Generating Natural Language Descriptions from OWL Ontologies: A Detailed Presentation of the NaturalOWL System

Ion Androutsopoulos,[1,2] Gerasimos Lampouras[1] and Dimitrios Galanis[1]
[1] *Department of Informatics, Athens University of Economics and Business, Greece*
[2] *Digital Curation Unit – IMIS, Research Centre "Athena", Athens, Greece*

Abstract

We present NaturalOWL, an open-source natural language generation system that produces texts describing individuals (e.g., products, museum exhibits) or classes of individuals from OWL ontologies optionally associated with linguistic and user modeling resources. Unlike simpler OWL verbalizers, which typically express a single axiom of the ontology at a time in controlled and often not entirely fluent natural language mostly for the benefit of domain experts, we aim to generate fluent, coherent, and interesting multi-sentence texts appropriate for end-users (e.g., customers, museum visitors). A key benefit of using a system like NaturalOWL on the Semantic Web is that it becomes possible to publish information in the form of OWL ontologies, and rely on natural language generation to automatically produce personalized texts in multiple languages from the ontologies, thus making the information easily accessible not only to computer programs and domain experts, but also to end-users. We discuss NaturalOWL's processing stages of generating texts, the optional domain-specific linguistic and user modeling resources that can be used at each stage, why such resources are useful, and how they can be created and represented. We also report on trials we performed to measure the effort that is required to configure NaturalOWL for new ontologies, and the quality of the resulting texts.

Key words: Semantic Web; natural language generation; ontologies; OWL verbalizers; language resources; user modeling.

1. Introduction

The Semantic Web [20,155,6] is an effort to establish standards and mechanisms that will allow computers to reason more easily about the semantics of the Web's resources (documents, data etc.), enabling them and ultimately their users to share, locate, and integrate resources more easily. Ontologies play a central role in this endeavour. Each ontology provides a conceptualization of a knowledge domain (e.g., consumer electronics) by defining the classes and subclasses of the domain's individuals (entities), the types of possible relations between them etc.

The current standard to specify ontologies for the Semantic Web is OWL, a formal language based on description logics [8], RDF, and RDF SCHEMA, with OWL2 being OWL's latest version [65].[1] Given an OWL ontology for a knowledge domain, it is possible to publish machine-readable datasets pertaining to that domain (e.g., catalogues of products, their prices, features etc.) on the Web, with the data having formally defined semantics based on the ontology's conceptualization. It is also possible to mark up Web resources (e.g., documents) with rich machine-readable meta-data, again with formally defined semantics, to describe their authors, content etc. In both cases, the datasets or meta-data provide instances of the ontology's concepts (e.g., particu-

[1] Unless otherwise stated, we adopt OWL2; consult `http://www.w3.org/TR/2009/REC-owl2-primer-20091027/` for an introduction. For background information on RDF and RDF SCHEMA, see `http://www.w3.org/TR/rdf-primer/`.

lar individuals of its classes, particular instances of its relation types). Extensions of existing OWL ontologies may also be published, for example to define finer classes, or to combine concepts from several ontologies. Reasoning engines for OWL are also available [68,163,156,129], and they can be used, for example, to deduce additional information.

Following common practice in Semantic Web research, we use the term *ontology* to refer jointly to (a) information that establishes a conceptualization of a knowledge domain, often called *terminological knowledge* or *TBox*, and (b) additional *assertional knowledge*, *ABox*, that describes particular instances of the domain's concepts. OWL can express knowledge of both types. Several semantically equivalent OWL syntaxes have been developed. People unfamiliar with formal knowledge representation, however, often have difficulties understanding them [143,53]. For example, the following statement defines the class of St. Emilion wines, using OWL's functional-style syntax, one of the easiest to understand, which we also adopt throughout this article. [2]

```
EquivalentClasses(:StEmilion
  ObjectIntersectionOf(:Bordeaux
    ObjectHasValue(:locatedIn :stEmilionRegion)
    ObjectHasValue(:hasColor :red)
    ObjectHasValue(:hasFlavor :strong)
    ObjectHasValue(:madeFromGrape
      :cabernetSauvignonGrape)
    ObjectMaxCardinality(1 :madeFromGrape)))
```

The statement above defines `StEmilion` as the intersection of: (i) the class of `Bordeaux` wines; (ii) the class of all individuals whose `locatedIn` property has (for each individual) `stEmilionRegion` among its values (OWL properties are generally many-valued); (iii)–(v) the classes of individuals whose `hasColor`, `hasFlavor`, and `madeFromGrape` property values include `red`, `strong`, and `cabernetSauvignonGrape`, respectively, without excluding wines that have additional values in these properties; and (vi) the class of individuals whose `madeFromGrape` property has exactly one value; hence, a St. Emilion wine is made *exclusively* from Cabernet Sauvignon grapes.

In this article, we provide a detailed description of an open-source natural language generation (NLG) system, called NaturalOWL, that produces texts describing classes or individuals of OWL ontologies. [3] For example, the system can automatically generate the following text from the OWL statement above, if the ontology has been annotated with domain-specific linguistic resources (e.g., lexicon entries, sentence plans) that we discuss below.

> St. Emilion is a kind of Bordeaux from the St. Emilion region. It has red color and strong flavor. It is made from exactly one grape variety: Cabernet Sauvignon grapes.

NaturalOWL currently supports both English and Greek. [4] Hence, Greek texts can also be generated from the same ontology, provided that it has also been annotated with appropriate Greek linguistic resources. For example, given the following product description in OWL, the English and Greek texts shown below can be generated.

```
ClassAssertion(:Laptop :tecraA8)
ObjectPropertyAssertion(:manufacturedBy
  :tecraA8 :toshiba)
ObjectPropertyAssertion(:hasProcessor
  :tecraA8 :intelCore2)
DataPropertyAssertion(:hasMemoryInGB
  :tecraA8 "2"^^xsd:nonNegativeInteger)
DataPropertyAssertion(:hasHardDiskInGB
  :tecraA8 "110"^^xsd:nonNegativeInteger)
DataPropertyAssertion(:hasSpeedInGHz
  :tecraA8 "2"^^xsd:float)
DataPropertyAssertion(:hasPriceInEuro
  :tecraA8 "850"^^xsd:nonNegativeInteger)
```

> Tecra A8 is a laptop, manufactured by Toshiba. It has an Intel Core 2 processor, 2 GB RAM and an 110 GB hard disk. Its speed is 2 GHz and it costs 850 Euro.

> Ο Tecra A8 είναι ένας φορητός υπολογιστής, κατασκευασμένος από την Toshiba. Διαθέτει επεξεργαστή Intel Core 2, 2 GB RAM και σκληρό δίσκο 110 GB. Η ταχύτητά του είναι 2 GHz και κοστίζει 850 Ευρώ.

For readers unfamiliar with OWL, we note that an *object property* (e.g., `manufacturedBy`) maps an individual (e.g., `tecraA8`) to one or more individuals (e.g., `toshiba`), whereas a *datatype property* (e.g., `hasMemoryInGB`) maps an individual to one or more datatype values (e.g., strings, numbers). We follow the convention that class identifiers start with capitals, whereas the identifiers of individuals and properties start with lower-case letters. To save space, we omit the definitions of common namespaces; for ex-

[2] Consult `http://www.w3.org/TR/owl2-syntax/` for more information on OWL's functional-style syntax. NaturalOWL actually uses OWL's RDF/XML syntax, which can be converted to functional-style and vice versa; see, for example, `http://owl.cs.manchester.ac.uk/converter/`. We often use examples loosely inspired from the Wine Ontology, which we discuss in Section 4.1.

[3] A much shorter description of an earlier version of NaturalOWL has also been published [60]. An earlier version of the system has been demonstrated at major conferences [86,61].

[4] We welcome collaborations to support other languages.

ample, `xsd` is a common abbreviation for the namespace `http://www.w3.org/2001/XMLSchema`.

The examples above illustrate how NLG can help publish information on the Web both as OWL statements and as texts generated from the OWL statements; the generated texts can also be turned into HTML pages, including images of the individuals etc. This way, information becomes easily accessible to both computer applications, which can process the OWL statements, and end-users speaking different languages. Changes in the OWL statements can be automatically reflected in the texts by regenerating them. Other well-known benefits of NLG include the ability to tailor the texts per user type. For example, in a machine-generated text describing a medicine [30], doctors may wish to see more specialized information than non-experts, who may need texts with more background knowledge, expressed in simpler terms. The texts can also vary depending on the user's interaction history, for example to avoid repeating information or to include comparisons to previously encountered individuals. [5]

Although NLG from symbolic information is an established area [116,145,16,100], NaturalOWL is currently one of the very few NLG systems for OWL; we compare to related systems below. We also note that NaturalOWL aims to generate fluent, coherent, and interesting multi-sentence texts appropriate for end-users (e.g., museum visitors, customers of online shops), unlike simpler systems often called *ontology verbalizers* [151]. The latter usually translate the ontology's axioms (in our case, OWL statements) one by one to controlled and often not entirely fluent English statements, typically without considering the coherence of the resulting texts, the interests of the readers, or their interaction histories, and mostly for the benefit of domain experts. Conveying the exact meaning of all the axioms is considered more important in ontology verbalizers than composing fluent, coherent, and interesting texts. By contrast, NaturalOWL may not express information that a particular end-user presumably already knows, and it may opt for more fluent, but technically less precise phrasings. Furthermore, it includes mechanisms to order sentences, aggregate them into longer ones, generate referring expressions etc. We return to OWL verbalizers below, where we also compare NaturalOWL's texts to those of a verbalizer.

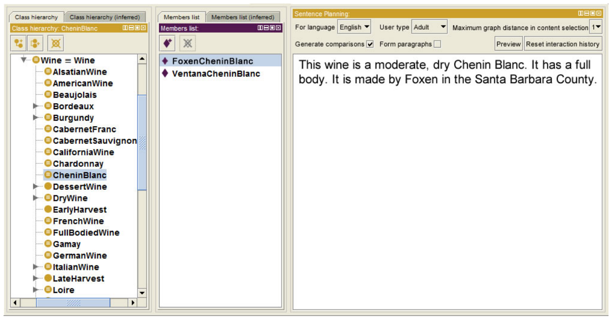
NaturalOWL is intended to generate texts describing individuals (or classes of individuals) such as physical objects, organizations, or persons. It is not intended to describe sequences of events [10,91], nor to produce instructions to perform tasks [75,76,167], or reports from numerical data [179,17]. Also, we do not discuss generating *spoken* descriptions [162], although NaturalOWL was recently used in spoken dialogues with robotic museum guides [96,169,97]. Our work is based on ideas from ILEX [132] and M-PIRO [82]. The ILEX project developed an NLG system that was demonstrated mostly with museum exhibits, but did not support OWL; Dale *et al.* [38] and Dannels [43] also discuss NLG for museums. The M-PIRO project produced a multilingual extension of ILEX's system, which was tested in several domains, including museum exhibits and computing equipment [4]. Attempts to use M-PIRO's generator with OWL ontologies, however, ran into problems, because of incompatibilities between OWL and M-PIRO's ontological model [2]. By contrast, NaturalOWL was especially developed for OWL.

NaturalOWL can be used with different domain ontologies, but the resulting texts may not sound fluent, coherent, or interesting enough, until appropriate domain-specific linguistic and user modeling resources, hereafter called *domain-dependent generation resources*, have been created. For example, the ontology's classes can be mapped to natural language names, the ontology's properties to sentence plans, other resources may indicate which sentence plans to prefer per user type etc. The domain-dependent generation resources are created by a person we call the *domain author*, when the system is configured for a new ontology. Similar resources are used in most NLG systems for symbolic information. NaturalOWL's domain-dependent generation resources can be created using a plug-in of the Protégé ontology editor. [6] The plug-in also allows NaturalOWL to be invoked within Protégé (Figure 1). We do not discuss the plug-in in any detail in this article, since it is very similar to M-PIRO's authoring tool, which has been presented elsewhere [4]. [7]

[5] We do not discuss how NaturalOWL generates comparisons in this article, but see Milosavljevic [126], Isard [81], Karakatsiotis [85], and Marge *et al.* [115] for related work.

[6] See `http://protege.stanford.edu/` for information on Protégé. NaturalOWL and its Protégé plug-in are freely available from `http://nlp.cs.aueb.gr/software.html`.

[7] We describe NaturalOWL version 2 in this article. Version 1 used a less principled representation of its domain-dependent generation resources. M-PIRO's authoring tool, now called ELEON [23], can currently be used only with NaturalOWL version 1; see `http://users.iit.demokritos.gr/~eleon/`.



For language: English
User type: Adult
Maximum graph distance in content selection: 1
Generate comparisons ✓
Form paragraphs ☐

This wine is a moderate, dry Chenin Blanc. It has a full body. It is made by Foxen in the Santa Barbara County.

Figure 1. Invoking NaturalOWL within Protégé. An English description of an individual has been generated.

Different NLG systems often adopt different linguistic theories, generation algorithms, and user models, thus requiring different domain-dependent generation resources. There is currently little consensus on exactly what information the domain-dependent generation resources should capture, or how this information should be represented. In the spirit of the Semantic Web, however, it seems reasonable to expect that when generating texts from OWL ontologies, the domain-dependent generation resources themselves should be instances of other OWL ontologies, with each ontology of the latter kind defining the types of resources required by a particular NLG system. To avoid confusion, we hereafter use the term *domain ontology* to refer to the ontology whose individuals and classes are described, as opposed to the *generation resources ontology* that defines the types of domain-dependent generation resources a particular NLG system, in our case NaturalOWL, requires.[8] By representing the domain-dependent generation resources in OWL, it becomes easier to publish them on the Web, reuse them, check them for inconsistencies using OWL reasoners etc. as with other OWL ontologies.

OWL ontologies often use English words or concatenations of words (e.g., `manufacturedBy`) as identifiers of classes, properties, and individuals. Hence, some of the domain-dependent generation resources can often be extracted from the domain ontologies. For example, Sun and Mellish [159,160] use WordNet [56] and some heuristics to split the identifiers of classes and properties into English words, and to assign them part-of-speech (POS) tags. This allows guessing that a class identifier like `Laptop` in our earlier example is a noun that can be used to refer to that class, or that a statement of the form:

`ObjectPropertyAssertion(:manufacturedBy` X Y`)`

should be expressed in English as a sentence of the form "X was manufactured by Y". Most OWL verbalizers follow the same strategy.

If domain-dependent generation resources are not provided, NaturalOWL adopts a similar behavior. It attempts to guess the resources from the domain ontology, or it uses generic resources. The resulting texts, however, are of lower quality, compared to those generated with appropriate domain-dependent generation resources, and closer to the output of ontology verbalizers. For example, the laptop description of page 2 becomes:

> Tecra A 8 is a laptop. Tecra A 8 manufactured by Toshiba. Tecra A 8 has processor Intel Core 2. Tecra A 8 has memory in GB 2. Tecra A 8 has hard disk in GB 110. Tecra A 8 has speed in GHz 2. Tecra A 8 has price in Euro 850.

The sentences are now less fluent, because they are based on templates (e.g., "X has price in Euro Y") extracted directly from the domain ontology, instead of better sentence plans that would have been specified by a domain author (e.g., "X costs Y Euro"). Sentences with the same subject and verb are no longer aggregated, because NaturalOWL does not know which words are verbs, information that would be present in the sentence plans and lexicon entries provided by the author. Without appropriate lexicon entries, the system also no longer knows the gender of laptops; hence, it refrains from using personal pronouns (e.g., "it"). Furthermore, Greek texts cannot be generated, because the domain ontology's identifiers are English-like.

Extracting domain-dependent generation resources from the domain ontology reduces the effort needed to configure NaturalOWL for a new domain; hence, this approach is appealing when the texts do not need to be particularly fluent, coherent, nor personalized, and a single language suffices. On the other hand, most of NaturalOWL's domain-dependent generation resources are associated with classes and properties of the domain ontology (TBox), rather than individuals (ABox);

[8] See also Cimiano *et al.* [34], Montiel-Ponsoda *et al.* [127], and Gracia *et al.* [64] for discussion on associating liguistic resources with ontologies. Mellish [119] suggests representing in OWL the abstract NLG architecture of RAGS [123]. The resources of every RAGS-compliant NLG system could then be represented in OWL, using specializations of abstract concepts from RAGS's architecture. We note, however, that the RAGS architecture is deliberatively very high level, and substantial work would be necessary to establish exactly how the processing stages and resources of NaturalOWL or other NLG systems relate to RAGS concepts.

consequently, when there are relatively few classes and properties, many more individuals, and the texts need to be fluent, coherent, personalized, or in multiple languages, authoring domain-dependent generation resources may be preferable. There is a tradeoff between investing less effort to construct domain-dependent generation resources and obtaining higher-quality texts in multiple languages.

To the best of our knowledge, this article is the first detailed description of a complete general-purpose NLG system for OWL, excluding simpler ontology verbalizers. The article also discusses trials we conducted with two independently created OWL domain ontologies to assess our system's portability to ontologies created by others, the effort that is required to construct NaturalOWL's domain-dependent generation resources, and the gain in text quality, as opposed to using a verbalizer or NaturalOWL without domain-dependent generation resources. We believe these trials are also novel.

Overall, the main contributions of this article are: (i) it is the first (excluding verbalizers) detailed description of a complete, general-purpose, open-source NLG system for OWL domain ontologies; (ii) it shows how the same system can produce rough-quality texts for end-users directly from OWL domain ontologies, but also how additional domain-dependent generation resources can improve the quality of the texts, help personalize them, and generate them in multiple languages; (iii) it shows how the domain-dependent generation resources themselves can be represented in OWL; (iv) it reports on trials with two independently created OWL domain ontologies, where the texts of our system, with and without domain-dependent generation resources, were compared against those of a simpler verbalizer. The article does not present novel algorithms from a theoretical NLG perspective. In fact, some of the algorithms that NaturalOWL employs are of a much narrower scope, compared to more fully-fledged NLG algorithms. Nevertheless, the trials show that the system produces texts of reasonable quality, especially when domain-dependent generation resources are provided. We hope that if NaturalOWL contributes towards a wider adoption of NLG methods on the Semantic Web, other researchers may wish to develop improved components of the system, based on more elaborate algorithms. [9] To this end, we propose several ways to extend the current system, also pointing to the prominent relevant literature.

Section 2 below explains how NaturalOWL generates texts; it also discusses the domain-dependent generation resources. Section 3 presents related work. Section 4 describes the trials we performed. Section 5 concludes and proposes future work.

2. Processing stages and resources

NaturalOWL adopts a pipeline architecture, which is common in NLG [145], though the number and exact purpose of the pipelined components often vary [123]. Our system generates texts in three stages, *document planning*, *micro-planning*, and *surface realization*, discussed in the following sections.

2.1. *Document planning*

Document planning consists of two sub-stages: *content selection*, where the system selects the logical facts to convey, and *text planning*, where it plans the structure of the text to be generated.

2.1.1. *Content selection*

In content selection, the system retrieves from the domain ontology candidate facts, it converts them to message triples, which are easier to express as sentences, and it then selects among the candidate message triples the ones to be expressed.

Candidate facts for individual targets

Let us first consider content selection when NaturalOWL is asked to describe an *individual* (an entity), and let us call *target* the individual being described. The system scans the OWL statements of the domain ontology, looking for statements of the forms listed in Table 1, which are considered *candidate facts*, i.e., facts that could be mentioned. [10] We have already used most kinds of statements of Table 1 in the laptop example of page 2; `SameIndividual(X Y)` and `DifferentIndividuals(X Y)` signal that `X` and `Y` denote the same or different individuals. [11] Negative property assertions (e.g., stating that a particular laptop is *not* manufactured by a particular company)

[9] A collaborative source code repository for NaturalOWL is available at `http://naturalowl.sourceforge.net/`.

[10] NaturalOWL scans the domain ontology using OWL API; consult `http://owlapi.sourceforge.net/`.

[11] These statements and some others that we show as having two arguments can actually have more arguments, but they can be converted to binary statements.

✓	ClassAssertion(*Class target*)
✓	ObjectPropertyAssertion(*objProp target indiv*)
✓	DataPropertyAssertion(*dataProp target dataValue*)
×	negative property assertions
✓	DifferentIndividuals(*target indiv*)
✓	DifferentIndividuals(*indiv target*)
✓	SameIndividual(*target indiv*)
✓	SameIndividual(*indiv target*)

Table 1
OWL statements used (✓) or not (×) as candidate facts, when generating a text for an **individual** whose identifier is *target*. *Class* is either a class identifier or an expression constructing an unnamed class using the operators of Table 2; *objProp* and *dataProp* stand for object and datatype properties, respectively; *indiv* is the identifier of an individual; and *dataValue* is a datatype value.

are also allowed in OWL2, but they are currently not supported by NaturalOWL; they are also rarely used.

We note that ClassAssertion(*Class target*) statements may be quite complex, because *Class* is not necessarily a class name. It may also be an expression that constructs a new, unnamed class using operators from Table 2, as in the following example.

```
ClassAssertion(
 ObjectIntersectionOf(:Wine
  ObjectHasValue(:locatedIn :stEmilionRegion)
  ObjectHasValue(:hasColor :red)
  ObjectHasValue(:hasFlavor :strong)
  ObjectHasValue(:madeFromGrape
   :cabernetSauvignonGrape)
  ObjectMaxCardinality(1 :madeFromGrape))
 :chateauTeyssier2007)
```

The statement above says that chateauTeyssier2007 is an individual wine with the same characteristics as the StEmilion wines of page 2, except that it is not necessarily a Bordeaux. Notice that here the statement mentions the broader class Wine, instead of its subclass Bordeaux; it also uses an unnamed class as the first argument of the ClassAssertion. Assuming that the target is chateauTeyssier2007, NaturalOWL could express the OWL statement above by generating a text like the following. The exact text would depend on the domain-dependent generation resources.

> The 2007 Chateau Teyssier is a wine from the Saint-Emilion region. It has red color and strong flavor. It is made from exactly one grape variety: Cabernet Sauvignon grapes.

