.' A similar process is performed on first arguments when the second arguments are identical.

To construct the grammar, we first formulated rules for realising single axioms, and then added rules for the aggregated patterns, incorporating aggregation cues such as 'both' and 'the following:' (Dalianis and Hovy, 1996). For the wording of single axioms we relied mainly on proposals from the OWL Controlled Natural Language task force (Schwitter et al., 2008), so obtaining reasonably natural sentences for common axiom patterns, even though some less common axioms such as those describing attributes of properties (e.g., domain, range, functionality, reflexivity, transitivity) are hard to express without falling back on technical concepts from the logic of relations; for these we have (for now) allowed short technical formulations (e.g., 'The property "has as part" is transitive'). With these limitations, the grammar currently realises any single axiom conforming to $\mathcal{EL}$++, or any aggregation of $\mathcal{EL}$++ axioms through the merge rules described above. Table 4 lists example aggregated axiom patterns and English realisations generated with our grammar.

5 Testing the 'Merging' Algorithm

Unit	Original	Aggregated	Reduction
Sentences	35,542	11,948	66%
Words	320,603	264,461	18%

Table 5: Reduction achieved by aggregating

We have tested our generic merging rules on axioms conforming to $\mathcal{EL}$++ in a sample of around 50 ontologies. Table 5 shows the reduction in the number of generated sentence after aggregation. Remember that previously, the system generated one sentence for every axiom (35,542 sentences), but with aggregation this is reduced to 11,948 sentences, an overall reduction of 66%. However, aggregation increases sentence length so the saving in words is only 18%.

The effect of merging is to replace a large number of short sentences with a smaller number of longer ones. Sometimes the aggregated sentences were very long indeed, e.g., when a travel ontology cited 800 instances of the class `island` — perhaps such cases would be expressed better by a table than by prose[6].

The algorithm computes all possible merges, so we get, for instance, Fred described as a person in both 'The following are people: Fred, . . . ' and 'Fred is all of the following: a person, . . . '. This means that the greater efficiency achieved through aggregation may be counterbalanced by the extra text required when the same axiom participates in several merges — for a few of our ontologies, in

[5]We convert OWL to OWL/XML with the Manchester OWL Syntax Converter http://owl.cs.manchester.ac.uk/converter/

[6]In a summary one might instead simply give a count and an example: 'There are 800 islands, e.g., The Isle of Skye'.

Aggregated Axiom Pattern	**Example of Generated Text**
subClassOf([C_1,C_2,...], C_3).	The following are kinds of vehicles: a bicycle, a car, a truck and a van.
subClassOf(C_1, [C_2,C_3,...]).	Every old lady is all of the following: a cat owner, an elderly and a woman.
subClassOf([C_1,C_2,...],	The following are kinds of something that has as topping a tomato: a fungi,
objectSomeValuesFrom(P_1, C_3)).	a fiorella and a margherita.
subClassOf(C_1, [objectSomeValuesFrom(P_1, C_2)	Every fiorella is something that has as topping a mozzarella and is
objectSomeValuesFrom(P_2, C_3)]).	something that has as topping an olive.
classAssertion(C_1, [I_1, I_2, ...]).	The following are people: Fred, Joe, Kevin and Walt.
classAssertion([C_1,C_2,...], I).	Fred is all of the following: an animal, a cat owner and a person.
objectPropertyAssertion(P_1, [I_1, I_2, I_3], I_4).	The following are pet of Walt: Dewey, Huey and Louie.
objectPropertyAssertion(P_1, I_4, [I_1, I_2, I_3]).	Walt has as pet Dewey, Huey and Louie.
disjointClasses([C_1,C_2,...], C_3).	None of the following are mad cows: an adult, ... a lorry or a lorry driver.
disjointClasses(C_1, [C_2,C_3,...]).	No grownup is any of the following: a kid, a mad cow, a plant, or a tree.
dataPropertyDomain([P_1, P_2, ...], C_1).	If any of the following relationships hold between X and Y then X
	must be a contact: "has as city", "has as street" and "has as zip code".
dataPropertyRange([P_1, P_2, ...], C_1).	If any of the following relationships hold between X and Y then Y
	must be a string: "has as city", "has as e mail" and "has as street".
differentIndividuals(I_1, [I_2, I_3, ...]).	The One Star Rating is a different individual from any of the following:
differentIndividuals([I_1, I_2, ...], I_3).	the Three Star Rating or the Two Star Rating.
equivalentDataProperties(P_1, [P_2,P_3,...]).	The following properties are equivalent to the property "has as zip code":
equivalentDataProperties([P_1,P_2,...], P_3).	"has as post code", "has as zip" and "has as postcode".
equivalentObjectProperties([P_1,P_2,...], P_3).	The following properties are equivalent to the property "has as father": ....
negativeobjectPropertyAssertion(P_1, [I_1, I_2, ...], I_3).	None of the following are pet of Walt: Fluffy, Mog or Rex.
negativeobjectPropertyAssertion(P_1, I_1, [I_2, I_3, ...]).	It is not true that Walt has as pet Fluffy or Rex.