Notice that a single ClassAssertion gives rise to multiple sentences. The order of the sentences is not necessarily the same as that of the corresponding OWL expressions inside the ClassAssertion; sentences are ordered during text planning.

	(i) Multi-class operators (MultiClassOper):
✓	ObjectIntersectionOf({*NamedClass* | *SingleClassOper*}$^+$)
×	ObjectIntersectionOf(... *MultiClassOper* ...)
✓	ObjectUnionOf({*NamedClass* | *SingleClassOper*}$^+$)
×	ObjectUnionOf(... *MultiClassOper* ...)
	(ii) Single-class operators (SingleClassOper):
✓	ObjectComplementOf(*NamedClass*)
×	ObjectComplementOf(*UnnamedClass*)
✓	ObjectOneOf(*indiv*$^+$)
✓	ObjectHasValue(*objProp indiv*)
✓	ObjectHasValue(*dataProp dataValue*)
✓	ObjectHasSelf(*objProp*)
✓	ObjectMaxCardinality(*number prop* [*NamedClass*])
×	ObjectMaxCardinality(*number prop UnnamedClass*)
✓	ObjectMinCardinality(*number prop* [*NamedClass*])
×	ObjectMinCardinality(*number prop UnnamedClass*)
✓	ObjectExactCardinality(*number prop* [*NamedClass*])
×	ObjectExactCardinality(*number prop UnnamedClass*)
✓	ObjectSomeValuesFrom(*objProp NamedClass*)
×	ObjectSomeValuesFrom(*objProp UnnamedClass*)
✓	ObjectAllValuesFrom(*objProp NamedClass*)
×	ObjectAllValuesFrom(*objProp UnnamedClass*)

Table 2
OWL operators that may (✓) or may not (×) be used to **construct unnamed classes** when using NaturalOWL. Expressions marked with '$^+$' may be repeated, '|' denotes a disjunction, and square brackets indicate optional arguments. *NamedClass* is a class identifier; *UnnamedClass* is an expression constructing an unnamed class; *MultiClassOper* and *SingleClassOper* are expressions starting with operators of the corresponding types; *objProp* is an object property, *dataProp* a datatype property, *prop* any property, *indiv* an individual's identifier, and *dataValue* a datatype value.

OWL allows arbitrarily many nested ObjectUnionOf (class union) and ObjectIntersectionOf (class intersection) operators, which may lead to statements that are very difficult to express in natural language. To simplify text generation and to ensure that the resulting texts are reasonably easy to comprehend, we do not allow nested ObjectIntersectionOf and ObjectUnionOf operators in the domain ontologies that NaturalOWL is used to generate texts from. In Table 2 (part i), this restriction is enforced by not allowing what we call *multi-class operators* (operators with more than one class arguments), namely `ObjectIntersectionOf` and `ObjectUnionOf`) inside the arguments of other multi-class operators. For the same reasons, i.e., to simplify the generation process and produce texts that are easier to comprehend, we do not allow unnamed classes inside some other operators, as shown again in Table 2 (part ii). If a domain-ontology violates the constraints of Table 2, it can be easily modified to comply with the constraints by defining new named classes for nested unnamed classes. The OWL ontologies we have encountered so far did not violate the constraints. See Power *et al.* [137,141] for a study of the OWL statements ontology authors use most frequently.

For the benefit of readers unfamiliar with OWL, let us quickly explain the operators of Table 2 we have not used so far. `ObjectComplementOf` constructs the class of all individuals *not* in the original class. `ObjectOneOf` constructs a class by enumerating all of its individuals. `ObjectHasValue(`*objProp indiv*`)` denotes the class of individuals that have *indiv* among the values of their *objProp* property; recall that OWL's properties are generally many-valued. As a special case, `ObjectHasSelf(`*objProp*`)` is the class of individuals that have themselves as one of the (possibly many) values of their *objProp*. `ObjectMaxCardinality(`*number prop*`)` denotes the class of individuals that have (each) no more than *number* values in their *prop* property. `ObjectMinCardinality` and `ObjectExactCardinality` are similar, but they specify the minimum or exact number of values. In all three cardinality operators, an optional third argument specifying a class may be present; then the restriction refers to values from the third argument's class. For example, the following is the class of individuals that are made from at most one Italian grape and any number of other grapes.

```
ObjectMaxCardinality(1 :madeFromGrape :ItalianGrape)
```

Finally, `ObjectSomeValuesFrom(`*objProp Class*`)` is the class of individuals that have at least one member of *Class* among the values of their *objProp*. For example, the following is the class of individuals that manufacture (each) at least one laptop.

```
ObjectSomeValuesFrom(:manufactures :Laptop)
```

Similarly, `ObjectAllValuesFrom(`*objProp Class*`)` is the class of individuals that do not have values outside *Class* in their *objProp*, including individuals that have no values at all in *objProp*.

Recall that NaturalOWL's texts are intended to be read by end-users. Hence, we prefer to generate texts that may not emphasize enough, from a knowledge representation point of view, some of the subtleties of OWL's statements, in order to produce more readable texts. An OWL expert might prefer, for example, the following natural language description of `chateauTeyssier2007`, which mirrors more closely the corresponding OWL statements, than the text of page 6. The text below also makes it clearer that, in the absence of other information, nothing rules out the possibility that `chateauTeyssier2007` may, for example, have both a strong and another flavor.

> The 2007 Chateau Teyssier is a member of the intersection of the following classes: (a) the class of wines, (b) the class of individuals from (not necessarily exclusively) the St. Emilion region, (c) the class of individuals that have (not necessarily exclusively) red color, (d) the class of individuals that have (not necessarily exclusively) strong flavor, (e) the class of individuals that are made exclusively from Cabernet Sauvignon grapes.

Stricter texts of this kind, however, seem inappropriate for end-users. In fact, it could be argued that mentioning that the 2007 Chateau Teyssier is made from exactly one grape variety in the text of page 6 is also inappropriate for end-users. NaturalOWL can be instructed to avoid mentioning this information via user modeling annotations, discussed below.

Candidate facts for class targets

We have so far discussed how NaturalOWL selects candidate facts when asked to describe an individual. If the system is asked to describe a *class*, it scans the domain ontology for statements of the forms of Table 3; the class to be described must be a named one, meaning that it must have an OWL identifier, and *Target* is its identifier.[12] A statement of the form `DisjointClasses(`*Class1 Class2*`)` signals that *Class1* and *Class2* can never have a common individual. Again, to simplify the generation process and to avoid producing complicated texts, we require both arguments of `DisjointClasses` to be named classes. Statements violating this restriction can be transformed by defining new named classes.

In texts describing classes, it is difficult to express informally the difference between `EquivalentClasses` and `SubClassOf`. `EquivalentClasses(`*Class1 Class2*`)`

[12] `SubClassOf(`*NamedClass Target*`)` statements could also be selected as candidate facts. They would lead to sentences naming *subclasses* of *Target*, as in the following text where the target is the class of vases: "A vase was used to store food or liquids. Known types of vases are: hydriae, amphorae, and lekythoi." Similarly, known individuals of a target class (or some of them) could be mentioned.

√ EquivalentClasses(*Target Class*)
√ EquivalentClasses(*Class Target*)
√ SubClassOf(*Target Class*)
√ DisjointClasses(*Target NamedClass*)
× DisjointClasses(*Target UnnamedClass*)
√ DisjointClasses(*NamedClass Target*)
× DisjointClasses(*UnnamedClass Target*)

Table 3
OWL statements used (√) or not (×) as candidate facts when generating a text for a **class** whose identifier is *Target*. *Class* can be either a class identifier or an unnamed class constructed with the operators of Table 2. Statements marked with '×' can be tranformed to use named classes.

means that any individual of *Class1* also belongs in *Class2*, and vice versa. By contrast, SubClassOf(*Class1 Class2*) means that any member of *Class1* also belongs in *Class2*, but the reverse is not necessarily true. If we replace EquivalentClasses by SubClassOf in the definition of StEmilion of page 2, as shown below, then any member of StEmilion is still necessarily also a member of the intersection, but a wine with all the characteristics of the intersection is not necessarily a member of StEmilion.

```
SubClassOf(:StEmilion
  ObjectIntersectionOf(:Bordeaux
    ObjectHasValue(:locatedIn :stEmilionRegion)
    ObjectHasValue(:hasColor :red)
    ObjectHasValue(:hasFlavor :strong)
    ObjectHasValue(:madeFromGrape
      :cabernetSauvignonGrape)
    ObjectMaxCardinality(1 :madeFromGrape)))
```

Consequently, one should perhaps add sentences like the ones shown in italics below, when expressing EquivalentClasses and SubClassOf, respectively. [13]

St. Emilion is a kind of Bordeaux from the St. Emilion region. It has red color and strong flavor. It is made from exactly one grape variety: Cabernet Sauvignon grapes. *Every St. Emilion has these properties, and anything that has these properties is a St. Emilion.*

St. Emilion is a kind of Bordeaux from the St. Emilion region. It has red color and strong flavor. It is made from exactly one grape variety: Cabernet Sauvignon grapes. *Every St. Emilion has these properties, but something may have these properties without being a St. Emilion.*

[13] Power and Third [141] use "X is defined as" with EquivalentClasses, and presumably expressions like "every X is a" with SubClassOf. Cregan *et al.* [36] use "is fully defined as" and "is partly defined as", respectively. We suspect that the difference would still be unclear to end-users.

Although it is trivial to add the sentences shown in italics, NaturalOWL currently produces the same texts, without the sentences in italics, for both EquivalentClasses and SubClassOf, in order to avoid generating texts that sound too formal. Furthermore, the system may not mention some of the domain ontology's information about a target class (e.g., that a St. Emilion has strong flavor), when user modeling indicates that this information is already known or that the text should not exceed a particular length. Hence, strictly speaking the resulting texts generally express *necessary*, not *sufficient* conditions for individuals to belong in the target class. This behavior is consistent with the fact that when asked to describe an individual, rather than a class, the system does not necessarily convey all the information it knows about the individual, in order to comply with space constraints and other user modeling requirements.

Additional candidate facts

In some applications, expressing additional facts that are *indirectly* related to the target may be desirable. Let us assume, for example, that the target is an individual whose identifier is exhibit24, and that the content selection mechanisms discussed above have retrieved the following candidate facts from the domain ontology. NaturalOWL would express them by generating a text like the one below.

```
ClassAssertion(:Aryballos :exhibit24)
ObjectPropertyAssertion(:locationFound
  :exhibit24 :heraionOfDelos)
ObjectPropertyAssertion(:creationPeriod
  :exhibit24 :archaicPeriod)
ObjectPropertyAssertion(:paintingTechniqueUsed
  :exhibit24 :blackFigureTechnique)
ObjectPropertyAssertion(:currentMuseum
  :exhibit24 :delosMuseum)
```

This is an aryballos, found at the Heraion of Delos. It was created during the archaic period and it was decorated with the black-figure technique. It is currently in the Museum of Delos.

The natural language names of classes and individuals can be shown as hyperlinks to indicate that they can be used as subsequent targets. Clicking on a hyperlink would be a request to describe the corresponding class or individual. [14] Alternatively, we may retrieve the candidate facts of these possi-

[14] Similar hyperlinks were used, for example, in demonstrators of ILEX and M-PIRO [132,82,4], the system described by Dale *et al.* [38], and that of Halaschek-Wiener *et al.* [69]. They are also used by some OWL verbalizers [178].

ble subsequent targets in advance and add them to those of the current target.

More precisely, assuming that the target is an individual, the subsequent possible targets, called *second-level targets*, are taken to be the target's class, provided that it is a named one, and the individuals the target is directly linked to via object properties. NaturalOWL considers second-level targets only when the current target is an individual, because expressing information about possible subsequent targets when the current target is a class often leads to complicated texts. The system can be set to retrieve candidate facts for both the current and the second-level targets (when applicable), or only for the current target; we say that the *maximum fact distance* is two or one, respectively.

Returning to `exhibit24`, let us assume that the maximum fact distance is two and that only the following candidate facts for second-level targets are available; we explain the OWL statements below.[15]

```
SubClassOf(:Aryballos :Vase)
SubClassOf(:Aryballos
  ObjectHasValue(:exhibitTypeCannedDescription
    "An aryballos was a small spherical vase with a
    narrow neck, in which the athletes kept the oil
    they spread their bodies with"^^xsd:string))
DatatypePropertyAssertion(:periodDuration
  :archaicPeriod "700 BC to 480 BC"^^xsd:string)
DatatypePropertyAssertion(:periodCannedDescription
  :archaicPeriod "The archaic period was when the
  Greek ancient city-states developed"^^xsd:string)
DataPropertyAssertion(:techniqueCannedDescription
  :blackFigureTechnique "In the black-figure
  technique, the silhouetes are rendered in black
  on the pale surface of the clay, and details are
  engraved"^^xsd:string)
```

Notice that we associated the `Aryballos` class with the string "An aryballos... bodies with" by using an `ObjectHasValue` inside a `SubClassOf` statement; in effect, the statement says that every individual of the `Aryballos` class has that string as the value of its `exhibitTypeCannedDescription` property. We could not have associated the class directly with the string via a property assertion, as we did with the `archaicPeriod` and `blackFigureTechnique` individuals, because OWL's properties map only from individuals, not classes, to datatype values or other individuals.

To express all the candidate facts, NaturalOWL would now generate a text like the following, which may be preferable, if this is the first time the user encounters an aryballos and archaic exhibits.

This is an aryballos, a kind of vase. An aryballos was a small spherical vase with a narrow neck, in which the athletes kept the oil they spread their bodies with. This particular aryballos was found at the Heraion of Delos and it was created during the archaic period. The archaic period was when the Greek ancient city-states developed and it spans from 700 BC to 480 BC. This aryballos was decorated with the black-figure technique. In the black-figure technique, the silhouetes are rendered in black on the pale surface of the clay, and details are engraved. This aryballos is currently in the Museum of Delos.

We note that in many domain ontologies it is impractical to represent all the information in purely logical terms. In our example, it is much easier to store the information that "An aryballos was a small... spread their bodies with" directly as a string, i.e., as a *canned* sentence in NLG terminology, as opposed to defining classes, properties, and individuals for spreading actions, bodies, athletes, etc. and generating the sentence from a logical meaning representation. Similar comments apply to the other canned sentences in the example above. A disadvantage of using canned sentences is that they may have to be entered in multiple versions, if several languages or user types need to be supported. Some of the advantages of NLG are still available, however, when the canned sentences are property values of the domain ontology as above. For example, the canned sentences can still be placed at appropriate positions in the overall text, as with sentences generated from purely logical facts, and some of their referring expressions (e.g., pronouns) can be automatically produced, as we discuss below.[16]

Limitations of content selection

Apart from the restrictions of Tables 1–3, a further limitation of NaturalOWL's content selection is that it only retrieves information that is *explicit* in the domain ontology. It cannot *deduce* additional information about the target from other statements in the domain ontology; see Mellish *et al.* [122] for mechanisms to deduce facts in content selection.

[15] Consult `http://www.w3.org/TR/owl-time/` for more principled representations of time in OWL.

[16] In NaturalOWL, canned texts can also be entered as natural language names (see Section 2.2.1) of pseudo-individuals; the pseudo-individuals can then be used instead of the canned texts in the domain ontology, provided that the corresponding string-valued properties are converted to object properties. This way, canned texts can be entered (in the domain-dependent generation resources) in multiple versions (for different languages and user types) as different versions of the natural language names of the pseudo-individuals, without cluttering up the domain ontology with multiple strings.

We also note that OWL allows one to define the broadest possible domain and range of a particular property, using statements like the following.

```
ObjectPropertyDomain(:madeFromGrape :Wine)
ObjectPropertyRange(:madeFromGrape :Grape)
```

In practice, more specific range restrictions are then imposed for particular subclasses of the domain. For example, the following statements specify that when `madeFromGrape` is used with individuals from the subclass `GreekWine` of `Wine`, the range (possible values) of `madeFromGrape` should be restricted to individuals from the subclass `GreekGrape` of `Grape`.

```
SubClassOf(:GreekWine :Wine)
SubClassOf(:GreekGrape :Grape)
SubClassOf(:GreekWine
  AllValuesFrom(:madeFromGrape :GreekGrape))
```

NaturalOWL considers `AllValuesFrom` and similar restrictions, but it ignores `ObjectPropertyDomain` and `ObjectPropertyRange` statements.[17] Considering the latter statements as well would be easy, but they typically provide too general and, hence, uninteresting information from the perspective of end-users.

More generally, NaturalOWL does not consider OWL statements that express axioms about properties, meaning statements declaring that a property is symmetric, asymmetric, reflexive, irreflexive, transitive, functional, that its inverse is functional, that a property is the inverse of, or disjoint with another property, that it is subsumed by a chain of other properties, or that it is a subproperty (more specific) of another property. Statements of this kind are mostly useful in consistency checks, in deduction, or when generating texts describing the properties themselves (e.g., what being a grandparent of somebody means), cases that are not directly relevant to the work of this article.[18]

Converting candidate facts to message triples

Tables 4 and 5 list more exhaustively all the forms of candidate facts that are considered when describing a target individual or class, i.e., all the candidate facts that are admitted by Tables 1–3. Tables 4 and 5 also show how the candidate facts can be rewritten as triples of the form $\langle S, P, O\rangle$, where S is always the target or a second-level target; O is an individual, a datatype value, a class, or a set of individuals,

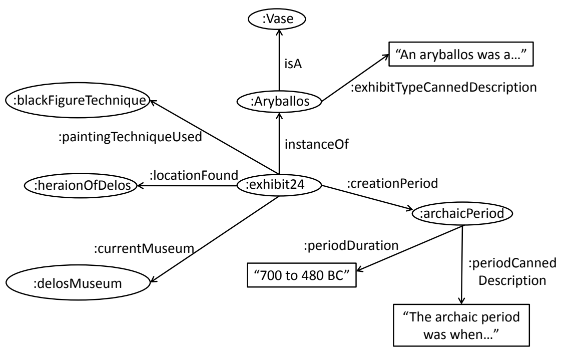


Figure 2. Graph view of message triples corresponding to candidate facts. The target is the individual `exhibit24`.

datatype values, or classes that S is mapped to; P specifies the kind of mapping. We call S the *semantic subject* or *owner* of the triple, and O the *semantic object* or *filler*; the triple can also be viewed as a field named P, owned by S, and filled by O. For example, the candidate facts about `exhibit24` on page 8, including the additional facts about the second-level targets, are converted to the following triples.

```
<:exhibit24, instanceOf, :Aryballos>
<:exhibit24, :locationFound, :heraionOfDelos>
<:exhibit24, :creationPeriod, :archaicPeriod>
<:exhibit24, :paintingTechniqueUsed,
 :blackFigureTechnique>
<:exhibit24, :currentMuseum, :delosMuseum>
<:Aryballos, isA, :Vase>
<:Aryballos, :exhibitTypeCannedDescription,
 "An aryballos was a small spherical vase with a
 narrow neck, in which the athletes kept the oil
 they spread their bodies with"^^xsd:string>
<:archaicPeriod, :periodDuration,
 "700 BC to 480 BC"^^xsd:string>
<:archaicPeriod, :periodCannedDescription,
 "The archaic period was when the Greek ancient
 city-states developed"^^xsd:string>
<:blackFigureTechnique,
 :techniqueCannedDescription,
 "In the black-figure technique, the silhouetes
 are rendered in black on the pale surface of the
 clay, and details are engraved"^^xsd:string>
```

More precisely, P can be: (i) a property of the domain ontology; (ii) one of the keywords `instanceOf`, `oneOf`, `differentIndividuals`, `sameIndividuals`, `isA`; or (iii) an expression of the form *modifier*(ρ), where *modifier* may be `not`, `maxCardinality` etc. (see Tables 4 and 5) and ρ is a property of the domain ontology. We hereafter call *properties* all three types of P, though types (ii) and (iii) are strictly not properties in OWL's terminology. When we need to distinguish between the three types, we use the terms *property of the domain ontology*, *domain-independent property*, and *modified property*, respectively.

For readers familiar with RDF, we note that the

[17] The only exception is when it expresses cardinality restrictions; see Table 7 of Section 2.2.1 below.

[18] Subproperties without sentence plans, discussed below, could inherit sentence plans from their super-properties, but in that case we automatically extract sentence plans from the domain ontology instead.

candidate facts (for an individual target) as OWL statements	candidate facts as message triples
ClassAssertion(*NamedClass target*)	<*target*, instanceOf, *NamedClass*>
ClassAssertion(ObjectComplementOf(*NamedClass*) *target*)	<*target*, not(instanceOf), *NamedClass*>
ClassAssertion(ObjectOneOf(*indiv1 indiv2 ...*) *target*)	<*target*, oneOf, or(*indiv1*, *indiv2*, ...)>
ClassAssertion(ObjectHasValue(*objProp indiv*) *target*)	<*target*, *objProp*, *indiv*>
ClassAssertion(ObjectHasValue(*dataProp dataValue*) *target*)	<*target*, *dataProp*, *dataValue*>
ClassAssertion(ObjectHasSelf(*objProp*) *target*)	<*target*, *objProp*, *target*>
ClassAssertion(ObjectMaxCardinality(*number prop* [*NamedClass*]) *target*)	<*target*, maxCardinality(*prop*), *number*[:*NamedClass*]>
ClassAssertion(ObjectMinCardinality(*number prop* [*NamedClass*]) *target*)	<*target*, minCardinality(*prop*), *number*[:*NamedClass*]>
ClassAssertion(ObjectExactCardinality(*number prop* [*NamedClass*]) *target*)	<*target*, exactCardinality(*prop*), *number*[:*NamedClass*]>
ClassAssertion(ObjectSomeValuesFrom(*objProp NamedClass*) *target*)	<*target*, someValuesFrom(*objProp*), *NamedClass*>
ClassAssertion(ObjectAllValuesFrom(*objProp NamedClass*) *target*)	<*target*, allValuesFrom(*objProp*), *NamedClass*>
ClassAssertion(ObjectIntersectionOf(*C1 C2 ...*) *target*)	*convert*(ClassAssertion(*C1 target*)) *convert*(ClassAssertion(*C2 target*)) ...
ClassAssertion(ObjectUnionOf(*C1 C2 ...*) *target*)	or(*convert*(ClassAssertion(*C1 target*)), *convert*(ClassAssertion(*C2 target*)), ...)
ObjectPropertyAssertion(*objProp target indiv*)	<*target*, *objProp*, *indiv*>
DataPropertyAssertion(*dataProp target dataValue*)	<*target*, *dataProp*, *dataValue*>
DifferentIndividuals(*target indiv*)	<*target*, differentIndividuals, *indiv*>
DifferentIndividuals(*indiv target*)	<*target*, differentIndividuals, *indiv*>
SameIndividual(*target indiv*)	<*target*, sameIndividual, *indiv*>
SameIndividual(*indiv target*)	<*target*, sameIndividual, *indiv*>

Table 4
Candidate facts when generating a text for an **individual** (see also Tables 1 and 2), and how they can be converted to message triples. Square brackets indicate optional arguments, and *convert* (ξ) denotes a recursive application of the conversion to ξ.

triples of Tables 4 and 5 are not exactly RDF triples. Most notably, expressions of the form *modifier*(ρ) cannot be used as P in RDF triples. Overall, each triple produced by Tables 4 and 5 is intended to be easily expressible as a single sentence, which is not always the case with RDF triples representing OWL statements. To avoid confusion, we use the term *message triples* to refer to the triples that Tables 4 and 5 generate, as opposed to RDF triples. [19] As with RDF triples, message triples can be viewed as a graph, which is very similar to ILEX's *content potential* [132]. Figure 2 shows a graph for the message triples of `exhibit24`; to save space, we exclude the triple that links `blackFigureTechnique` to a canned sentence. The second-level targets are the nodes of classes and individuals at distance one from `exhibit24`. The graph for the RDF triples would be more complex, and second-level targets would not always be at distance one from the current target.

Notice that Table 5 converts `EquivalentClasses` and `SubClassOf` statements to identical triples, where P is `isA`, since NaturalOWL produces the same texts from both kinds of statements, as already discussed. As a further example, OWL statements like the following two are mapped to identical message triples, apart

[19] Message triples correspond to Reiter and Dale's [145] *messages*. Message triples can be converted to RDF triples.

candidate fact (for a class target) as OWL statements	candidate facts as message triples
`EquivalentClasses(`*Target Class*`)`	*convert* (SubClassOf(*Target Class*))
`EquivalentClasses(`*Class Target*`)`	*convert* (SubClassOf(*Target Class*))
`SubClassOf(`*Target NamedClass*`)`	`<`*Target*`, isA, `*NamedClass*`>`
`SubClassOf(`*Target* `ObjectComplementOf(`*NamedClass*`))`	`<`*Target*`, not(isA), `*NamedClass*`>`
`SubClassOf(`*Target* `ObjectOneOf(`*indiv1 indiv2* `...))`	`<`*Target*`, oneOf, or(`*indiv1*`, `*indiv2*`, ...)>`
`SubClassOf(`*Target* `ObjectHasValue(`*objProp indiv*`))`	`<`*Target*`, `*objProp*`, `*indiv*`>`
`SubClassOf(`*Target* `ObjectHasValue(`*dataProp dataValue*`))`	`<`*Target*`, `*dataProp*`, `*dataValue*`>`
`SubClassOf(`*Target* `ObjectHasSelf(`*objProp*`))`	`<`*Target*`, `*objProp*`, `*Target*`>`
`SubClassOf(`*Target* `ObjectMaxCardinality(`*number prop* `[`*NamedClass*`]))`	`<`*Target*`, maxCardinality(`*prop*`), `*number*`[:`*NamedClass*`]>`
`SubClassOf(`*Target* `ObjectMinCardinality(`*number prop* `[`*NamedClass*`]))`	`<`*Target*`, minCardinality(`*prop*`), `*number*`[:`*NamedClass*`]>`
`SubClassOf(`*Target* `ObjectExactCardinality(`*number prop* `[`*NamedClass*`]))`	`<`*Target*`, exactCardinality(`*objProp*`), `*number*`[:`*NamedClass*`]>`
`SubClassOf(`*Target* `ObjectSomeValuesFrom(`*objProp NamedClass*`))`	`<`*Target*`, someValuesFrom(`*objProp*`), `*NamedClass*`>`
`SubClassOf(`*Target* `ObjectAllValuesFrom(`*objProp NamedClass*`))`	`<`*Target*`, allValuesFrom(`*objProp*`), `*NamedClass*`>`
`SubClassOf(`*Target* `ObjectIntersectionOf(`*C1 C2* `...))`	*convert* (SubClassOf(*C1 Target*)) *convert* (SubClassOf(*C2 Target*)) ...
`SubClassOf(`*Target* `ObjectUnionOf(`*C1 C2* `...))`	or(*convert* (SubClassOf(*C1 Target*)), *convert* (SubClassOf(*C2 Target*)), ...)
`DisjointClasses(`*Target NamedClass*`)`	`<`*Target*`, not(isA), `*NamedClass*`>`
`DisjointClasses(`*NamedClass Target*`)`	`<`*Target*`, not(isA), `*NamedClass*`>`

Table 5
Candidate facts when generating a text for a **class** (see also Tables 2 and 3), and how they can be converted to message triples. Square brackets indicate optional arguments, and *convert* (ξ) denotes a recursive application of the conversion to ξ.

from the identifiers of the individual and the class.

```
ClassAssertion(
  ObjectMaxCardinality(1 :madeFromGrape)
  :product145)
SubClassOf(:StEmilion
  ObjectMaxCardinality(1 :madeFromGrape))
```

The resulting message triples, shown below, are intended to reflect the similarity of the corresponding sentences that NaturalOWL would generate.

```
<:product145, maxCardinality(:madeFromGrape), 1>
```

Product 145 is made from at most one grape.

```
<:StEmilion, maxCardinality(:madeFromGrape), 1>
```

St. Emilion is made from at most one grape.

Tables 4 and 5 also discard `ObjectIntersectionOf` operators, producing multiple message triples instead. For example, the `SubClassOf` for `StEmilion` on page 8 would be converted to the following triples.

```
<:StEmilion, isA, :Bordeaux>
<:StEmilion, :locatedIn, :stEmilionRegion>
<:StEmilion, :hasColor, :red>
<:StEmilion, :hasFlavor, :strong>
<:StEmilion, :madeFromGrape,
 :cabernetSauvignonGrape>
<:StEmilion, maxCardinality(:madeFromGrape), 1>
```

The triples correspond to the sentences below, where subsequent references to `StEmilion` have been replaced by pronouns. The sentences could be aggregated into longer ones as discussed below.

St. Emilion is a kind of Bordeaux. It is from the St. Emilion region. It has red color. It has strong flavor. It is made from Cabernet Sauvignon grape. It is made from at most one grape variety.

Tables 4 and 5 also replace `ObjectUnionOf` operators by disjunctions of message triples. The assertion:

```
ClassAssertion(
 UnionOf(
  ObjectHasValue(:hasFlavor :strong)
  ObjectHasValue(:hasFlavor :medium))
 :houseWine)
```

becomes:

```
or(<:houseWine, :hasFlavor, :strong>,
   <:houseWine, :hasFlavor, :medium>)
```

which corresponds to the following setence.

The house wine has strong flavor or it has medium flavor.

The need for interest scores

Conveying to an end-user all the message triples of all the candidate facts is not always appropriate. Let us assume, for example, that the maximum fact distance is 2 and that a description of `exhibit24` of Figure 2 has been requested by a museum visitor. It may be the case that the visitor has already encountered other exhibits of the archaic period, and that the duration of that period was mentioned in the previous descriptions. Repeating the archaic period's duration may, thus, be undesirable. We may also want to exclude candidate facts that are uninteresting to particular types of users. For example, there may be message triples providing bibliographic references; archaeologists may find them interesting, but children would probably not. Therefore, mechanisms are needed to determine which candidate message triples should be conveyed to each user.

Following ILEX and M-PIRO, each message triple is assigned an *interest score*, possibly different per user type. The score is a non-negative integer indicating how interesting a user of the corresponding type will presumably find the triple's information, if the information has not already been conveyed to the user.[20] Furthermore, personal user models, to be discussed, keep track of the message triples the system has expressed to each particular user.

In the museum projects NaturalOWL was originally developed for, the interest scores ranged from 0 (completely uninteresting) to 3 (very interesting); a different maximum score can also be used. The scores were set by consulting museum curators, who were shown a list of all the properties of the domain ontology (e.g., `locationFound`) that applied to targets (exhibits) or second-level targets (e.g., historical periods), along with sample sentences expressing message triples involving each property. The curators were asked to specify how interesting the information expressed by each property would generally be per user type.[21] Each message triple was then assigned the interest score of its property. The user types (e.g., non-expert adult, expert adult, child) were also suggested by the curators. The curators later examined the generated texts and suggested further refinements of the interest score assignments, which would occasionally apply to $\langle S, P, O\rangle$ message triples with a particular individual or class as their S, rather than to all the messages triples that involved a particular property P. Hence, we had to allow refinements of this kind.

We note that when large numbers of human-authored texts describing individuals and classes of the domain ontology are available, along with the corresponding logical facts expressed by each text, statistical and machine learning methods can be employed to learn to automatically select or assign interest scores to logical facts (or message triples) [52,10,91,93,5]. Another possibility would be to compute the interest scores with centrality algorithms. Algorithms of this kind assign higher or lower importance to a graph's nodes by considering how well connected they are and how important their neighbors are. Demir *et al.* [45], for example, applied a version of PageRank [29] to a graph that had nodes standing for candidate logical facts, and edges corresponding to relations between facts (e.g., showing that two facts share individuals, or that they should or should not be selected together).

Let us now explain how message triples $\langle S, P, O\rangle$ are (manually) assigned interest scores in NaturalOWL. Three types of annotations of the domain ontology can be used; we discuss them in turn.

Assigning interest scores by specifying P only

An annotation of this type applies to all the message triples involving a particular P, regardless of S and O. For example, let us assume that any triple whose P is `currentMuseum`, like the one below, should have an interest score of 1 when generating texts for children; by contrast, the interest score should be 3 when generating texts for experts.

[20] ILEX distinguishes between *interest* and *importance*, the latter being the educational value of each fact, which may be different from its interest [132]. We use only interest scores.

[21] Consult Reiter *et al.* [146] for a discussion of knowledge acquisition methods that can be used in NLG.

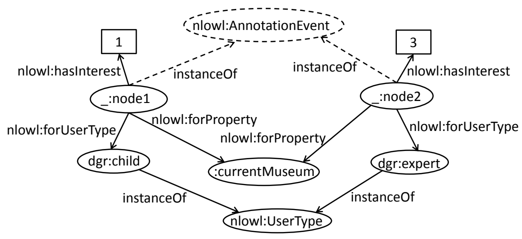


Figure 3. Graph view of annotations providing interest scores. Message triples involving the `currentMuseum` property are assigned an interest of 1 for children and 3 for experts.

```
<:exhibit24, :currentMuseum, :delosMuseum>
```

The following annotations can be used. The annotations themselves are also OWL statements, and Figure 3 shows them as a graph.

```
AnnotationAssertion(nlowl:forProperty
  _:node1 :currentMuseum)
ObjectPropertyAssertion(nlowl:forUserType
  _:node1 dgr:child)
ObjectPropertyAssertion(nlowl:hasInterest
  _:node1 "1"^^xsd:nonNegativeInteger)

AnnotationAssertion(nlowl:forProperty
  _:node2 :currentMuseum)
ObjectPropertyAssertion(nlowl:forUserType
  _:node2 dgr:expert)
ObjectPropertyAssertion(nlowl:hasInterest
  _:node2 "3"^^xsd:nonNegativeInteger)
```

Intuitively, the statements above introduce two annotation events. The first one (`node1`) concerns children and annotates the property `currentMuseum` with an interest score of 1. The second event (`node2`) concerns experts and annotates the same property with an interest score of 3. If the `forUserType` of an annotation event is left unspecified, the interest score applies to all the user types.

For simplicity, we use the default namespace (a colon without a prefix) with properties (e.g., `:currentMuseum`), classes etc. of the domain ontology. The `nlowl` namespace prefix is used with *domain-independent* properties, classes etc. that are defined in NaturalOWL's generation resources ontology (e.g., `nlowl:forProperty`); and the `dgr` prefix is used with *domain-dependent* instances (e.g., individuals) of NaturalOWL's generation resources ontology. The user types `child` and `expert` in the example above would be declared to be individuals of the class `UserType` (as shown below), which is defined in NaturalOWL's generation resources ontology.

```
ClassAssertion(dgr:child nlowl:UserType)
ClassAssertion(dgr:expert nlowl:UserType)
```

Since we may wish to use different user types with different domain ontologies, `child` and `expert` are considered domain-dependent, which is why they have the `dgr` prefix.[22] Individuals whose prefixes are underscores (e.g., `_:node1`, `_:node2`) are considered anonymous in OWL; strictly speaking, they should be shown as nodes without labels in the graphs.

Statements of the form `AnnotationAssertion(`*`anProp element value`*`)` are used in OWL to map any *`element`* (e.g., class, property, individual) of an ontology to any *`value`* (e.g., individual, datatype value) via an annotation property *`anProp`* in order to express meta-information that is not considered part of the ontology's conceptualization. For example, annotation properties are commonly used to associate elements with comments. We use annotation properties when linking elements of the domain ontology to elements of the generation resources ontology, or vice versa, but we use ordinary properties when linking (internally) elements of the domain ontology or when linking (internally) elements of the generation resources ontology. The `AnnotationAssertion` statements that we use can be thought of as annotations of the domain ontology that associate it with linguistic and user modeling resources, which are not part of the domain ontology's conceptual model.[23] Like object and datatype properties, annotation properties can express directly only binary relations. Hence, we follow common practice and use anonymous individuals (e.g., `_:node1`) that stand for reified events to express relations of more than two arguments.

Recall that the P of a message triple $\langle S, P, O\rangle$ is not always a property of the domain ontology (see Tables 4 and 5). It may be a modified property, like `maxCardinality(currentMuseum)`, in which case we say that `maxCardinality` is a *modifier* of `currentMuseum`. The statements below specify that message triples whose P is `exactCardinality(currentMuseum)` have an interest score of zero when interacting with children; users are never told zero interest facts.

```
AnnotationAssertion(nlowl:forProperty
  _:node3 :currentMuseum)
ObjectPropertyAssertion(nlowl:forModifier
  _:node3 nlowl:exactCardinality)
ObjectPropertyAssertion(nlowl:forUserType
  _:node3 dgr:child)
```

[22] In practice, `dgr` would be an abbreviation of a namespace other than that of the domain ontology and NaturalOWL's generation resources ontology, to avoid name clashes. A complete definition (in OWL) of NaturalOWL's generation resources ontology is included in the system's software.

[23] Annotation properties have the additional advantage that they can map classes or properties, not just individuals, to values, and the values themselves can also be properties; this turns out to be useful in some of NaturalOWL's annotations.

```
ObjectPropertyAssertion(nlowl:hasInterest
  _:node3 "0"^^xsd:nonNegativeInteger)
```

By omitting the `forProperty`, we can also specify, for example, that triples with an `exactCardinality` should never be expressed to children, regardless of the property the modifier applies to. When there are conflicting interest scores (e.g., for `exactCardinality(currentMuseum)` and `exactCardinality` in general), the most specific one prevails.[24]

In practice, the user modeling annotations are entered by using NaturalOWL's Protégé plug-in, instead of writing OWL statements. The annotation events are automatically generated by the plug-in, and they are not shown to the domain author.

Assigning interest scores by specifying both S and P

Most message triples $\langle S, P, O\rangle$ are assigned interest scores by using annotations that specify a particular P (and/or possibly a modifier), an interest score, and possibly a particular user type, as discussed above. In some cases, however, we may also want the annotations to specify a particular S. For example, we may have assigned, say for all user types, an interest score of 3 to message triples whose P is `instanceOf`, i.e., triples expressing the class of S. In the following triple, however, we would probably want to set the interest score to zero, because the natural language name of S makes it obvious that S is a museum, i.e., we would want to avoid generating the corresponding sentence.

```
<:delosMuseum, instanceOf, :Museum>
```

The Museum of Delos is a museum.

This is achieved below by specifying that message triples whose P is `instanceOf` and S is `delosMuseum` have zero interest for all user types; recall that S is also called the *owner* of the triple.

```
AnnotationAssertion(nlowl:forProperty
  _:node4 nlowl:instanceOf)
AnnotationAssertion(nlowl:forOwner
  _:node4 :delosMuseum)
ObjectPropertyAssertion(nlowl:hasInterest
  _:node4 "0"^^xsd:nonNegativeInteger)
```

Similarly, we may wish to specify that the materials of a museum's exhibits are generally of medium interest (score 2), that the materials of statues are of lower interest (score 1), perhaps because they are all made from stone, but that the material of a particular statue, `exhibit10`, is very important (score 3), perhaps because it is gold. We could use the following statements. The first annotation event (`node5`) is overriden by the second one (`node6`), when S is the class `Statue` or any individual that belongs in that class. The third annotation event (`node7`) overrides both of the previous ones, when S is `exhibit10`.

```
AnnotationAssertion(nlowl:forProperty
  _:node5 :hasMaterials)
ObjectPropertyAssertion(nlowl:hasInterest
  _:node5 "2"^^xsd:nonNegativeInteger)

AnnotationAssertion(nlowl:forProperty
  _:node6 :hasMaterials)
AnnotationAssertion(nlowl:forOwner
  _:node6 :Statue)
ObjectPropertyAssertion(nlowl:hasInterest
  _:node6 "1"^^xsd:nonNegativeInteger)

AnnotationAssertion(nlowl:forProperty
  _:node7 :hasMaterials)
AnnotationAssertion(nlowl:forOwner
  _:node7 :exhibit10)
ObjectPropertyAssertion(nlowl:hasInterest
  _:node6 "3"^^xsd:nonNegativeInteger)
```

There should perhaps also be a mechanism to assign interest scores by specifying P and O, for example to assign a high interest to triples indicating that some (any) exhibit is made of gold. NaturalOWL, however, currently provides no such mechanism.

Assigning interest scores at the global level

Message triples may also be assigned interest scores by using *global* annotations specifying neither P nor S; the annotations then apply to all the message triples. For example, the following statement assigns an interest score of 1 to all triples, unless otherwise specified; this is also the default.

```
ObjectPropertyAssertion(nlowl:hasInterest
  _:node8 "1"^^xsd:nonNegativeInteger)
```

A `forUserType` can be used to restrict the scope of a global annotation to a particular user type.

Repetitions and personal user models

Following ILEX and M-PIRO, NaturalOWL's user modeling annotations also specify how many times each message triple has to be repeated in the generated texts, before it can be assumed that users of different types have assimilated it. Once a triple has been assimilated, it is never repeated in texts for the same user, unless the user's personal model is reset. For example, the annotations may indicate that for children, the duration of a historical period is assimilated only after it has been mentioned twice; hence, the system may repeat the duration of the

[24] In the case of disjunctions of message triples, the entire disjunction is assigned the average of the interest scores of its triples, and the disjunction is expressed as a single sentence.

archaic period in two texts about exhibits of that period. The necessary repetitions are specified much as when specifying interest scores. For example, the following statements set both the interest score (to 1) and the required repetitions (to 2) for children at the global level, i.e., for all properties, using a single annotation event. If not specified otherwise, message triples are expressed only once.

```
ObjectPropertyAssertion(nlowl:forUserType
  _:node9 dgr:child)
ObjectPropertyAssertion(nlowl:hasInterest
  _:node9 "1"^^xsd:nonNegativeInteger)
ObjectPropertyAssertion(nlowl:maxRepetitions
  _:node9 "2"^^xsd:nonNegativeInteger)
```

A `maxRepetitions` score of 0 signals that the triple should never be considered assimilated.

NaturalOWL maintains a personal model for each end-user. The model shows which message triples were conveyed to that user in previous texts, and how many times. Each end-user is assigned a unique identifier during a login phase, and the identifiers are used to retrieve their personal models. During the login phase, the users also select their types; in one of M-PIRO's demonstrators, for example, the user types were depicted using cartoon characters, and the users selected the character they felt closer to.

Selecting the message triples to convey

When asked to describe a target, NaturalOWL first retrieves from the domain ontology the candidate facts up to the maximum fact distance. It then converts the candidate facts to message triples, and it employs the user modeling annotations and the personal user models to rank the triples by decreasing interest score, discarding triples that have already been assimilated. If a message triple at distance one has been assimilated, then all the distance two triples that are connected to the assimilated triple are also discarded; for example, if the `creationPeriod` triple (edge) of Figure 2 has been assimilated, then the triples about the archaic period (the edges leaving from `archaicPeriod`) are also discarded. The system then selects up to `maxMessagesPerPage` triples from the most interesting remaining ones. The value of `maxMessagesPerPage` can be set to smaller or larger values for types of users that prefer shorter or longer texts. The following statement indicates that each text should express at most 10 message triples when interacting with children. The default is to convey all the non-assimilated candidate message triples.

```
DataPropertyAssertion(nlowl:maxMessagesPerPage
  dgr:child "10"^^xsd:nonNegativeInteger)
```

In some applications, it may be possible to allow users to specify their personal `maxMessagesPerPage` value, for example via a slide-bar, but a reasonable initial value per user type is desirable in most cases.

2.1.2. *Text planning*

For each target class or individual, the mechanisms of the previous section produce the message triples to be expressed, with each triple intended to be easily expressible as a single sentence. NaturalOWL's text planning stage then orders the message triples, in effect ordering the sentences of the resulting text. We discuss the functionality and resources of this stage, after first providing some background on text planning, for the benefit of readers unfamiliar with NLG and also to explain why some particular methods were chosen in NaturalOWL.