Table 4: Example realisations of common aggregated $\mathcal{EL}$++ axiom patterns.

fact, the word count for the aggregated version was *greater*. This is an interesting problem that we have not seen treated elsewhere. Merely pursuing brevity, one might argue that an axiom already included in a merge should be removed from any other merges in which it participates; on the other hand, the arbitrary exclusion of an axiom from a list might be regarded as misleading. For now we have allowed repetition, leaving the problem to future work.

6 Related Work

Reape and Mellish's (1999) survey of aggregation in NLG proposed a continuum of definitions ranging from narrow to wide. Our technique fits into the narrow definition, i.e., it is language-independent, operating on non-linguistic conceptual representations with the aim of minimising redundancy and repetition. It implements the subject and predicate grouping rules and aggregation cues suggested by Dalianis and Hovy (1996).

Recent NLG systems that aggregate data from ontologies (Hielkema, 2009; Galanis and Androutsopoulos, 2007; Dongilli, 2008) do not perform aggregation directly on axioms, but only *after* converting them to linguistic representations. Moreover, their systems generate only from ABox axioms in restricted domains while ours generates English for *both* ABox and TBox in *any domain*.

The approach most similar to ours is that of Bontcheva and Wilks (2004), who aggregate a subset of RDF triples after domain-dependent discourse structuring — a task equivalent to merging axioms that conform to the `objectPropertyAssertion` pattern in table 4.

7 Conclusion

We have demonstrated that for the $\mathcal{EL}$++ DL that underlies many Semantic Web ontologies we can define generic aggregation rules based on logical structure, each linked to a syntactic rule for expressing the aggregated axioms in English. The work described here is a first step in tackling a potentially complex area, and relies at present on several intuitive assumptions that need to be confirmed empirically. First, from an examination of all combinations of the four commonest axiom patterns, we concluded that axioms sharing the same pattern could be combined more effectively than axioms with different patterns, and therefore focussed first on same-pattern merges with variations in only one constituent. Secondly, after systematically enumerating all such merges for $\mathcal{EL}$++, we have implemented a grammar that expresses each aggregated pattern in English, relying on an intuitive choice of the best form of words: at a later stage we need to confirm that the resulting sentences are clearly understood, and to consider whether different formulations might be better.

Acknowledgments

This work is supported by the UK Engineering and Physical Sciences Research Council (EPSRC) grant EP/G033579/1 (SWAT: Semantic Web Authoring Tool). We thank our colleagues and the anonymous reviewers.

References

K. Bontcheva and Y. Wilks. 2004. Automatic report generation from ontologies: the MIAKT approach. In *Nineth International Conference on Applications of Natural Language to Information Systems (NLDB'2004)*, pages 214–225, Manchester, UK.

Hercules Dalianis and Eduard H. Hovy. 1996. Aggregation in natural language generation. In *EWNLG '93: Selected papers from the Fourth European Workshop on Trends in Natural Language Generation, An Artificial Intelligence Perspective*, pages 88–105, London, UK. Springer-Verlag.

Paolo Dongilli. 2008. Natural language rendering of a conjunctive query. Technical Report Knowledge Representation Meets Databases (KRDB) Research Centre Technical Report: KRDB08-3, Free University of Bozen-Bolzano.

Dimitrios Galanis and Ion Androutsopoulos. 2007. Generating multilingual descriptions from linguistically annotated OWL ontologies: the NaturalOWL system. In *Proceedings of the 11th European Workshop on Natural Language Generation*, pages 143–146, Morristown, NJ, USA. Association for Computational Linguistics.

Dimitrios Galanis, George Karakatsiotis, Gerasimos Lampouras, and Ion Androutsopoulos. 2009. An open-source natural language generator for OWL ontologies and its use in protégé, and second life. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics: Demonstrations Session*, pages 17–20, Morristown, NJ, USA. Association for Computational Linguistics.

Feikje Hielkema. 2009. *Using Natural Language Generation to Provide Access to Semantic Metadata*. Ph.D. thesis, University of Aberdeen.

Kaarel Kaljurand and Norbert Fuchs. 2007. Verbalizing OWL in Attempto Controlled English. In *Proceedings of the Third International Workshop on OWL: Experiences and Directions OWLED 2007*.

Holger Knublauch, Ray W. Fergerson, Natalya Fridman Noy, and Mark A. Musen. 2004. The Protégé OWL Plugin: An Open Development Environment for Semantic Web Applications. In *International Semantic Web Conference*, pages 229–243.

Richard Power. 2010. Complexity assumptions in ontology verbalisation. In *48th Annual Meeting of the Association for Computational Linguistics*.

Michael Reape and Chris Mellish. 1999. Just what is aggregation anyway? In *Proceedings of the 7th European Workshop on Natural Language Generation*, pages 20–29, Toulouse, France.

Rolf Schwitter, Kaarel Kaljur, Anne Cregan, Catherine Dolbear, and Glen Hart. 2008. A comparison of three controlled natural languages for owl 1.1. In *4th OWL Experiences and Directions Workshop (OWLED 2008)*.

Paul Smart. 2008. Controlled Natural Languages and the Semantic Web. Technical Report Technical Report ITA/P12/SemWebCNL, School of Electronics and Computer Science, University of Southampton),.