Related work on global coherence

Among other considerations, text planners often aim to maximize the global or local coherence of the resulting texts, or both. When considering *global coherence*, they attempt to build a structure, usually a tree, that shows how the clauses, sentences, or larger segments (or their meaning representations, in our case the message triples) are related to each other, often in terms of rhetorical relations, as in Rhetorical Structure Theory (RST) [113]. For example, a sentence or other segment may justify, elaborate, or contradict another one, it may provide background information etc. The allowed or preferred orderings of the text's sentences (or segments) often follow, at least partially, from its global coherence structure.

In the texts, however, that NaturalOWL is intended to generate, i.e., factual descriptions of individuals or classes, the global coherence structures tend to be rather uninteresting, because most of the sentences simply provide additional information about the target or the second-level targets; the sentences are connected mostly via what are usually called 'elaboration', 'list', or 'background' relations. Hence, global coherence structures are not particularly informative in our case, which is why they are not used in NaturalOWL. Consult, for example, Hovy [80], Moore and Paris [128], and Paris *et al.* [134] for descriptions of systems that use global coherence structures, rhetorical relations, and plan libraries. Mellish *et al.* [120] employ genetic algorithms to search for a good global coherence structure, given a set of facts to convey (per text) and a set of possible rhetorical relations that connect them. Duboue and McKeown [51] also use genetic algorithms, but to learn a single text planner (for all the texts to be generated), given a training set of semantic inputs and the corresponding human-authored texts.

We note that Liang *et al.* [109] discuss using RST in an ontology verbalizer; they seem to agree, however, that very few types of RST relations are relevant when generating texts from OWL ontologies. Also, Power [138] studied when two sentences expressing `ClassAssertion`, `SubClassOf`, or `ObjectPropertyAssertion` axioms are judged by humans to be rhetorically related; and when they are, what kinds of rhetorical relations they express, and how the relations are expressed (e.g., by using discourse connectives or sentence aggregation). Power found 11 patterns of rhetorically related pairs of sentences (and axioms). Most of these patterns, however, do not apply to the sentences that NaturalOWL produces; for example, many patterns express contrast or comparisons between *two* target classes, whereas in NaturalOWL there is always only one target. Only two of Power's patterns seem to apply to NaturalOWL's sentences: *additive elaboration* (e.g., "Dogs are canines; dogs are domestic mammals") and *forward reasoning* (e.g., "Dogs are canines; canines are vertebrates").[25] Additive elaboration is handled reasonably well by NaturalOWL's sentence aggregation rules, discussed below. Forward reasoning sentences are placed next to each other by NaturalOWL's text planner; no discourse connective is used, but Power's data suggest that humans also rarely use discourse connectives in this case.

Local coherence

When considering *local coherence*, text planners usually aim to maximize measures that examine, roughly speaking, whether or not adjacent sentences (or other segments) continue to focus on (i.e., talk primarily about) the same entities or, if the focus changes, how smooth the transition is. Many local coherence measures are based on Centering Theory (CT) [66,170,135], which we briefly discuss first.[26]

In CT, each utterance u_n (in our case, sentence) is associated with a set of *forward-looking centers* of attention, denoted $C_f(u_n)$. The members of $C_f(u_n)$ are discourse entities realized as noun phrases in u_n; in our case, the discourse entities are individuals or classes of the domain ontology. The members of $C_f(u_n)$ can be ordered by their salience, which reflects the salience of the noun phrases that realize them in u_n; for example, subject noun phrases are considered more salient than object noun phrases. The most salient member of $C_f(u_n)$ is u_n's *preferred center*, denoted $c_p(u_n)$. In a sentence that NaturalOWL generates for a message triple $\langle S, P, O\rangle$, typically $C_f(u_n) = \{S, O\}$ and $c_p(u_n) = S$, since S is typically realized as the subject of u_n.[27]

Let u_{n-1} be the utterance (sentence) immediately before u_n. The most salient member of $C_f(u_{n-1})$ that is also realized in u_n is the *backward-looking center* of u_n, denoted $c_b(u_n)$. If no member of $C_f(u_{n-1})$ is realized (repeated) in u_n, then $c_b(u_n)$ is undefined and we have a type of transition from u_{n-1} to u_n that Karamanis *et al.* [87] and others call NOCB, a transition type to be avoided. The preferred type of sentence-to-sentence transition, CONTINUE, occurs when $c_p(u_n) = c_b(u_n) = c_b(u_{n-1})$. If u_{n-1} is the first utterance of the text, $c_b(u_{n-1})$ is undefined. However, if $c_b(u_{n-1})$ is undefined but $c_b(u_n)$ exists, $c_b(u_n) = c_b(u_{n-1})$ is taken to hold; a NOCB occurs (from u_{n-1} to u_n) when $c_b(u_n)$ is undefined.

When NaturalOWL's maximum fact distance is one, all the transitions in the generated texts are CONTINUE, because $c_p(u_n) = c_b(u_n) = c_b(u_{n-1}) = t$, for every u_n with $n > 1$, where t is the target, i.e., the individual or class described by the text. To illustrate this, we repeat the short description of the aryballos of page 8, without aggregating sentences.

> *This* (exhibit) is an aryballos. <u>*It*</u> was found at the Heraion of Delos. <u>*It*</u> was created during the archaic period. <u>*It*</u> was decorated with the black-figure technique. <u>*It*</u> is currently in the Museum of Delos.

The subjects of all the sentences, shown in italics, realize t, and $c_p(u_n) = t$ in every utterance. We show underlined the noun phrase of each sentence that realizes its $c_b(u_n)$. In our example, for every $n > 1$, $c_b(u_n)$ is the discourse entity realized by the

[25] *Widening elaboration* and *narrowing elaboration* pairs [138] may also, in principle, be generated by NaturalOWL, but they seem to be rare in practice, because they occur when the domain ontology contains redundant statements, for example explicitly stating that dogs are both canines and vertebrates, and also that canines are vertebrates.

[26] Consult Karamanis *et al.* [87] for a summary of measures based on CT. Readers unfamiliar with CT should still be able to follow most of the discussion here. For those who wish to grasp the full details, the article of Karamanis *et al.* also serves as a concise introduction to relevant CT concepts.

[27] We say typically, because there is actually nothing to prevent a domain author from providing an unusual sentence plan that, for example, causes NaturalOWL to realize S as the object of the sentence; sentence plans are discussed below. Also, when canned texts are used, $c_p(u_n)$ may occasionally not be realized as the subject; we provide an example below.

subject of u_{n-1}, i.e., t; hence, $c_p(u_n) = c_b(u_n) = t$. Furthermore, for every $n > 2$, $c_b(u_n) = c_b(u_{n-1}) = t$; and for $n = 2$, $c_b(u_{n-1})$ is undefined and $c_b(u_n) = t$, hence $c_b(u_n) = c_b(u_{n-1})$ is taken to hold.

If the maximum fact distance is two, however, the transitions are not always CONTINUE.[28] In our example, the description of the aryballos becomes as follows. We now repeat the long description of page 9, again without sentence aggregation. The most salient noun phrase of each sentence, which realizes $c_p(u_n)$, is shown in italics. In the (canned) ninth sentence, the most salient noun phrase is the one inside the sentence-initial prepositional phrase; in all the other sentences, it is the subject. Again, the underlined noun phrases realize $c_b(u_n)$.

(1) *This* (exhibit) is an aryballos. (2) *An aryballos* is a kind of vase. (3) *An aryballos* was a small spherical vase with a narrow neck, in which the athletes kept the oil they spread their bodies with. • (4) *This particular aryballos* was found at the Heraion of Delos. (5) *It* was created during the archaic period. (6) *The archaic period* was when the Greek ancient city-states developed. (7) *It* spans from 700 BC to 480 BC. • (8) *This aryballos* was decorated with the black-figure technique. (9) In *the black-figure technique*, the silhouetes are rendered in black on the pale surface of the clay, and details are engraved. • (10) *This aryballos* is currently in the Museum of Delos.

In the second sentence above, $c_p(u_2)$ is the class of aryballoi (plural of aryballos), not the particular target exhibit t the text is generated for; $c_b(u_2)$ is also the aryballos class; $c_b(u_1)$ is undefined, but $c_b(u_2) = c_b(u_1)$ is taken to hold. Hence, the transition from u_1 to u_2 is a CONTINUE. In sentence 3, $c_p(u_3)$ is again the class of aryballoi, and $c_p(u_3) = c_b(u_3) = c_b(u_2)$, i.e., the transition is again a CONTINUE. In sentence 4, however, where $c_p(u_4)$ is the particular target exhibit t, no forward-looking center of the previous sentence is mentioned and, hence, $c_b(u_4)$ is undefined and the transition from sentence 3 to sentence 4 is a NOCB; we mark NOCB transitions with bullets.[29] In sentence 5, $c_p(u_5) = t = c_b(u_5)$ and $c_b(u_4)$ is undefined; hence, the transition from sentence 4 to 5 is a CONTINUE. In sentence 6, $c_p(u_6)$ is the archaic period, which is also $c_b(u_6)$, but $c_b(u_5)$ is the exhibit t; hence, $c_p(u_6) = c_b(u_6) \neq c_b(u_5)$, and we have a kind of transition known as SMOOTH-SHIFT, which is less preferred than CONTINUE, but better than NOCB. A CONTINUE then occurs in sentence 7, a NOCB in sentence 8, followed by a SMOOTH-SHIFT in sentence 9, and another NOCB in sentence 10.

NaturalOWL's text planning algorithm, discussed below, always groups together sentences (more precisely, message triples) that describe a particular second-level target (e.g., sentences 2–3, 6–7, and 9) and it places each group immediately after the sentence that introduces the corresponding second-level target (immediately after sentences 1, 5, and 8, respectively). Thus the transition from a sentence that introduces a second-level target to the first sentence that describes the second-level target (e.g., from sentence 1 to 2, from 5 to 6, from 8 to 9) is a SMOOTH-SHIFT (or a CONTINUE in the special case from the initial sentence 1 to 2).[30] A NOCB occurs only at sentences that return to providing information about the primary target, after a group of sentences that provide information about a second-level target. All the other transitions are of type CONTINUE.

The resulting number of NOCBs equals the number of second-level targets described by at least a message triple, minus one if the last message triple (sentence) is not about the primary target. This is the smallest possible number of NOCBs for any reordering of the message triples, excluding special (and in practice rare) cases where a second-level message triple connects a second-level target back to the primary target, allowing a sentence about the primary target to follow without a NOCB.[31] Karamanis *et al.* [87] provide experimental evidence from multiple text genres showing that simply minimizing NOCB transitions leads to sentence orderings that people find better or as good as orderings that minimize several other, more elaborate CT-based local coherence measures. Hence, the oderings that NaturalOWL produces, which minimize NOCBs, are competitive to orderings produced by minimizing other

[28] See also the analysis of M-PIRO's generated texts by Karamanis *et al.* [87].

[29] It could be argued that the transition from (3) to (4) involves a kind of bridging relation [35] from the whole class to one of its individuals, which smoothens the transition; Kibble and Power [92] make a similar observation.

[30] The transition between sentences 1 and 2 should perhaps also be considered a SMOOTH-SHIFT. RETAIN transitions, where $c_p(u_n) \neq c_b(u_n) = c_b(u_{n-1})$, and ROUGH-SHIFT transitions, where $c_p(u_n) \neq c_b(u_n)$ and $c_b(u_n) \neq c_b(u_{n-1})$, do not occur in the sentences that NaturalOWL typically generates, unless NOCB is viewed as a ROUGH-SHIFT.

[31] We can sometimes reduce the NOCBs by one, by placing last, if possible, a sentence describing a second-level target. For example, moving sentence (10) immediately after (4) saves the NOCB transition from (9) to (10). NaturalOWL does not perform this check and, hence, the number of NOCBs may exceed the minimum by one, but we ignore this detail.

CT-based local coherence measures. Also, when using other local coherence measures, the optimum ordering may not be obvious; and searching for the ordering that maximizes the sum of the sentence-to-sentence local coherence scores is NP-complete [9,1].

A simple strategy to avoid NOCB transitions would be to end the generated text once all the message triples that describe a second-level target have been reported, and record in the user model that the other message triples that content selection had provided were not actually conveyed. In our example, this would generate sentences 1 to 3; then if the user requested more information about the exhibit, sentences 4 to 7 would be generated, and so on.

Topical order

When ordering sentences, we also need to consider the topical similarity of adjacent sentences. Compare, for example, the following two texts.

{$_{locationSection}$ The Stoa of Zeus Eleutherios is located in the western part of the Agora. It is located next to the Temple of Apollo Patroos.} {$_{buildSection}$ It was built around 430 BC. It was built in the Doric style. It was built out of porous stone and marble.} {$_{useSection}$ It was used during the Classical period, the Hellenistic period, and the Roman period. It was used as a religious place and a meeting point.} {$_{conditionSection}$ It was destroyed in the late Roman period. It was excavated in 1891 and 1931. Today it is in good condition.}

The Stoa of Zeus Eleutherios was built in the Doric style. It was excavated in 1891 and 1931. It was built out of porous stone and marble. It is located in the western part of the Agora. It was destroyed in the late Roman period. It was used as a religious place and a meeting point. It is located next to the Temple of Apollo Patroos. It was built around 430 BC. Today it is in good condition. It was used during the Classical period, the Hellenistic period, and the Roman period.

Both texts express the same message triples, shown below. Notice that we use a single `hasMaterial` message triple, whose filler is an `and(...)`, instead of two different triples (one for each material), and similarly for `usedDuringPeriod`. This kind of triple merging is in effect a form of aggregation, discussed below, but it takes place during content selection.

```
<:stoaZeusEleutherios, :isInArea, :westAgora>
<:stoaZeusEleutherios, :isNextTo,
 :templeApolloPatroos>
<:stoaZeusEleutherios, :hasApproxConstructYearBC,
 "430"^^xsd:nonNegativeInteger>
<:stoaZeusEleutherios, :hasStyle, :doricStyle>
<:stoaZeusEleutherios, :hasMaterial,
 and(:porousStone, :marble)>
<:stoaZeusEleutherios, :usedDuringPeriod,
 and(:classicalPeriod, :hellenisticPeriod,
 :romanPeriod)>
<:stoaZeusEleutherios, :usedAs,
 "a religious place and a meeting point"^^xsd:string>
<:stoaZeusEleutherios, :hasDestructionTime},
 "in the late Roman period"^^xsd:string>
<:stoaZeusEleutherios, :hasExcavationYearAD,
 "1891"^^xsd:nonNegativeInteger>
<:stoaZeusEleutherios, :hasExcavationYearAD,
 "1931"^^xsd:nonNegativeInteger>
<:stoaZeusEleutherios, :isInCurrentCondition,
 :goodCondition>
```

We listed the message triples with the order of the corresponding sentences in the first text, but at the beginning of text planning the triples are actually unordered. Even though both texts express the same message triples and contain the same sentences, the second text is more difficult to follow, if at all acceptable. The first one is better, because it groups together topically related sentences; the first two sentences are about the location of the stoa; the next three convey information about the building event; the following two are about the monument's use; and the last three are about its condition through history. [32] We mark the sentence groups in the first text by curly brackets, but the brackets would not be shown to end-users. In longer texts, sentence groups may optionally be shown as separate paragraphs or sections, which is why we call them *sections*.

To allow message triples (and sentences) to be grouped by topic, the domain author may define sections and assign each property (excluding modified ones) to exactly one section. In the example above, `locationSection`, `buildSection`, `useSection`, and `conditionSection` would be be declared to be individuals of the `Section` class of NaturalOWL's generation resources ontology. Properties are assigned to sections using annotation properties. For example, the following statements specify that `isInArea` and `isNextTo` are properties of `locationSection`. The statements would cause all message triples involving the two properties to be placed in `locationSection`.

```
AnnotationAssertion(nlowl:hasSection
 :isInArea dgr:locationSection)
AnnotationAssertion(nlowl:hasSection
 :isNextTo dgr:locationSection)
```

A partial order of properties inside their sections can also be specified, as shown below.

[32] Topically related sentences often share identical or semantically related words. Hence, grouping them also increases the *lexical cohesion* [72] of adjacent sentences.

```
AnnotationAssertion(nlowl:hasOrder
  :isInArea "1"^^xsd:nonNegativeInteger)
AnnotationAssertion(nlowl:hasOrder
  :isNextTo "2"^^xsd:nonNegativeInteger)
```

Message triples of properties with smaller `hasOrder` values are always placed before triples of properties of the same section with larger `hasOrder` values. [33] If two message triples involve properties of the same section with the same `hasOrder` values, their relative order within their section is selected randomly.

Similarly, a partial order can be imposed on sections, as shown below. Again, sections with smaller `hasOrder` values (and the sentences they contain) are placed before all other sections with larger `hasOrder` values. Sections with identical `hasOrder` values can either precede or follow each other.

```
AnnotationAssertion(nlowl:hasOrder
  :locationSection "1"^^xsd:nonNegativeInteger)
```

When using NaturalOWL's Protégé plug-in, the domain author orders sections and properties without seeing the `hasOrder` values, which are automatically generated. The sections and partial orders could be made sensitive to different user types, perhaps also different languages, though we have not encountered applications that required this.

The text planning algorithm

NaturalOWL's text planning algorithm is summarized in Figure 4. If the message triples to be ordered include triples that describe second-level targets, i.e., triples $\langle S, P, O\rangle$ whose owner S is a second-level target, then the triples of the primary and each second-level target are ordered separately. The ordered triples of each second-level target are then inserted into the ordered list of the primary target's triples immediately after the first triple that introduces the second-level target, i.e., immediately after the first triple whose O is the second-level target.

Further related work on text planning

NaturalOWL's ordering of properties and sections is in effect similar to using *text schemata* [116], roughly speaking domain-dependent patterns that specify the possible arrangements of different types of sentences (or other segments). The limitations of text schemata in more general settings have been discussed, for example, by Hovy [80] and Moore and Paris [128]. For the kinds of texts that NaturalOWL is intended to generate, however, the current ordering of properties and sections seems adequate.

```
procedure orderMessageTriples
inputs:
  t[0]: primary target
  t[1], ..., t[n]: second-level targets
  L[0]: unordered list of triples describing t[0]
  ...
  L[n]: unordered list of triples describing t[n]
  SMap: mapping from properties to sections
  SOrder: partial order of sections
  POrder: partial order of properties within sections
output:
  ordered list of message triples
steps:
  for i := 0 to n {
   orderMessageTriplesAux(L[i], SMap, SOrder, POrder)}
  for i := 1 to n {
    insertAfterFirst(<t[0], _, t[i]>, L[0], L[i])}
  return L[0]

procedure orderMessageTriplesAux
inputs:
  L: unordered list of triples about a single target
  SMap: mapping from properties to sections
  SOrder: partial order of sections
  POrder: partial order of properties within sections
variables:
 S[1], ..., S[k]: lists, each with triples of one section
output:
  ordered list of message triples about single target
steps:
  <S[1], ..., S[k]> := splitInSections(L, SMap)
  for i := 1 to k {
    S[i] := orderTriplesInSection(S[i], POrder)}
    <S[1], ..., S[k]> :=
      reorderSections(S[1], ..., S[k], SOrder)
  return concatenate(S[1], ..., S[k])
```

Figure 4. Algorithm used to order message triples.

Sentence ordering has also been studied extensively in extractive multi-document text summarization. For example, Barzilay *et al.* [9] automatically group the sentences that have been selected to be included in a summary into topical blocks; the selected sentences roughly correspond to our message triples and the topical blocks correspond to our sections. Subsequently, the blocks and the sentences inside them are automatically ordered. Barzilay *et al.* also provide experimental evidence showing that grouping sentences into topical blocks affects the comprehension and acceptability of the summaries.

Methods similar to those of Barzilay *et al.* [9] could be used in our case to automatically acquire the mapping from properties to sections, as well as the order of sections and properties, instead of providing them manually. This would require, however, a corpus of documents describing individuals and classes of the domain ontology; and the sentences

[33] The annotations that assign properties to sections and specify their partial order apply to all the triples of the particular properties, including triples involving modifiers.

of the corpus would have to be semantically tagged with the corresponding message triples, or at least their properties. Each document of the corpus could then be automatically segmented into topics, for example using methods discussed by Hearst [77]; or in some domains it may be possible to use Wikipedia or other documents with explicit sections [148]. Properties frequently expressed by sentences of the same topical segment would be placed in the same section. Inside a section, a property p_1 could be ordered before a property p_2, if more sentences expressing p_1 preceded sentences expressing p_2 in the corpus than the other way round; Barzilay *et al.* [9,148] call this majority ordering. The ordering of sections could be acquired similarly. Consult also Duboue and McKeown [50] and Dimitromanolaki and Androutsopoulos [49] for other methods to learn to order sentences or other segments in NLG; these methods, however, also require semantically tagged corpora.

An alternative is to generate a candidate text for each possible sentence ordering. The best candidate text can then be selected by a model trained to rank texts by their local coherence. Ranking models of this kind can be trained on corpora, without requiring their documents to be semantically tagged [12]. This overgenerate and rank approach, however, is tractable only when the message triples to be expressed are very few, since for n triples, there are $n!$ candidate texts; we return to this point when discussing sentence aggregation. Latent topic models, which do not require semantically tagged corpora, have also been used to rank candidate sentence orderings [13] or to directly order sentences or paragraphs [55,32]; some of these models attempt to combine local with global coherence constraints.

2.2. *Micro-planning*

The processing stages we have discussed so far select and order the message triples to be expressed. The next stage, *micro-planning*, consists of three sub-stages: *lexicalization*, *sentence aggregation*, and *generation of referring expressions*.

2.2.1. *Lexicalization*

During lexicalization, NLG systems usually turn messages (in our case, message triples) to abstract sentence specifications. The mechanisms used during this stage and the level of abstraction of the resulting specifications vary across NLG systems, from simple text templates with slots that are filled in to produce almost final sentences, to more complex rules, possibly with preconditions and actions, which may produce syntax trees [15,31,117,166].

In NaturalOWL, for every property of the domain ontology and every supported natural language, the domain author may specify one or more template-like sentence plans to indicate how message triples involving that property can be expressed. The properties are mapped to sentence plans using OWL annotation assertions. For example, the `usedDuringPeriod` property of page 19 is mapped below to a sentence plan whose identifier is `usedDuringPeriodEnglish`.

```
AnnotationAssertion(nlowl:hasSentencePlan
  :usedDuringPeriod
  dgr:usedDuringPeriodEnglish)
```

The annotation property `hasSentencePlan` is defined in NaturalOWL's generation resources ontology, as indicated by its `nlowl` prefix. Sentence plan identifiers are prefixed with `dgr`, since sentence plans are domain-dependent. We discuss below how sentence plans themselves, like `usedDuringPeriodEnglish`, are specified, but first a slight deviation is necessary, to briefly discuss NaturalOWL's lexicon entries.

Lexicon entries

For each verb, noun, or adjective that the domain author wishes to use in the sentence plans, a lexicon entry has to be provided, which specifies the inflectional forms of that word among other information. [34] All the lexicon entries are multilingual (currently bilingual); this could allow sentence plans to be reused across similar languages when no better option is available, as discussed elsewhere [4]. Figure 5 shows the lexicon entry for the verb whose English base form is "find". The entries for nouns and adjectives are similar; we provide examples below, when discussing the generation of referring expressions. The statements of Figure 5 specify that there is a lexicon entry whose identifier is `toFindLex`; the English and Greek parts of the lexicon entry have the identifiers `toFindEnglish` and `toFindGreek`, respectively. The English part shows that the base form is "find", that the simple past form is "found" etc. In practice, the lexicon entries are created by using the Protégé plug-in, as illustrated in Figure 6, instead of writing directly statements like those of Figure 5.

Most of the inflectional forms of English verbs, nouns, and adjectives could be automatically produced from the base forms by using relatively sim-

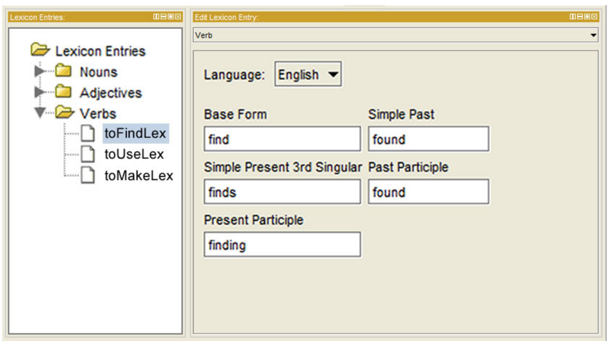
[34] The lexicon entries of closed-class words, like determiners and prepositions, are domain-independent.

```
ClassAssertion(nlowl:LexiconEntry dgr:toFindLex)
ObjectPropertyAssertion(nlowl:hasEnglishLexEntry
  dgr:toFindLex dgr:toFindEnglish)
ObjectPropertyAssertion(nlowl:hasGreekLexEntry
  dgr:toFindLex dgr:toFindGreek)

ClassAssertion(:EnglishLexEntry :toFindEnglish)
DataPropertyAssertion(:baseForm
  :toFindEnglish "find"^^xsd:string)
DataPropertyAssertion(:simplePres3rdSing
  :toFindEnglish "finds"^^xsd:string)
DataPropertyAssertion(:presParticiple
  :toFindEnglish "finding"^^xsd:string)
DataPropertyAssertion(:simplePast
  :toFindEnglish "found"^^xsd:string)
DataPropertyAssertion(:pastParticiple
  :toFindEnglish "found"^^xsd:string)

ClassAssertion(:GreekLexEntry :toFindGreek)
...
```

Figure 5. A bilingual lexicon entry for the verb "to find".


Lexicon Entries:
- Lexicon Entries
  - Nouns
  - Adjectives
  - Verbs
    - toFindLex
    - toUseLex
    - toMakeLex

Edit Lexicon Entry:

Verb

Language: English

Base Form: find	Simple Past: found
Simple Present 3rd Singular: finds	Past Participle: found
Present Participle: finding	

Figure 6. Creating a lexicon entry with NaturalOWL's Protégé plug-in. The English part of the entry is shown.

ple morphology rules. We hope to exploit an existing English morphology component, such as that of SIMPLENLG [63], for this purpose in future work. [35] Similar morphology rules for Greek were used in M-PIRO's authoring tool [4], and we hope to include them in a future version of NaturalOWL. Rules of this kind would reduce the time a domain author spends creating lexicon entries; for example, the Protégé plug-in could suggest automatically generated inflected forms, and the author could check them and correct irregular forms. We note, however, that in the domain ontologies we have considered, a few dozens of lexicon entries for verbs, nouns, and adjectives suffice; we provide estimates of the domain author's effort when discussing the trials that we conducted. Hence, even without facilities to automatically produce inflectional forms, creating the lexicon entries using the plug-in is rather trivial.

Another possibility would be to exploit a general-purpose lexicon, such as WordNet [56]. [36] General-purpose lexicons, however, often do not cover the highly technical concepts of domain ontologies.

Sentence plans

In NaturalOWL, a sentence plan consists of a sequence of slots, and instructions specifying how to fill them in; once the slots have been filled in, their contents are concatenated to produce a sentence. The number of slots of each sentence plan, their order, and the instructions that specify how to fill them in are represented in OWL, using concepts from NaturalOWL's generation resources ontology.

Figure 7 shows the OWL statements that define the `usedDuringPeriodEnglish` sentence plan of our earlier example; again, in practice the domain author defines sentence plans using the Protégé plug-in, as illustrated in Figure 8, which produces the corresponding OWL statements. The first two statements of Figure 7 assert that `usedDuringPeriodEnglish` is a sentence plan for English. The third statement allows the sentence plan to be aggregated; we discuss sentence aggregation below.

The next five statements define the first slot of the sentence plan. As with annotation events, slots are treated as individuals, but their identifiers (e.g., `_:slot1`) are automatically generated by the Protégé plug-in and they are not shown to the domain author. The statements declare the slot to be the first one; they also require it to be filled in with an automatically generated referring expression for the triple's owner (S). For example, if the triple to express is `<:stoaZeusEleutherios, :usedDuringPeriod, :classicalPeriod>`, an appropriate referring expression for S may be a demonstrative noun phrase like "this stoa", a pronoun ("it"), or the monument's natural language name ("the Stoa of Zeus Eleutherios"). We discuss the generation of referring expressions below, along with mechanisms to specify natural language names. The `useCase` requires the generated referring expression to be in nominative case (e.g., "it" or "this stoa", as opposed for example to the genitive case expressions "its" or "this stoa's", as in "This stoa's height is 5 meters").

The next seven statements define the second slot, to be filled in with a form of the verb whose lexicon entry is `toUseVerb`. The verb form must be in the

[35] See `http://code.google.com/p/simplenlg/`.

[36] Consult `http://www.w3.org/TR/wordnet-rdf/` for an OWL version of WordNet.

```
ClassAssertion(nlowl:SentencePlan
  dgr:usedDuringPeriodEnglish)
DataPropertyAssertion(nlowl:forLanguage
  dgr:usedDuringPeriodEnglish
  nlowl:englishLanguage)
DataPropertyAssertion(nlowl:aggregationAllowed
  dgr:usedDuringPeriodEnglish "true"^^xsd:boolean)

ObjectPropertyAssertion(nlowl:hasSlot
  dgr:usedDuringPeriodEnglish _:slot1)
DataPropertyAssertion(nlowl:hasOrder
  _:slot1 "1"^^xsd:nonNegativeInteger)
ClassAssertion(nlowl:forOwnerSlot _:slot1)
DataPropertyAssertion(nlowl:refExpressionType
  _:slot1 nlowl:autoRefExpression)
DataPropertyAssertion(nlowl:useCase
  _:slot1 nlowl:nominativeCase)

ObjectPropertyAssertion(nlowl:hasSlot
  dgr:usedDuringPeriodEnglish _:slot2)
DataPropertyAssertion(nlowl:hasOrder
  _:slot2 "2"^^xsd:nonNegativeInteger)
ObjectPropertyAssertion(nlowl:useLexiconEntry
  _:slot2 dgr:toUseVerb)
DataPropertyAssertion(nlowl:useTense
  _:slot2 nlowl:simplePast)
DataPropertyAssertion(nlowl:useVoice
  _:slot2 nlowl:passiveVoice)
DataPropertyAssertion(nlowl:usePolarity
  _:slot2 "true"^^xsd:boolean)
DataPropertyAssertion(nlowl:agreeWith
  _:slot2 _:slot1)

ObjectPropertyAssertion(nlowl:hasSlot
  dgr:usedDuringPeriodEnglish _:slot3)
DataPropertyAssertion(nlowl:hasOrder
  _:slot2 "3"^^xsd:nonNegativeInteger)
ObjectPropertyAssertion(nlowl:usePreposition
  _:slot3 nlowl:duringPrepEnglish)

ObjectPropertyAssertion(nlowl:hasSlot
  dgr:usedDuringPeriodEnglish _:slot4)
DataPropertyAssertion(nlowl:hasOrder
  _:slot4 "4"^^xsd:nonNegativeInteger)
ClassAssertion(nlowl:forFillerSlot _:slot4)
DataPropertyAssertion(nlowl:useCase
  _:slot4 nlowl:accusativeCase)
DataPropertyAssertion(nlowl:useBullets
  _:slot4 "false"^^xsd:boolean)
```

Figure 7. OWL statements defining a sentence plan for sentences like "It was used during the Classical period".

simple past and passive voice ("was used" or "were used"), in positive polarity (as opposed to the negative polarity "was *not* used") and its number must agree with the number of the expression in the first slot (the subject); for example, we want to generate "The Stoa of Zheus Eleutherios *was* used", but "Stoas *were* used".

The third slot is filled in with the preposition "during". The fourth slot must be filled in with an expression for the filler (O) of the message triple. In the case of `<:stoaZeusEleutherios, :usedDuringPeriod, :classicalPeriod>`, the slot would be filled in with the natural language name of `classicalPeriod`.[37] The expression's case is set to accusative; English prepositions usually require their noun phrase complements to be in accusative case (e.g., "on him"). In Greek and other languages, grammatical cases have more noticeable effects.

The sentence plan of Figure 7 produces sentences like the following two.

[$_{slot_1}$ This stoa] [$_{slot_2}$ was used] [$_{slot_3}$ during] [$_{slot_4}$ the Classical period].

[$_{slot_1}$ The Stoa of Zeus Eleutherios] [$_{slot_2}$ was used] [$_{slot_3}$ during] [$_{slot_4}$ the Classical period, the Hellenistic period, and the Roman period].

Changing the value of `useBullets` in the fourth slot to `true` would produce sentences with bullet lists, like the following one, when the filler is a disjunction or a conjunction.

The Stoa of Zeus Eleutherios was used during:
– the Classical period,
– the Hellenistic period, and
– the Roman period.

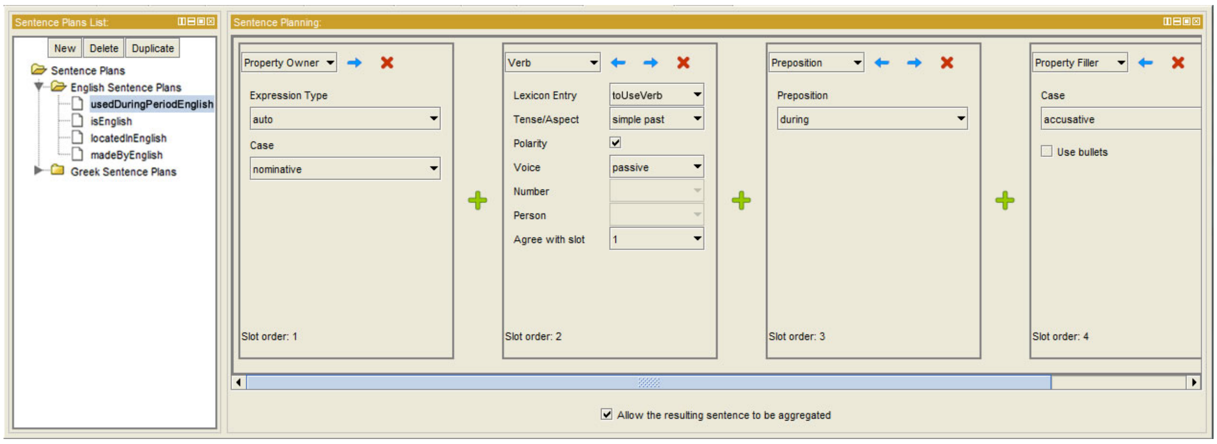
More generally, the instructions of a sentence plan may indicate that a slot should be filled in with one of the following (i–vii):

(i) *A referring expression for the S* (owner) of the message triple. A sentence plan may specify a particular type of referring expression to use (e.g., always use the natural language name of S) or, as in the example of Figure 7, it may allow the system to automatically produce the most appropriate type of referring expression depending on the context.

(ii) *A verb* for which there is a lexicon entry, in a particular inflectional form (e.g., a verb in a particular tense, voice etc.), possibly in a form that agrees with another slot (e.g., for subject-verb agreement). The verb's polarity can also be manually specified or, if the filler (O) of the message triple is a Boolean value, the polarity can be automatically set to match that value (e.g., to produce "It does *not* have a built-in flash" when O is `false`).

(iii) *A noun or adjective* from the lexicon. The case, number, and gender can be set to a specific value. Otherwise the form of the noun or adjec-

[37] Future versions of NaturalOWL may allow a referring expression for O other than the natural language name to be produced (e.g., a pronoun), as with S.



Property Owner
Expression Type: auto
Case: nominative
Slot order: 1

Verb
Lexicon Entry: toUseVerb
Tense/Aspect: simple past
Polarity: ✓
Voice: passive
Number:
Person:
Agree with slot: 1
Slot order: 2

Preposition
Preposition: during
Slot order: 3

Property Filler
Case: accusative
☐ Use bullets
Slot order: 4

✓ Allow the resulting sentence to be aggregated

Figure 8. Defining a sentence-plan with the Protégé plug-in.

tive can be set to agree with another slot (e.g. for adjective-noun agreement).

(iv) *A preposition.*

(v) *A fixed string.* For example, we may want the sentence to start with "It is now believed that".

(vi) *An expression for the O* (filler) of the triple. If O is an individual or class, then the expression is O's natural language name. If O is a datatype value (e.g., an integer), then the value itself is inserted in the slot. If O is a disjunction or conjunction of datatype values or individuals or classes, then the slot is filled in with a disjunction or conjunction of the datatype values or the natural language names of the individuals or classes.

(vii) *A concatenation of property values of O*, provided that O is an individual. For example, we may need to express a message triple like the first one below, whose anonymous node `_:n` is linked to both a numeric value (via `hasAmount`) and an individual standing for the currency (via `hasCurrency`) by the next two triples.[38]

```
<:tecra8, :hasPrice, _:n>
<_:n, :hasAmount, "850"^^xsd:float>
<_:n, :hasCurrency, :euroCurrency>
```

We would want the sentence plan to include a slot filled in with the concatenation of the `hasAmount` value and the natural language name of the `hasCurrency` value ("Euro" in English, "Ευρώ" in Greek) of the original triple's O (`_:n`). If a property vaue to be concatenated is a datatype value, then the value itself is concatenated. If it is an individual (e.g., `euroCurrency`), the individual's natural language name is concatenated. If the O of the original triple is a disjunction `or(_:n1, _:n2, ...)` or a conjunction `and(_:n1, _:n2, ...)`, then the slot is filled in with a disjunction or conjunction of the specified property values of `_:n1`, `_:n2`, etc.

Default sentence plan

If no sentence plan has been provided for a particular property of the domain ontology, NaturalOWL uses a default sentence plan, consisting of three slots:

– The first slot is filled in with an automatically generated referring expression for the triple's owner (S) in nominative case.
– The second slot is filled in with the OWL identifier of the property (without namespace) as a string, with underscores and dashes replaced by spaces, and additional spaces inserted at points where capitalization changes or groups of digits start.[39]
– The third slot is filled in with an appropriate expression for the triple's filler (O), as discussed above, in accusative case (if applicable).

For the `usedDuringPeriod` triple of page 19, the default sentence plan would produce the sentence:

> Stoa zeus eleutherios used during period classical period, hellenistic period, and roman period.

[38] The last two triples would be automatically retrieved from the domain ontology, since the sentence plan of the first triple requires them, even if they are not selected by content selection. Their interest scores would be automatically set to zero, to avoid generating sentences expressing them directly.

[39] Power [137] reports that heuristic tokenization rules of this kind usually produce reasonable tokens. Fliedl *et al.* [57] provide further examples of OWL identifiers to be tokenized. Cimiano *et al.* [34] propose a more elaborate approach, whereby grammar rules are applied to tokenized identifiers or `rdfs:label` strings (discussed below).

We assumed in the sentence above that the natural language names of the individuals have not been provided either; mechanisms to specify natural language names are discussed below. In this case, NaturalOWL uses the OWL identifiers of the individuals (e.g., `stoaZeusEleutherios`) as natural language names, applying the same tokenization rules that it uses for property identifiers.

Using rdfs:label strings

OWL properties (and other elements of OWL ontologies) can be labeled with strings in multiple natural languages using the `rdfs:label` annotation property, which is defined in the RDF and OWL standards. For example, the `usedDuringPeriod` property could be labeled with "was used during" as below; there could be similar labels for Greek and other languages.

```
AnnotationAssertion(rdfs:label :usedDuringPeriod
  "was used during"@en)
```

If an `rdfs:label` string has been specified for the property of a message triple, NaturalOWL uses that string in the second slot of the default sentence plan. The quality of the resulting sentences can, thus, be improved, if the `rdfs:label` strings are more natural phrases than the tokenized property identifiers. With the `rdfs:label` shown above, the default sentence plan would produce the following sentence.

Stoa zeus eleutherios was used during classical period, hellenistic period, and roman period.

Even with `rdfs:label` strings, however, the default sentence plan may produce sentences with disfluencies; for example, there may be violations of number agreement, or referring expressions with wrong cases (especially in Greek). Furthermore, the default sentence plan does not indicate which parts of the sentences are verbs or prepositions, and this does not allow the system to apply many of the sentence aggregation rules discussed below. A further limitation of the default sentence plan is that it does not allow the slots for S and O to be preceded or followed, respectively, by any other expression.

Sentence plans for domain-independent properties

The domain author does not need to provide sentence plans for domain-independent properties (e.g., `instanceOf`, `isA`, see Tables 4–5). These properties have fixed semantics, independent of the domain ontology; hence, built-in sentence plans can be used. We summarize the English built-in sentence plans in Table 6; the Greek ones are very similar. We do not show the sentence plans for negated domain-independent properties (e.g., `not(isA)`), which are very similar. To save space we show the sentence plans as templates, rather than OWL statements.

Additional slot restrictions not shown in Figure 6 require, for example, subject-verb number agreement and the verb forms ("is" or "was") to be in present tense. Note also that information provided when specifying the natural language names of individuals and classes, discussed below, shows if definite or indefinite articles or no articles at all should be used (e.g., "*the* N97 mini", "exhibit 24", "*a* St. Emilion" or "*the* St. Emilion" or simply "St. Emilion"), and what the default number of the names is (e.g., "A wine color is" or "Wine colors are").

In some domains, it may actually be desirable to slightly modify the built-in sentence plans. For example, in a museum context we may wish to generate sentences like "An aryballos *was* a kind of vase" instead of "An aryballos *is* a kind of vase". The built-in sentence plans can be modified via the Protégé plug-in or by editing directly the corresponding OWL statements, as with other sentence plans.

Sentence plans for modified properties

The sentence plans for modified properties (e.g., `minCardinality(manufacturedBy)`, see Tables 4–5) are automatically produced from the sentence plans of the unmodified properties (e.g., `manufacturedBy`), as shown in Table 7; again, we omit details such as subject-verb agreement, voice, tense etc. Hence, the domain author provides sentence plans only for the (unmodified) properties of the domain ontology.

Specifying the appropriateness of sentence plans

Multiple sentence plans may be provided for the same property of the domain ontology and the same language. Different appropriateness scores can then be assigned to alternative sentence plans per user type. This allows specifying, for example, that a sentence plan that generates sentences like "This amphora depicts Miltiades" is less appropriate when interacting with children, compared to an alternative sentence plan that uses a more common verb (e.g., "shows"). Sentence plans are assigned appropriateness scores using annotation events, much as when assigning interest scores to properties.

If multiple sentence plans are available for the same property and user type, NaturalOWL prefers the sentence plan with the highest appropriateness score the first time it needs to convey a message triple of that property to a particular user of that type. The second time that a triple of the same prop-

forms of message triples and corresponding built-in sentence plans	example message triples and possible resulting sentences
<S, `instanceOf`, O> *ref*(S) `toBeVerb` *name*(*indef*, O)	`<:eos450d, instanceOf, :PhotographicCamera>` The EOS 450D is a photographic camera.
<S, `instanceOf`, O> *ref*(S) `toBeVerb` *name*(*adj*, O)	`<:eos450d, instanceOf, :Cheap>` The EOS 450D is cheap.
<S, `oneOf`, O> *ref*(S) `toBeVerb` *name*(O)	`<:WineColor, oneOf, or(:white, :rose, :red)>` A wine color is white, rose, or red.
<S, `differentIndividuals`, O> *ref*(S) `toBeVerb` not identical to *name*(O)	`<:n97, differentIndividuals, :n97mini>` The N97 is not identical to the N97 mini.
<S, `sameIndividual`, O> *ref*(S) `toBeVerb` identical to *name*(O)	`<:eos450d, sameIndividual, :rebelXSi>` It is identical to the Rebel XSi.
<S, `isA`, O> *ref*(S) `toBeVerb` a kind of *name*(*noarticle*, O)	`<:StEmilion, isa, :Bordeaux>` St. Emilion is a kind of Bordeaux.
<S, `isA`, O> *ref*(S) `toBeVerb` *name*(*adj*, O)	`<:StEmilion, isa, :Red>` St. Emilion is red.

Table 6
Built-in English sentence plans for **domain-independent properties**. The notation *ref*(ξ) stands for a referring expression for ξ; *name*(ξ) is the natural language name of ξ; *name*(*indef*, ξ) and *name*(*noarticle*, ξ) mean that the name should be a noun phrase with an indefinite or no article, respectively. Sentence plans involving *name*(*adj*, ξ) are used when the natural language name of ξ is a sequence of one or more adjectives; otherwise the sentence plan of the previous row is used.

erty has to be conveyed to the same user, it prefers the second most appropriate sentence plan, provided that its appropriateness is positive, and so on, until we are left with no unused sentence plan with positive appropriateness for the property. At that point, the process restarts from using the most appropriate sentence plan. Sentence plans with negative or zero appropriateness are used only when there is no alternative sentence plan with a positive score. Automatically constructed sentence plans for modified properties inherit the appropriateness of the sentence plans they are constructed from.

Even when there is only one user type, multiple sentence plans per property may be desirable to avoid repeating the same sentences. Automatic paraphrase generation [3,112] could be used to suggest alternative sentence plans to the domain author. It may also be possible to obtain sentence plans from FrameNet [111], as suggested by Dannels [42].

A note on templates and sentence plans

Although we often informally show them as templates, NaturalOWL's sentence plans are not simply strings with slots filled in with elements of the message triples. Recall, for example, that a sentence plan may specify the lexicon entry of a verb, the desired tense and voice, and that the verb should match the number of a referring expression, relying on subsequent processing stages to produce the surface (final) form of the sentence (e.g., the exact verb form and referring expression). In that sense, NaturalOWL's sentence plans are similar to expressions of sentence planning languages (e.g., SPL [89]) that are used to formulate the inputs to generic surface realizers, like FUF/SURGE [54], KPML [15], REALPRO [107], NITROGEN/HALOGEN [105,104,106], and OPENCCG [173]; see also Guo *et al.* [67], Varges and Mellish [168], and the references therein.

Unlike the inputs to most generic surface realizers, however, NaturalOWL's sentence plans leave fewer decisions to subsequent processing stages. In many generic realizers, for example, the input would specify the base forms of the content words to use, and features like voice, tense, polarity etc. The correct order of the words (or syntactic constituents), several inflectional features (e.g., numbers, cases), and often the necessary function words (e.g., articles, prepositions) would be automatically selected, usually relying on large-scale grammars or statistical models. By contrast, information of this kind is explicitly specified in NaturalOWL's sentence plans. This has the disadvantage that our sentence plans often include information that could be automatically

example forms of triples involving unmodified and modified properties and the corresponding sentence plans	example triples involving unmodified and modified properties and possible resulting sentences
<S, `:manufacturedBy`, O>	`<:tecraA8, :manufacturedBy, :toshiba>`
ref(S) `toManufactureVerb` `byPrepEnglish` *name*(O)	Tecra A8 is manufactured by Toshiba.
<S, `minCardinality(:manufacturedBy)`, n:R>	`<:Laptop, minCardinality(:manufacturedBy), 1:Company>`
ref(S) `toManufactureVerb` `byPrepEnglish` at least *name*(n, R)	A laptop is manufactured by at least one company.
<S, `:currentMuseum`, O>	`<:exhibit24, :currentMuseum, :delosMuseum>`
ref(S) `toBeVerb` `inPrepEnglish` *name*(O)	Exhibit 24 is in the Museum of Delos.
<S, `maxCardinality(:currentMuseum)`, n:R>	`<:Exhibit, maxCardinality(:currentMuseum), 1:Museum>`
ref(S) `toBeVerb` `inPrepEnglish` at most *name*(n, R)	An exhibit is in at most one museum.
<S, `:madeFromGrape`, O>	`<:StEmilion, :madeFromGrape, :cabernetSauvignonGrape>`
ref(S) `toMakeVerb` `fromPrepEnglish` *name*(O)	St. Emilion is made from Cabernet Sauvignon grape.
<S, `exactCardinality(:madeFromGrape)`, n:R>	`<:StEmilion, exactCardinality(:madeFromGrape), 1>`
ref(S) `toMakeVerb` `fromPrepEnglish` exactly *name*(n, R)	St. Emilion is made from exactly one grape.
<S, `:madeFromGrape`, O>	`<:Assyrtiko, :madeFromGrape, :assyrtikoGrape>`
ref(S) `toMakeVerb` `fromPrepEnglish` *name*(O)	The Assyrtiko wine is made from Assyrtiko grape.
<S, `allValuesFrom(:madeFromGrape)`, O>	`<:GreekWine, allValuesFrom(:madeFromGrape), :GreekGrape>`
ref(S) `toMakeVerb` `fromPrepEnglish` only *name*(*plural*, *indef*, O)	Greek wines are made from only Greek grapes.
<S, `:manufactures`, O>	`<:toshiba, :manufactures, :tecraA8>`
ref(S) `toManufactureVerb` *name*(O)	Toshiba manufactures Tecra A8.
<S, `someValuesFrom(:manufactures)`, O>	`<:LaptopManuf, someValuesFrom(:manufactures), :Laptop>`
ref(S) `toManufactureVerb` at least *name*(1, O)	A laptop manufacturer manufactures at least one laptop.

Table 7
Automatically constructed sentence plans for **modified properties**. In each cell of the left column, the sentence plan in the last line is automatically constructed from the sentence plan in the second line. Square brackets indicate optional arguments. The notation *ref*(ξ) stands for a referring expression for ξ; *name*(ξ) is the natural language name of ξ; *name*(n, κ) is a noun phrase for n individuals of class κ, and *name*(*plural*, *indef*, κ) is a plural indefinite noun phrase for individuals of κ. When R is unspecified, it is taken to be the class that has been declared as the range of the unmodified property, if the range is a named class; if the range is an unnamed class, *name*(n, R) generates an expression like "n entities".

obtained from large-scale grammars or corpora.

On the other hand, the input to generic surface realizers often includes information pertaining to non-elementary linguistic concepts (e.g., syntactic categories, subcategories, and features of a particular syntax theory) and concepts of a large-scale high-level domain-independent ontology, usually called the *upper model* [14]. Hence, linguistic expertise (e.g., in Systemic Grammars [71] in the case of KPML [15]) and effort to understand the upper model are often required. By contrast, NaturalOWL's sentence plans require the domain author to be familiar with only elementary linguistic concepts (e.g., tense, voice, number, case), and they do not require familiarity with an upper model.

Overall, NaturalOWL's sentence plans, like the entire system, are intended to be easier to master by domain authors that are familiar with Semantic Web concepts and the domain ontology, but not computational linguistics. It would, however, be worth exploring in future work how generic surface realizers could be exploited, especially realizers that do not require linguistic expertise. In Ratnaparkhi's realizer [142], for example, the input is a set of attribute-value pairs specifying the semantics of the sentence to be produced; a corpus annotated with attributes, however, is required to extract templates from. In the realizer of Wan *et al.* [172], the input is an unordered set of words; the system orders them by considering possible dependency trees that span them.

We also note that NaturalOWL's sentence plans are simpler than, for example, the templates of Busemann and Horacek [31] or McRoy *et al.* [117], in that they do not allow, for instance, conditionals or recursive invokation of other templates. See also Reiter [144] for a discussion of the advantages and disadvantages of template-based vs. theoretically more principled NLG systems, and van Deemter *et al.* [166] for a discussion of how sentence templates can be enriched with syntactic and other information, blurring the distinction between simple templates and more complex sentence plans.

When corpora of target texts annotated with the message triples they express are available, templates can also be automatically extracted, for example as in the systems of Ratnaparkhi [142] and Angeli *et al.* [5]. Statistical methods that can, in effect, jointly perform content selection, lexicalization, and surface realization, have also been proposed [108,98,99], for cases where training target sentences and corresponding semantic inputs are available, but they are currently limited to generating single sentences from semantic inputs that have the form of flat records containing fields and field values.

Specifying natural language names

The domain author can assign *natural language* (NL) names to the domain ontology's individuals and named classes; recall that by named classes we mean classes that have OWL identifiers. If an individual or named class is not assigned an NL name, then its `rdfs:label` or a tokenized form of its identifier are used instead, as already discussed. The NL names that the domain author provides are specified much as sentence plans, i.e., as collections of slots whose contents are concatenated. For example, we may specify that the English NL name of the class `ItalianWinePiemonte` is the concatenation of the following slots; we explain the slots further below.

[$_{indef}$ an] [$_{adj}$ Italian] [$_{headnoun}$ wine] [$_{prep}$ from] [$_{def}$ the] [$_{noun}$ Piemonte] [$_{noun}$ region]

This would allow NaturalOWL to generate the sentence shown below from the following message triple; a tokenized form of `wine32`'s identifier is used.

`<:wine32, instanceOf, :ItalianWinePiemonte>`

Wine 32 is an Italian wine from the Piemonte region.

Similarly, we may assign the following NL names to the individuals `classicalPeriod`, `stoaZeusEleutherios`, `gl2011`, and the classes `ComputerScreen` and `Red`. The domain author can request the words of particular slots to be capitalized (e.g., "the Classical period"). Also, NaturalOWL makes no distinction between common and proper nouns; both are entered as nouns in the lexicon, and they may be multi-word.

[$_{def}$ the] [$_{adj}$ Classical] [$_{headnoun}$ period]

[$_{def}$ the] [$_{headnoun}$stoa] [$_{prep}$of] [$_{noun}$Zeus Eleutherios]

[$_{headnoun}$ GL-2011]

[$_{indef}$ a] [$_{noun}$ computer] [$_{headnoun}$ screen]

[$_{headadj}$ red]

These NL names could be used to express the message triples shown below; consult also Table 6.

```
<:stoaZeusEleutherios, :usedDuringPeriod,
:classicalPeriod>
```

The Stoa of Zeus Eleutherios was used during the Classical period.

`<:gl2011, instanceOf, :ComputerScreen>`

GL-2011 is a computer screen.

`<:gl2011, instanceOf, :Red>`

GL-2011 is red.

The color of `gl2011` could also be expressed via a property of the domain ontology, as in:

`<:gl2011, :hasColor, :redColor>`

We would assign to the individual `redColor` the same NL name we assigned to the class `Red`.

Unlike tokenized identifiers and `rdfs:label` strings, the NL names carry additional linguistic information (e.g., grammatical categories of words), which allows, for example, the aggregation stage, discussed below, to combine sentences as in the following cases.

It was used during the Classical period, the Hellenistic period, and the Roman period. ⇒ It was used during the Classical, the Hellenistic, and the Roman period.

It is a computer screen. It is red. ⇒ It is a red computer screen.

More precisely, each NL name is a sequence of slots, with accompanying instructions specifying how the slots are to be filled in. Each slot can be filled in with one of the following (i–v):

(i) *An article*, definite or indefinite (in English "the" or "a"). The article in the first slot (if present) is treated as the article of the overall NL name, but multiple articles may be present in the same name, as in the Piemonte example above.

(ii) *A noun or adjective flagged as the head* (main word) of the NL name. Exactly one head must be specified per NL name and it must have a lexicon entry. The number and case of the head, which is also taken to be the number and case of the overall NL name, can be automatically adjusted per context. For example, different sentence plans may require the same NL name to be in nominative case when used as a subject, but in accusative when used as the object of a verb; and some aggregation rules, discussed below, may require a singular NL name to be turned into plural. Using the lexicon entries, which list the inflectional forms of nouns and adjectives, NaturalOWL can adjust the NL names accordingly. The gender of head adjectives can also be automatically adjusted, whereas the gender of head nouns is fixed and specified by their lexicon entries.

(iii) *Any other noun or adjective*, among those listed in the lexicon. The NL name may require a particular inflectional form (e.g., number, case) to be used, or it may require an inflectional form that agrees with another slot of the NL name (e.g., for adjective-noun agreement).

(iv) *A preposition.*

(v) *Any fixed string.* For example, words of grammatical categories that are not included in the lexicon (e.g., adverbs) are entered as strings.

The OWL representation of NL names is very similar to that of sentence plans; we do not discuss it to save space. Again, in practice NL names are entered via the Protégé plug-in. As with sentence plans, multiple NL names can be specified for the same individual or class, and they can be assigned different appropriateness scores per user type; this allows more common terminology (e.g., common names of diseases) to be used when generating texts for non-experts, as opposed to texts for experts (e.g., doctors). NaturalOWL cycles through the available NL names (with positive appropriateness scores) of each individual or class, as with sentence plans.

The domain author can specify (via the Protégé plug-in or in the OWL representation of NL names) if the NL names of particular individuals or classes should involve definite, indefinite, or no articles, and if the NL names should be in singular or plural by default. For example, the domain author may prefer the texts to mention the class of Aryballoi as a single particular generic object (singular form with a definite article), or by using an indefinite singular or plural form, as shown below.

```
<:Aryballos, isA, :Vase>
```

The aryballos is a kind of vase.

An aryballos is a kind of vase.

Aryballoi are a kind of vase.

Notice that some of the sentence plans of Tables 6 and 7 require a particular form (e.g., indefinite or no article), which is produced by automatically modifying the NL names provided by the domain author; similar conversions are imposed by some of the aggregation rules discussed next.

In domain ontologies with very large numbers of individuals, manually providing NL names for all the individuals may be impractical. Hence, it may be necessary to use tokenized identifiers, existing `rdfs:label` strings, or `rdfs:label` strings automatically constructed (by using additional software) from string-valued properties of the domain ontology (e.g., properties linking person individuals to strings containing their names and surnames). We return to this issue when presenting the trials.

2.2.2. *Sentence aggregation*

The sentence plans of the previous section lead to a separate sentence for each message triple. NLG systems often aggregate sentences into longer ones to avoid lexical redundancy and improve readability. [40] In NaturalOWL, the maximum number of sentences that can be aggregated to form a single longer sentence is specified per user type via a property called `maxMessagesPerSentence`. The following statement allows each new sentence to be formed by aggregating at most two original sentences, when interacting with children. In the museum applications NaturalOWL was originally developed for, setting `maxMessagesPerSentence` to 3 or 4 led to reasonable texts for adult visitors, whereas a value of 2 was used when interacting with children.

```
DataPropertyAssertion(nlowl:maxMessagesPerSentence
  dgr:child "2"^^xsd:nonNegativeInteger)
```

The idea that the number of messages per sentence or text (see also `maxMessagesPerPage` on page 16) should depend on the user type is inherited from ILEX and M-PIRO, and it was originally inspired by work in the psychology of text comprehension, especially by Kintsch *et al.* [94,95]. Message triples can be viewed as expressing simple ideas, and it has been argued, roughly speaking, that sentences or texts with many ideas are harder to process. Hence, less skilled readers should benefit from texts with fewer messages per sentence, and fewer messages per text. [41]

NaturalOWL's sentence aggregation is performed by a set of manually crafted rules, intended to be

[40] There is also evidence that aggregation may improve factual recall and, more generally, learning [88,48], though we have no such evidence for NaturalOWL's aggregation rules.
[41] See, for example, Nenkova *et al.* [130] for other factors that affect the perceived linguistic quality of texts.

domain-independent. We do not claim that this particular set of rules, which was initially based on M-PIRO's corresponding rules [118], is in any way complete, and we hope that it will be extended in future work; see, for example, Dalianis [41] for a study of aggregation types and a rich set of aggregation rules. Nevertheless, NaturalOWL's current rules already illustrate several aggregation opportunities that arise when generating texts from OWL ontologies. We also note that when appropriate datasets are available, it may be possible to automatically train aggregation modules [171,11], though this was not the case with the domain ontologies we considered.

To save space, we discuss only English sentence aggregation; Greek aggregation is similar. Also, we show mostly example *sentences* before and after aggregation, but the rules actually operate on *sentence plans* and they also consider the message triples the sentence plans express. The rules are intended to aggregate short single-clause sentences, each with exactly one verb. Sentence plans that produce more complicated sentences (e.g., involving long canned texts) may be flagged by setting their `aggregationAllowed` property to false (see Figure 7 on page 23), to signal that aggregation should not affect the resulting sentences.

NaturalOWL's aggregation rules apply almost exclusively to sentences that are adjacent in the ordering produced by the text planner; the only exception are aggregation rules that involve messages about cardinality restrictions. Hence, depending on the text planner's ordering there may be more or fewer aggregation opportunities; see, for example, Cheng and Mellish [33] for discussion on the interaction between aggregation and text planning. Also, NaturalOWL's aggregation rules operate on sentences of the same topical section, because aggregating topically unrelated sentences often sounds unnatural.

The system's aggregation is currently greedy. For each one of the rules discussed below, starting from those discussed first, the system scans the original (already ordered) sentences from first to last, and applies the rule wherever possible, provided that the rule's application does not lead to a sentence expressing more than `maxMessagesPerSentence` original messages. If a rule can be applied in multiple ways, for example to aggregate two or three sentences, the application that aggregates the most sentences without violating `maxMessagesPerSentence` is preferred.

Avoid repeating a noun with multiple adjectives: Recall that message triples of the form $\langle S, P, O_1 \rangle, \ldots, \langle S, P, O_n \rangle$ for the same S and the same property P of the domain ontology will have already been merged into a single message triple $\langle S, P, \texttt{and}(O_1, \ldots, O_n) \rangle$. If the NL names of $O_1, \ldots, O_n$ are, apart from possible initial determiners, sequences of adjectives followed by the same head noun, then the head noun does not need to be repeated. Let us consider the following message triple.

```
<:stoaZeusEleutherios, :usedDuringPeriod,
 and(:classicalPeriod, :hellenisticPeriod,
 :romanPeriod)>
```

Assuming that the NL names of `classicalPeriod`, `hellenisticPeriod`, and `romanPeriod` correspond to "the Classical period", "the Hellenistic period", and "the Roman period", the original sentence will repeat "period" three times. The aggregation rule omits all but the last occurrence of the head noun.

It was used during the Classical period, the Hellenistic period, and the Roman period. ⇒ It was used during the Classical, the Hellenistic, and the Roman period.

The rule above actually shortens a *single* sentence, which expresses, however, a message triple formed by merging multiple triples of the same S and P.

Cardinality restrictions and values: This is actually a set of rules, summarized in Tables 8 and 9. [42] These rules aggregate *all* the sentences (not necessarily adjacent ones) that express message triples of the form $\langle S, M(P), O \rangle$ and $\langle S, P, O \rangle$, for the same S and P, with M being any of `minCardinality`, `maxCardinality`, or `exactCardinality`. When these rules are applied, `MaxMessagesPerSentence` is ignored.

Class and passive sentence: This rule aggregates (i) a sentence expressing a message triple of the form $\langle S, \texttt{instanceOf}, C \rangle$ or $\langle S, \texttt{isA}, C \rangle$ and (ii) a passive immediately subsequent sentence expressing a single message triple of the form $\langle S, P, O \rangle$, for the same S, where P is an (unmodified) property of the domain ontology. The subject and auxiliary verb of the second sentence are omitted.

Bancroft Chardonnay is a kind of Chardonnay. It is made in Bancroft. ⇒ Bancroft Chardonnay is a kind of Chardonnay made in Bancroft.

Class and prepositional phrase: In a variant of the previous rule, the second sentence involves the verb "to be" in the active simple present immediately

[42] We assume there are no inconsistencies in the triples (e.g., minimum and maximum cardinalities set to 4 and 3, respectively, for the same S and P) and no redundancies (e.g., message triples for both exact and maximum cardinalities).

example rule application	abstract rule description
`<:model35, exactCardinality(:soldInCountry), 3>`	`<`S`, exactCardinality(`*prop*`), `n`:`R`>`
Model 35 is sold in exactly three countries.	*ref*(S) … exactly *name*(n, R)
`<:model35, :soldInCountry, and(:greece, :italy, :spain)>`	`<`S`, `*prop*`, and(`$O_1, \dots, O_n$`)>`
Model 35 is sold in Greece, Italy, and Spain.	*ref*(S) … *name*($O_1, \dots, O_n$)
⇒ Model 35 is sold in exactly three countries:	⇒ *ref*(S) … exactly *name*(n, R):
Greece, Italy, and Spain.	*name*($O_1, \dots, O_n$)
`<:model35, exactCardinality(:soldInCountry), 3>`	`<`S`, exactCardinality(`*prop*`), `n`:`R`>`
Model 35 is sold in exactly three countries.	*ref*(S) … exactly *name*(n, R)
`<:model35, :soldInCountry, and(:greece, :italy)>`	`<`S`, `*prop*`, and(`$O_1, \dots, O_m$`)>` ($m < n$)
Model 35 is sold in Greece and Italy.	*ref*(S) … *name*($O_1, \dots, O_m$)
⇒ Model 35 is sold in exactly three countries, including	⇒ *ref*(S) … exactly *name*(n, R), including
Greece and Italy.	*name*($O_1, \dots, O_m$)
`<:model35, maxCardinality(:soldInCountry), 3>`	`<`S`, maxCardinality(`*prop*`), `n`:`R`>`
Model 35 is sold in at most three countries.	*ref*(S) … at most *name*(n, R)
`<:model35, :soldInCountry, and(:greece, :italy, :spain)>`	`<`S`, `*prop*`, and(`$O_1, \dots, O_n$`)>`
Model 35 is sold in Greece, Italy, and Spain.	*ref*(S) … *name*($O_1, \dots, O_n$)
⇒ Model 35 is sold in exactly three countries:	⇒ *ref*(S) … exactly *name*(n, R):
Greece, Italy, and Spain.	*name*($O_1, \dots, O_n$)
`<:model35, maxCardinality(:soldInCountry), 3>`	`<`S`, maxCardinality(`*prop*`), `n`:`R`>`
Model 35 is sold in at most three countries.	*ref*(S) … at most *name*(n, R)
`<:model35, :soldInCountry, and(:greece, :italy)>`	`<`S`, `*prop*`, and(`$O_1, \dots, O_m$`)>` ($m < n$)
Model 35 is sold in Greece and Italy.	*ref*(S) … *name*($O_1, \dots, O_m$)
⇒ Model 35 is sold in at most three countries, including	⇒ *ref*(S) … exactly *name*(n, R), including
Greece and Italy.	*name*($O_1, \dots, O_m$)
`<:model35, minCardinality(:soldInCountry), 3>`	`<`S`, minCardinality(`*prop*`), `n`:`R`>`
Model 35 is sold in at least three countries.	*ref*(S) … at least *name*(n, R)
`<:model35, :soldInCountry, and(:greece, :italy, :spain)>`	`<`S`, `*prop*`, and(`$O_1, \dots, O_n$`)>`
Model 35 is sold in Greece, Italy, and Spain.	*ref*(S) … *name*($O_1, \dots, O_n$)
⇒ Model 35 is sold in at least three countries:	⇒ *ref*(S) … at least *name*(n, R):
Greece, Italy, and Spain.	*name*($O_1, \dots, O_n$)
`<:model35, minCardinality(:soldInCountry), 3>`	`<`S`, minCardinality(`*prop*`), `n`:`R`>`
Model 35 is sold in at least three countries.	*ref*(S) … at least *name*(n, R)
`<:model35, :soldInCountry, and(:greece, :italy)>`	`<`S`, `*prop*`, and(`$O_1, \dots, O_m$`)>` ($m < n$)
Model 35 is sold in Greece and Italy.	*ref*(S) … *name*($O_1, \dots, O_m$)
⇒ Model 35 is sold in at least three countries, including	⇒ *ref*(S) … at least *name*(n, R), including
Greece and Italy.	*name*($O_1, \dots, O_m$)

Table 8
Aggregation rules that merge **one sentence expressing a cardinality constraint** on a property *prop* of the domain ontology, and another sentence expressing the values of *prop*. The notation *ref*(ξ) stands for a referring expression for ξ; *name*(n, R) is a noun phrase for n individuals of class R; *name*($\xi_1, \dots, \xi_k$) is a conjunction of the natural language names of $\xi_1, \dots, \xi_k$. These rules are tried after the rules of Table 9.

example rule application	abstract rule description
`<:model35, minCardinality(:soldInCountry), 2>`	`<`S`, minCardinality(`prop`), `n_1`:`R`>`
Model 35 is sold in at least two countries.	$\mathit{ref}(S)$ … at least $\mathit{name}(n_1, R)$
`<:model35, maxCardinality(:soldInCountry), 3>`	`<`S`, maxCardinality(`prop`), `n_2`:`R`>`
Model 35 is sold in at most three countries.	$\mathit{ref}(S)$ … at most $\mathit{name}(n_2, R)$ $(n_1 < n_2)$
⇒ Model 35 is sold in at least two and at most three countries.	⇒ $\mathit{ref}(S)$ … at least $\mathit{name}(n_1, R)$ and at most $\mathit{name}(n_2, R)$
`<:model35, minCardinality(:soldInCountry), 2>`	`<`S`, minCardinality(`prop`), `n_1`:`R`>`
Model 35 is sold in at least two countries.	$\mathit{ref}(S)$ … at least $\mathit{name}(n_1, R)$
`<:model35, maxCardinality(:soldInCountry), 3>`	`<`S`, maxCardinality(`prop`), `n_2`:`R`>`
Model 35 is sold in at most three countries.	$\mathit{ref}(S)$ … at most $\mathit{name}(n_2, R)$ $(n_1 < n_2)$
`<:model35, :soldInCountry, and(:greece, :italy, :spain)>`	`<`S`, `prop`, and(`$O_1, \ldots, O_{n_2}$`)>`
Model 35 is sold in Greece, Italy, and Spain.	$\mathit{ref}(S)$ … $\mathit{name}(O_1, \ldots, O_{n_2})$
⇒ Model 35 is sold in exactly three countries:	⇒ $\mathit{ref}(S)$ … exactly $\mathit{name}(n_2, R)$:
Greece, Italy, and Spain.	$\mathit{name}(O_1, \ldots, O_{n_2})$
`<:model35, minCardinality(:soldInCountry), 2>`	`<`S`, minCardinality(`prop`), `n_1`:`R`>`
Model 35 is sold in at least two countries.	$\mathit{ref}(S)$ … at least $\mathit{name}(n_1, R)$
`<:model35, maxCardinality(:soldInCountry), 4>`	`<`S`, maxCardinality(`prop`), `n_2`:`R`>`
Model 35 is sold in at most four countries.	$\mathit{ref}(S)$ … at most $\mathit{name}(n_2, R)$ $(n_1 < n_2)$
`<:model35, :soldInCountry, and(:greece, :italy, :spain)>`	`<`S`, `prop`, and(`$O_1, \ldots, O_m$`)>` $(n_1 < m < n_2)$
Model 35 is sold in Greece, Italy, and Spain.	$\mathit{ref}(S)$ … $\mathit{name}(O_1, \ldots, O_m)$
⇒ Model 35 is sold in at least two and at most four countries,	⇒ $\mathit{ref}(S)$ … at least $\mathit{name}(n_1, R)$ and at most $\mathit{name}(n_2, R)$,
including Greece, Italy, and Spain.	including $\mathit{name}(O_1, \ldots, O_m)$
`<:model35, minCardinality(:soldInCountry), 2>`	`<`S`, minCardinality(`prop`), `n_1`:`R`>`
Model 35 is sold in at least two countries.	$\mathit{ref}(S)$ … at least $\mathit{name}(n_1, R)$
`<:model35, maxCardinality(:soldInCountry), 3>`	`<`S`, maxCardinality(`prop`), `n_2`:`R`>`
Model 35 is sold in at most three countries.	$\mathit{ref}(S)$ … at most $\mathit{name}(n_2, R)$ $(n_1 < n_2)$
`<:model35, :soldInCountry, and(:greece, :italy)>`	`<`S`, `prop`, and(`$O_1, \ldots, O_{n_1}$`)>`
Model 35 is sold in Greece and Italy.	$\mathit{ref}(S)$ … $\mathit{name}(O_1, \ldots, O_{n_1})$
⇒ Model 35 is sold in at most three countries, including	⇒ $\mathit{ref}(S)$ … at most $\mathit{name}(n_2, R)$, including
Greece and Italy.	$\mathit{name}(O_1, \ldots, O_{n_1})$

Table 9
Aggregation rules that merge **two sentences expressing cardinality constraints** on the same property *prop* of the domain ontology, and possibly a third sentence expressing the values of *prop*. The first rule of this table is applied only if the other rules of this table cannot be applied. All the rules of this table are tried before the rules of Table 8. The notation $\mathit{ref}(\xi)$ stands for a referring expression for ξ; $\mathit{name}(n, R)$ is a noun phrase for n individuals of class R; $\mathit{name}(\xi_1, \ldots, \xi_k)$ is a conjunction of the natural language names of $\xi_1, \ldots, \xi_k$.

followed by a preposition, instead of being a passive sentence; the other conditions are as in the previous rule. The subject and verb of the second sentence are omitted.

Bancroft Chardonnay is a kind of Chardonnay. It is from Bancroft. ⇒ Bancroft Chardonnay is a kind of Chardonnay from Bancroft.

Class and multiple adjectives: This rule aggregates (i) a sentence of the same form as in the previous two rules, i.e., a sentence expressing a message triple of the form $\langle S, \texttt{instanceOf}, C\rangle$ or $\langle S, \texttt{isA}, C\rangle$ and (ii) one or more immediately preceding or subsequent sentences, each expressing a single message triple $\langle S, P_i, O_i\rangle$, for the same S, where P_i are (unmodified) properties of the domain ontology. Each of the preceding or subsequent sentences must involve the verb "to be" in the active simple present immediately followed by only an adjective. The adjectives are absorbed into sentence (i) maintaining their order.

This is a motorbike. It is red. It is expensive. ⇒ This is a red, expensive motorbike.

Same verb conjunction/disjunction: When there is a sequence of sentences, all involving the same verb form and each expressing a single message triple of the form $\langle S, P_i, O_i\rangle$, where S is the same in all the triples and P_i are (unmodified) properties of the domain ontology, a conjunction can be formed by mentioning the subject and verb only once. [43]

It has medium body. It has moderate flavor. ⇒ It has medium body and moderate flavor.

The "and" is omitted when a preposition follows, as illustrated below. The particles of phrasal verbs (e.g., "shut *down*") are not considered prepositions.

He was born in Athens. He was born in 1918. ⇒ He was born in Athens in 1918.

A similar rule applies to sentences produced from disjunctions of message triples, as illustrated below. A variant of the first rule, which avoids repeating the same head noun with multiple adjectives, is also applied (if possible) to the resulting sentence.

The house wine has strong flavor or it has medium flavor. ⇒ The house wine has strong flavor or medium flavor. ⇒ The house wine has strong or medium flavor.

Different verbs conjunction: When there is a sequence of sentences, not involving the same verb, but each expressing a message triple of the form $\langle S, P_i, O_i\rangle$, where S is the same in all the triples and P_i are (unmodified) properties of the domain ontology, a conjunction can be formed, as shown below. [44]

Bancroft Chardonnay is dry. It has moderate flavor. It comes from Napa. ⇒ Bancroft Chardonnay is dry, it has moderate flavor, and it comes from Napa.

We discuss the generation of pronouns below. The subjects of the second, third etc. sentences aggregated by the rule above can be omitted, though the result may sometimes sound less fluent.

A note on the limitations of the pipeline architecture

By operating on both message triples and ordered sentence plans, NaturalOWL's aggregation rules consider not only the semantics of the sentences, but also the particular text planning and lexicalization choices that have been made to express them. For example, the "same verb conjunction/disjunction" rule can be applied only to adjacent sentences involving the same verb, and the "class and prepositional phrase" rule can be applied only if the second sentence involves a prepositional phrase. Recall that several sentence plans may apply to the same message triple, and they may involve different verbs, use of prepositional phrases vs. other phrasings etc. Hence, if aggregation took place before lexicalization [11,177], it would be impossible to decide if several of the aggregation rules that we currently use could be applied or not, unless aggregation was more intertwined with lexicalization, allowing aggregation rules to select among alternative sentence plans.

With a broader perspective, if aggregation and lexicalization were both intertwined with content selection and text planning, it might be possible to select message triples not only with high interest scores, but also with sentence plans of high appropriateness that can be aggregated with many other sentence plans; and aggregatable sentences could be placed in adjacent positions to maximize aggregation opportunities, while also attempting to maximize local coherence. This would require, however, replacing the pipeline architecture and its sequential, in effect greedy, choices by an architecture where decisions pertaining to all the stages of NLG would be considered in parallel [114,17]. Another possibility would be to retain the pipeline, but allow

[43] Consult Harbusch and Kempen [73] for a broader treatment of elliptical clausal coordinations.

[44] Disjunctions of message triples give rise to a disjunction of sentences without any need for further aggregation.

its modules to overgenarate by producing alternatives (e.g., several alternative sets of selected message triples, several alternative orderings of each set of message triples, several lexicalizations, aggregations etc.) and rank the resulting texts at the end of the pipeline.[45] The latter approach, however, leads to an exponentially large number of alternatives, when several modules are allowed to overgenerate.

2.2.3. *Generating referring expressions*

A sentence plan may require an appropriate referring expression to be generated for the S of an $\langle S, P, O\rangle$ message triple, as already discussed. Depending on the context, it may be better, for example, to use the NL name of S (e.g., "the Stoa of Zeus Eleutherios"), a pronoun (e.g., "it"), a demonstrative noun phrase (e.g., "this stoa") etc. Similar alternatives could be made available for O, but NaturalOWL currently always uses O itself, if it is a datatype value; or the NL name of O, its tokenized identifier, or its `rdfs:label`, if O is an entity or class; and similarly for conjunctions and disjunctions in O. Hence, below we focus only on referring expressions for S. Adding mechanisms to generate more varied expressions for O is a possible future extension.

Generating the most appropriate referring expression is a complex problem. It involves considering multiple candidate referring expressions, whether or not each candidate expression identifies a unique referent, how easily a hearer can understand each expression, the length of, or time to utter each expression, the desire to avoid repeating the same expressions etc. Krahmer and van Deemter [101] provide an extensive survey of research on referring expression generation. NaturalOWL currently uses a very limited range of referring expressions, which includes only NL names (or tokenized identifiers or `rdfs:label` strings), pronouns, and noun phrases involving only a demonstrative and the NL name of a class (e.g., "this vase"). For example, referring expressions that mention properties of S (e.g., "the vase from Rome") are not generated.[46] Although NaturalOWL's current referring expression generation mechanisms work reasonably well in the ontologies we have experimented with, they are best viewed as a placeholder for more elaborate algorithms [37,39,164,79,102,165,133,40], especially algorithms based on description logics [7,147], which we hope to see included in future versions.[47]

To explain how referring expressions for S are currently generated, let us consider the following text, which expresses the message triples below. We do not aggregate sentences in this section, to illustrate more cases where referring expressions are needed; note, however, that aggregation would reduce the number of pronouns, making the text less repetitive.

(1) *Exhibit 7* is a statue. (2) *<u>It</u>* was sculpted by Nikolaou. (3) *<u>Nikolaou</u>* was born in Athens. (4) *<u>He</u>* was born in 1918. (5) *<u>He</u>* died in 1998. • (6) *Exhibit 7* is now in the National Gallery. (7) *<u>It</u>* is in excellent condition.

```
<:exhibit7, instanceOf, :Statue>
<:exhibit7, :hasSculptor, :nikolaou>
<:nikolaou, :cityBorn, :athens>
<:nikolaou, :yearBorn,
  "1918"^^xsd:nonNegativeInteger>
<:nikolaou, :yearDied,
  "1998"^^xsd:nonNegativeInteger>
<:exhibit7, :currentLocation, :nationalGallery>
<:exhibit7, :currentCondition, :excellentCondition>
```

Recall that in a sentence u_n that NaturalOWL generates for a message triple $\langle S_n, P_n, O_n\rangle$, typically $C_f(u_n) = \{S_n, O_n\}$ and $c_p(u_n) = S_n$, since S_n is typically realized as the most salient noun phrase of u_n, usually the subject. As in Section 2.1.2, we show in italics the noun phrase realizing $c_p(u_n)$; we show underlined the noun phrase realizing $c_b(u_n)$; and we mark NOCB transitions with bullets.

NaturalOWL pronominalizes S_n (for $n > 1$) only if $S_n = S_{n-1}$, as in sentences 2, 4, 5, and 7. Since typically $c_p(u_i) = S_i$, we typically obtain $c_p(u_n) = c_p(u_{n-1})$, whenever S_n is pronominalized, if the pronoun is resolved by the reader as intended. When reading a text, there is a tendency to prefer readings where $c_p(u_n) = c_p(u_{n-1})$, if no other restriction is violated (e.g., gender, number, case agreement, world knowledge constraints). This helps the pronouns that NaturalOWL generates to be correctly resolved by readers, even when they would appear to be potentially ambiguous. For example, the pronoun of sentence 7 is most naturally understood as referring to the exhibit, as it is intended to, not to

[45] See, for example, Walker *et al.* [171] for a dialogue system that uses an overgenerate and rank approach to aggregation.
[46] When generating comparisons, NaturalOWL also produces expressions referring to previously encountered individuals or sets of individuals (e.g., "Unlike *all the previous vessels that you have seen*, this lekythos is decorated with the black-figure technique"), and also some spatial referring expressions (e.g., "Like *the tetradrachm that you saw, which is now behind me*, this drachma was found in Athens") [85], but we do not consider comparisons in this article.

[47] See also Kibble and Power [92] for a CT-based method for both text planning and choice of referring expressions.

the gallery, even though both the gallery and the exhibit are neuter and can be in excellent condition. Note that with both referents, the transition from (6) to (7) is a CONTINUE; hence, transition type preferences play no role.

The gender of each generated pronoun is the gender of the (most appropriate) NL name of the S that the pronoun realizes.[48] If S does not have an NL name, NaturalOWL uses the gender of the (most appropriate) NL name of the most specific class that includes S and has an NL name (or one of these classes, if they are many). If `exhibit7` in our previous example does not have an NL name, i.e., "exhibit 7" is a tokenized identifier or `rdfs:label`, if the most specific class of `exhibit7` is `Exhibit`, and if `Exhibit` has only one NL name in the target language and its gender is neuter, NaturalOWL uses a neuter pronoun, as in the example above. NL names can also be associated with *sets* of genders, which give rise to pseudo-pronouns like "he/she"; this may be desirable, for example, in the NL name of a class like `Person`.

With some individuals or classes, we may not wish to use NL names, nor tokenized identifiers nor `rdfs:label` strings. This is common, for example, in museum ontologies, where some exhibits are known by particular names (e.g., a classical statue known as the "Doryphoros"), but many other exhibits are anonymous and their OWL identifiers (e.g., `exhibit317`) are not particularly meaningful to end-users. NaturalOWL allows the domain author to mark individuals and classes as *anonymous*, to indicate that their NL names, tokenized identifiers, and `rdfs:label` strings should be avoided. When the primary target (the individual or class the text is generated for) is marked as anonymous, NaturalOWL uses a demonstrative noun phrase (e.g., "this statue") to refer to it. The demonstrative phrase involves the NL name of the most specific class that subsumes the primary target, that has an NL name, and has not been marked as anonymous (or one of these classes, if they are many). Especially in sentences that express `isA` or `instanceOf` message triples about the primary target, the demonstrative noun phrase is simply "this", to avoid generating sentences like "This statue is a statue". Assuming, for example, that `exhibit7` has been marked as anonymous, that its most specific super-class that has not been marked as anonymous is `Statue`, and that `Statue` has the NL name "statue", the text of our previous example becomes as follows.[49]

(1) *This* is a statue. (2) *It* was sculpted by Nikolaou. (3) *Nikolaou* was born in Athens. (4) *He* was born in 1918. (5) *He* died in 1998. • (6) *This statue* is now in the National Gallery. (7) *It* is in excellent condition.

The marking of anonymous individuals and classes currently applies only to the primary target. For example, if the primary target is Nikolaou, then the generated text may still contain sentences like "Nikolaou sculpted exhibit 7 and exhibit 9", even if `exhibit7` and `exhibit9` have been marked as anonymous. Although a sentence like "Nikolaou sculpted two statues" could easily be produced instead, complications arise if, for example, there is information to be conveyed for both statues, in which case we would need to produce appropriate referring expressions (e.g., "the first one", "the second statue"). We leave such improvements for future work, along with work to produce more varied referring expressions for the Os of message triples (e.g., reflexives in sentences like "He painted *himself*").[50]

2.3. *Surface realization*

In many NLG systems, the sentences at the end of micro-planning are underspecified; for example, the exact order of their syntactic constituents or the exact forms of their words (e.g., gender, number, case) may still be unspecified. Generic surface realizers based on large-scale grammars or statistical models can then be used to fill in the missing information during surface realization, as already discussed (Section 2.2.1).[51] By contrast, in NaturalOWL (and most template-based NLG systems) the sentence plans at the end of micro-planning already completely specify the surface (final) form of each sentence; recall that referring expressions will have

[48] Note that in languages like Greek, which use grammatical instead of natural genders, the pronouns' genders cannot be determined by consulting the domain ontology (e.g., to check if the referent is animate or inanimate). For example, the Greek NL names (nouns) for "computer" ("υπολογιστής"), "screen" ("οθόνη"), and "keyboard" ("πληκτρολόγιο") are masculine, feminine, and neuter, respectively, and pronouns must be in the same gender as the corresponding NL names.

[49] If a sentence about a *parent* class of the primary target has been generated and the next sentence uses a demonstrative noun phrase to refer to the primary target, then the demonstrative includes "particular" to signal more clearly that the discussion has shifted back to the primary target; see, for example, the generated text of page 18.

[50] The TUNA [62] and GREC [18] challenges and their datasets could be particularly useful in future improvements.

[51] ILEX and M-PIRO used Systemic Grammars [71,15].

already been generated, and aggregation will have been performed. The order of the sentences will have also been specified during text planning. Hence, NaturalOWL's surface realization is mostly a process of converting internal, but fully specified and ordered sentence specifications to the final form of the text. Punctuation is also added, the initial letter of each sentence is capitalized etc.

Application-specific markup (e.g., HTML tags, hyperlinks) or images (e.g., of individuals being described) can also be added by modifying NaturalOWL's surface realization code. In project INDIGO, for example, where NaturalOWL was embedded in robots acting as a museum guides, surface realization was modified to include XML syntactic and semantic markup.[52] There were, for instance, tags that marked sentences with high interest scores, leading the robot to use an emphasis voice and an appropriate facial expression. The sentences were also marked up with the message triples they expressed, to help a dialogue manager keep track of the information conveyed to each end-user.

3. Other related work

We have already provided pointers to relevant work that could be used to enhance NaturalOWL's processing stages. We now turn to controlled natural languages and ontology verbalizers for OWL, and subsequently to other NLG work for OWL ontologies.

3.1. *Controlled languages and verbalizers for OWL*

As already noted, we have been using OWL's functional-style syntax, but several semantically equivalent OWL syntaxes have been proposed. There has also been work to develop controlled natural languages (CNLs), mostly English-like, to be used as alternative OWL syntaxes. For example, Sydney OWL Syntax (SOS) [36] is an English-like CNL with a bidirectional mapping to and from OWL's functional-style syntax; SOS is based on experience from PENG [150,154]. A similar bidirectional mapping has been defined for Attempto Controlled English (ACE) [58,84,83], a CNL originally developed for software specifications. Rabbit [74,46] and CLOnE [59] are other OWL CNLs, mostly intended to be used by domain experts during ontology authoring [47], i.e., when creating OWL ontologies. Some OWL CNLs are only partial OWL syntaxes, meaning that they cannot express all the statements of other OWL syntaxes. Consult Schwitter *et al.* [153] for related discussion and a comparison of SOS, ACE, and Rabbit. Schwitter [151] provides a broader introduction to CNLs for knowledge representation.

Much work on OWL CNLs focuses on ontology authoring and ontology querying [21,22,19,103,152,90]; in both cases, the emphasis is mostly on the direction from CNL to OWL or formal query languages.[53] In Rabbit, for example, the main goal seems to be to translate from CNL to a normative OWL syntax, not backwards. More relevant to this article are CNLs like SOS and ACE, to which automatic mappings from normative OWL syntaxes are available.

By feeding an OWL ontology expressed, for example, in functional-style syntax to a mapping that translates to an English-like CNL, all the axioms of the ontology can be turned into English-like sentences. Systems of this kind are often called *ontology verbalizers*, a term we introduced briefly in Section 1. This term, however, also includes systems that translate from OWL to English-like statements that do not necessarily belong in an explicitly defined CNL [78,69,149,141,137,157,158,109,110].

Although ontology verbalizers can be viewed as performing a kind of light NLG, they are different from NaturalOWL and other similar, more complex NLG systems. As already discussed, verbalizers typically translate axioms one by one, without considering the coherence (or topical cohesion) of the resulting texts, usually without aggregating sentences nor generating referring expressions, and often by producing sentences that are not entirely fluent or natural; for example, ACE and SOS occasionally use variables instead of noun phrases and referring expressions [153]. Also, verbalizers typically do not employ any user modeling mechanisms; for example, they do not attempt to model the interests and linguistic preferences (e.g., vocabulary, length of texts) of dif-

[52] INDIGO was an FP6 IST project of the European Union; consult `http://www.ics.forth.gr/indigo/`. NaturalOWL's Protégé plug-in can optionally display the generated texts with INDIGO's markup. In previous demonstrators, the generated texts included RDFa markup; see `http://www.w3.org/TR/xhtml-rdfa-primer/`. In a similar manner, SCHEMA.ORG markup could also be included by generating texts from SCHEMA.ORG ontologies represented in OWL; consult `http://www.schema.org/`.

[53] The WYSIWYM approach or *conceptual authoring* [139,70], which has recently been applied to OWL ontologies [140,136], and *round-trip authoring* [161,59,44] are bidirectional, but again they focus mostly on ontology authoring and querying.

ferent types of readers, nor the information that has already been conveyed to them (e.g., to avoid repetitions). Expressing the exact meaning of the ontology's axioms in an unambiguous manner is considered more important in verbalizers than composing a fluent, coherent, and interesting text, partly because the verbalizers' texts are typically intended to be read by domain experts during ontology authoring. Furthermore, verbalizers treat the domain ontology as the only source of domain-dependent information, without employing additional domain-dependent generation resources.

3.2. *NLG systems for OWL ontologies*

We now consider more complex, compared to verbalizers, NLG systems for OWL ontologies. We note that the distinction between the two types of systems is not always clear. For example, some verbalizers include basic sentence aggregation [177] and text planning [109], and some of the systems we discuss below attempt to extract domain-dependent linguistic resources from the domain ontology. Schutte's system [149] is particularly difficult to classify, as it also generates pronouns and invokes KPML [15], otherwise being closer to verbalizers.

3.2.1. *Ontosum*

ONTOSUM [25] generates natural language descriptions of individuals, but apparently not classes, from RDF and OWL ontologies. It is an extension of MIAKT [28,24], which was used to generate medical reports. Both were implemented in GATE [27]. They provide graphical user interfaces to manipulate domain-dependent generation resources, and the domain ontologies can be edited with a version of Protégé embedded in GATE [26].

ONTOSUM adopts a pipeline architecture, similar to NaturalOWL's. Content selection identifies the facts to express, excluding facts that have been mentioned. ONTOSUM's user modeling mechanisms appear to be more limited than NaturalOWL's; for example, there do not seem to be mechanisms to specify the interest of the domain ontology's facts.

In text planning, ONTOSUM uses text schemata (Section 2.1.2). For sentence planning, it provides built-in sentence plans for four basic properties (e.g., `active-action`, `part-whole`). Every other property is made (manually or via heuristics that examine the properties' identifiers) a sub-property of a basic one, which allows it to inherit a sentence plan from a basic property. ONTOSUM can also express properties by using their `rdfs:label` strings, strings obtained by tokenizing the properties' identifiers, or strings provided by the domain author; the effect is similar, as when using NaturalOWL's default sentence plan.

Unlike NaturalOWL, ONTOSUM's aggregation operates only on logical facts, before micro-planning, and it aggregates only facts involving the same semantic subject and property. Like NaturalOWL, ONTOSUM maps classes and individuals to NL names, specified in a domain-dependent lexicon. ONTOSUM's NL names, however, are simpler; for example, they do not mark head words, prepositions, etc., information that is used in some of NaturalOWL's aggregation rules. There are also mechanisms in ONTOSUM to extract NL names from `rdfs:label` strings or identifiers, when lexicon entries are unavailable.

No detailed description of ONTOSUM, at least not as detailed as this article, appears to have been published, and the system does not seem to be publicly available, unlike NaturalOWL. Furthermore, no trials of ONTOSUM with independently created domain ontologies seem to have been published.

3.2.2. *Other NLG systems for OWL and RDF*

Mellish and Sun [125,159,160] focus mostly on lexicalization and aggregation, aiming to produce a single aggregated sentence from an input collection of RDF triples; by contrast, NaturalOWL produces multi-sentence texts. In complementary work, Mellish *et al.* [124,121,122] also consider content selection in the generation of natural language descriptions of OWL classes. Unlike NaturalOWL, their system does not select only facts that are explicitly mentioned in the domain ontology, but it also constructs facts that can be inferred from the domain ontology by performing reasoning in description logic. It would be particularly interesting to examine how reasoning mechanisms of this kind could be added to NaturalOWL's content selection.

Wilcock's NLG engine [174,175,176] has been used to generate texts from RDF and DAML+OIL ontologies, DAML+OIL being a predecessor of OWL.[54] We are not aware, however, of any attempts to use that engine with OWL ontologies.

[54] See `http://www.w3.org/Submission/2001/12/`.

4. Trials

In previous work, NaturalOWL was used mostly to describe cultural heritage objects. In project XENIOS, for example, it was tested with an OWL version of a domain ontology that was created during M-PIRO to document approximately 50 archaeological exhibits (e.g., statues, coins, vases) [4]. The OWL version comprised 76 classes, 343 individuals (including cities, persons, historical periods, etc.), and 41 properties. In XENIOS, NaturalOWL was also embedded in a robotic avatar that presented M-PIRO's exhibits in a virtual world museum [131], and in a real-world mobile robot that provided information about a cultural heritage centre and its exhibitions [169].[55] More recently, in project INDIGO [96,97], NaturalOWL was embedded in mobile robots acting as tour guides in an exhibition about the ancient Agora of Athens. The robots escorted visitors to wall-mounted screens showing present day photos and digital reconstructions of the Agora's monuments.[56] An OWL domain ontology documenting 43 monuments was used; there were 49 classes, 494 individuals, and 56 properties in total.

In the projects mentioned above, NaturalOWL's texts were eventually almost indistinguishable from human-authored texts. We participated, however, along with curators and other colleagues, in the development of the domain ontologies, and we may have biased the domain ontologies towards choices (e.g., classes and properties) that made it easier for NaturalOWL to generate high-quality texts. Hence, in the trials discussed below, we wanted to experiment with independently developed OWL domain ontologies. We also wanted to experiment with different domains, as opposed to cultural heritage.

A further goal was to compare NaturalOWL's texts against those of a simpler verbalizer. We used the OWL verbalizer of the SWAT project [158,178], which we found to be particularly robust and useful, as it provides a quick English-like view of an OWL domain ontology with no domain authoring effort.[57] The verbalizer produces, among other outputs, an alphabetical glossary with an entry for each named class, property, and individual. Each glossary entry is a sequence of English-like sentences expressing the OWL statements of the ontology that involve the particular class, property, or individual.

The SWAT verbalizer uses a predetermined partial order of statements in each glossary entry; for example, when describing a class, statements about equivalent classes or super-classes of the target class are mentioned first, and individuals belonging in the target class are mentioned last.[58] The verbalizer actually translates the OWL ontology to Prolog, it extracts lexicon entries from OWL identifiers and `rdfs:label` strings, and it uses predetermined sentence plans specified as a DCG grammar. It also aggregates, in effect, message triples of the same property that share one argument (S or O) [177].

Our hypothesis was that the domain-dependent generation resources would help NaturalOWL produce texts that end-users would consider more fluent, coherent, and interesting compared to those produced by the SWAT verbalizer, but also those produced by NaturalOWL without domain-dependent generation resources. We also wanted to measure the effort that is required to create NaturalOWL's domain-dependent generation resources for existing domain ontologies. This was not measured in our previous work, because the development of the domain-dependent generation resources was combined with the development of the domain ontologies. Since the time needed to create the domain-dependent generation resources depends on one's familiarity with NaturalOWL and its Protégé plug-in, exact times are not particularly informative. Instead, we report figures such as the number of sentence plans, lexicon entries, user modeling annotations etc. that were required, along with approximate times. We do not evaluate the usability of NaturalOWL's Protégé plug-in, since it is very

[55] XENIOS was co-funded by the European Union and the Greek General Secretariat of Research and Technology; see `http://www.ics.forth.gr/xenios/`.

[56] Videos of the robotic guides of XENIOS and INDIGO in action are available at `http://nlp.cs.aueb.gr/projects.html`. Two AUEB students, George Karakatsiotis and Vangos Pterneas won the Interoperability Challenge of Microsoft's Imagine Cup 2011 with a similar mobile phone application, called Touring Machine, that uses NaturalOWL; see `http://www.youtube.com/watch?v=PaNAmNC7dZw`. In Touring Machine and INDIGO, the domain-dependent generation resources were also used to answer follow-up questions (e.g., about persons mentioned in descriptions of monuments).

[57] The SWAT verbalizer can be used on the Web at `http://swat.open.ac.uk/tools/`. We used the version that was on-line in July and August 2011. A similar verbalizer from OWL to ACE (Section 3.1) is available at `http://attempto.ifi.uzh.ch/site/docs/owl_to_ace.html`.

[58] The verbalizer also organizes the English-like sentences of each glossary entry under sub-headings like 'Definition', 'Taxonomy', 'Description', 'Distinctions' [178]. We discarded these subheadings, whose meanings were not entirely clear to us, but we retained the order of the sentences.

similar to M-PIRO's authoring tool. Previous usability experiments [4] showed that computer science graduates with no expertise in NLG could learn to use effectively M-PIRO's authoring tool to create the necessary domain-dependent generation resources for existing or new domain ontologies, after receiving the equivalent of a full-day introduction course.

4.1. *Trials with the Wine Ontology*

In the first trial, we experimented with the Wine Ontology, which is often used in Semantic Web tutorials.[59] It comprises 63 wine classes (and subclasses), 52 wine individuals, a total of 238 classes and individuals (including kinds of grapes, wineries, regions, etc.), and 14 properties.

We submitted the Wine Ontology to the SWAT verbalizer to obtain its glossary of English-like descriptions of classes, properties, and individuals. We retained only the descriptions of the 63 wine classes and the 52 wine individuals. Subsequently, we also discarded 20 of the 63 wine class descriptions, as they were for trivial classes (e.g., `RedWine`) and they were stating the obvious (e.g., "A red wine is defined as a wine that has as color Red"). In the descriptions of the remaining 43 wine classes and 52 wine individuals, we discarded sentences expressing axioms that NaturalOWL does not consider, for example sentences providing examples of individuals that belong in a class being described, or sentences that in effect express message triples of the form $\langle S, P, O\rangle$ when the target is O (not S). The remaining texts express the same OWL statements that NaturalOWL expresses when its maximum fact distance is set to one and no user modeling anotations are provided.[60] Two examples of texts obtained by using the SWAT verbalizer follow.

> Chenin Blanc (class): A chenin blanc is defined as something that is a wine, is made from grape the Chenin Blanc Grape, and is made from grape at most one thing. A chenin blanc both has as flavor Moderate, and has as color White. A chenin blanc both has as sugar only Off Dry and Dry, and has as body only Full and Medium.

> The Foxen Chenin Blanc (individual): The Foxen Chenin Blanc is a chenin blanc. The Foxen Chenin Blanc has as body Full. The Foxen Chenin Blanc has as flavor Moderate. The Foxen Chenin Blanc has as maker Foxen. The Foxen Chenin Blanc has as sugar Dry. The Foxen Chenin Blanc is located in the Santa Barbara Region.

We then generated texts for the 43 classes and 52 individuals using NaturalOWL without domain dependent generation resources, setting the maximum fact distance to one; example texts follow.

> Chenin Blanc (class): Chenin Blanc is wine. Chenin Blanc has body only Full or Medium. Chenin Blanc has sugar only Off Dry or Dry. Chenin Blanc has color White. Chenin Blanc has flavor Moderate. Chenin Blanc made from grape exactly one Wine Grape: Chenin Blanc Grape.

> The Foxen Chenin Blanc (individual): Foxen Chenin Blanc is Chenin Blanc. Foxen Chenin Blanc has sugar Dry. Foxen Chenin Blanc has maker Foxen. Foxen Chenin Blanc has body Full. Foxen Chenin Blanc located in Santa Barbara Region. Foxen Chenin Blanc has flavor Moderate.

Subsequently, we constructed NaturalOWL's domain-dependent generation resources for the Wine Ontology. The resources are summarized in Table 10. They were constructed by the second author, who devoted approximately three days to their construction, testing, and refinement. Only English texts were generated in this trial; hence, no resources for Greek were constructed. We defined only one user type, and we added user modeling annotations only to block sentences stating the obvious, by assigning zero interest scores to the corresponding message triples; we also set the maximum messages per (aggregated) sentence to 3. Only 7 of the 14 properties of the Wine Ontology are used in the OWL statements that describe the 43 classes and 52 individuals. We defined 5 sentence plans, which could be used to express the 7 properties; some properties could be expressed by the same sentence plan (e.g., involving the verb "to be"), which is why there are only 5 sentence plans. We did not define multiple sentence plans per property. We also assigned the 7 properties to 2 sections, and we ordered the sections and their properties. We created NL names only when the automatically extracted ones were causing disfluencies. The automatically extracted names were obtained from the OWL identifiers of classes and individuals; no `rdfs:label` strings were available. To reduce the number of NL names further, we declared the 52 individual wines to be anonymous (and provided no NL names for them), which caused NaturalOWL to

[59] See `http://www.w3.org/TR/owl-guide/wine.rdf`.

[60] In the verbalizer's texts, we also discarded sentences expressing `DifferentIndividuals` statements, as they tended to be particularly long and boring; these statements were also discarded when generating NaturalOWL's texts.

resources	English	Greek
sections	2	
property assignments to sections	7	
interest score assignments	8	
sentence plans	5	–
lexicon entries	67	–
natural language names	41	–

Table 10
Domain-dependent generation resources created for the **Wine Ontology** (one user type).

use "this wine" in the first sentence of each wine's description, instead of using the wine's name. Most of the 67 lexicon entries were used in the remaining 41 NL names of classes and individuals (including regions, wineries, kinds of grapes, etc.) that we had to construct. Hence, most of the authoring effort was devoted to NL names (see Table 10), despite the actions we took to reduce their number; we investigate this issue further in the next trial. We note, however, that most of the NL names were very simple, having only 2 slots on average.

We used NaturalOWL with the domain-dependent resources to re-generate the 95 texts for the 43 classes and 52 individuals, again setting the maximum fact distance to one; example texts follow.

Chenin Blanc (class): A Chenin Blanc is a moderate, white wine. It has only a full or medium body. It is only off-dry or dry. It is made from exactly one wine grape variety: Chenin Blanc grapes.

The Foxen Chenin Blanc (individual): This wine is a moderate, dry Chenin Blanc. It has a full body. It is made by Foxen in the Santa Barbara County.

The resulting 285 texts (95 × 3) of the three systems (SWAT verbalizer, NaturalOWL with and without domain-dependent generation resources) were shown to 10 computer science students (both undergraduates and graduate students), who were not involved in the development of NaturalOWL; they were all fluent in English, though not native English speakers, and they did not consider themselves wine experts. The students were told that a glossary of wines was being developed for people who were interested in wines and knew basic wine terms (e.g., wine colors, wine flavors), but who were otherwise not wine experts. Each one of the 285 texts was given to exactly one student. Each student was given approximately 30 texts, approximately 10 randomly selected texts from each system. The OWL statements that the texts were generated from were not shown, and the students did not know which system had generated each text. Each student was shown all of his/her texts in random order, regardless of the system that had generated them. The students were asked to score the texts for *sentence fluency*, *referring expressions*, *text structure*, *clarity*, and *interest*, by stating for each text how strongly they agreed or disagreed with statements S_1–S_5 below. A scale from 1 to 3 was used (1: disagreement, 2: ambivalent, 3: agreement).

(S_1) *Sentence fluency*: The sentences of the text are fluent, i.e., each sentence *on its own* is grammatical and sounds natural. When two or more smaller sentences are merged to form a single, longer sentence, the resulting longer sentence is also grammatical and sounds natural.

(S_2) *Referring expressions*: The use of pronouns (e.g., "it", "his") and other referring expressions (e.g., "this wine") in the text is appropriate. The choices of referring expressions (e.g., when to use a pronoun or other expression instead of the name of an object) sound natural, and it is easy to understand what these expressions stand for (e.g., which object a pronoun refers to).

(S_3) *Text structure*: The order of the sentences is appropriate. The text presents information by moving reasonably from one topic to another.

(S_4) *Clarity*: The text is easy to understand, provided that the reader is familiar with basic wine terms (e.g., wine colors, wine flavors).

(S_5) *Interest*: People who are interested in wines, but who are not wine experts, would find the information of the text interesting. Furthermore, there are no redundant sentences in the text (e.g., sentences stating the obvious). [61]

We had to avoid NLG terminology in statements S_1–S_5 as much as possible, to ensure that the students would be able to understand them. S_5 assesses *content selection*, the first processing stage; we expected the differences across the three systems to be very small, as they all reported the same information, with the exception of redundant sentences blocked by using zero interest assignments in NaturalOWL. S_3 assesses *text planning*, the second processing stage; again we expected small differences, as many of the wines' properties can be mentioned in any order, though there are some properties (e.g.,

[61] The students were instructed not to consider whether or not *additional* information should have been included.

maker, location) that are most naturally reported separately from others (e.g., color, flavor, taste), which is why we used 2 sections in the domain-dependent generation resources. S_1 assesses *lexicalization* and *aggregation*; we decided not to use separate statements for these two processing stages, since it might have been difficult for the students to understand exactly when aggregation takes place. S_2 assesses *referring expression* generation. S_4 does not correspond to a particular processing stage; it measures the overall perceived clarity of the texts. There was no statement for *surface realization*, as this stage (as used in all three systems) had a rather trivial effect on the texts.

Table 11 shows the average scores of the three systems, with averages computed on the 95 texts of each system, along with 95% confidence intervals (of sample means). [62] As expected, the domain-dependent generation resources clearly help NaturalOWL produce more fluent sentences and much better referring expressions, despite NaturalOWL's simplistic referring expression generation techniques. The text structure scores show that the assignment of the domain ontology's properties to sections and the ordering of the sections and properties had a greater (positive) impact on the perceived structure of the texts than we expected, also indicating how important the corresponding resources are. The highest score of the SWAT verbalizer was obtained in the clarity criterion, which agrees with our experience that one can usually understand what the texts of the SWAT verbalizer mean, even if their sentences are often not entirely fluent, not particularly well ordered, and keep repeating proper names. With domain-dependent resources, NaturalOWL had the highest clarity score, but the difference from the SWAT verbalizer, which had the second highest score, is not statistically significant. [63] With domain-dependent generation resources, NaturalOWL also obtained higher interest scores than the other two systems, with statistically significant differences from both; these differences, which are larger than we expected, can only be attributed to the zero interest score assignments of the domain-dependent generation resources, which are used to block sentences stating the obvious, because otherwise all three systems report the same information. The SWAT verbalizer obtained higher scores than NaturalOWL *without* domain-dependent generation resources, with the text structure score being the only exception. Only the difference in the referring expression scores of the two systems, though, is statistically significant. Both systems, however, received particularly low scores for their referring expressions, which is not surprising, given that they both refer to individuals and classes by always using automatically extracted names; the slightly higher score of the SWAT verbalizer is probably due to better tokenization of the automatically extracted names.

criteria	SWAT	NaturalOWL −	NaturalOWL +
sentence fluency	2.00 ± 0.15	1.76 ± 0.15	**2.80 ± 0.10**
ref. expressions	1.40 ± 0.13	1.15 ± 0.09	**2.72 ± 0.13**
text structure	2.15 ± 0.16	2.20 ± 0.16	**2.94 ± 0.05**
clarity	2.66 ± 0.13	2.55 ± 0.13	**2.74** ± 0.11
interest	2.30 ± 0.15	2.14 ± 0.16	**2.68 ± 0.12**

Table 11
Average results, with 95% confidence intervals, for **English** texts generated from the **Wine Ontology** by the SWAT verbalizer and NaturalOWL with (+) and without (−) domain-dependent generation resources. For each criterion, the best scores are shown in bold; the confidence interval of the best score is also shown in bold if it does not overlap with the confidence intervals of the other scores of the criterion.

4.2. *Trials with the Consumer Electronics Ontology*

In the second trial, we experimented with the Consumer Electronics Ontology, an OWL ontology for consumer electronics products and services. [64] The ontology comprises 54 classes and 441 individuals (e.g., printer types, paper sizes, well-known manufacturers), but no information about particular products. We added 60 individuals describing 20 digital cameras, 20 camcorders, and 20 printers. The 60 individuals were randomly selected from a publicly available dataset of 286 digital cameras, 613

[62] In a pilot study, we also measured the inter-annotator agreement of two of the students on a sample of 30 texts (10 from each system). The agreement was very high (sample Pearson correlation $r \geq 0.91$) in all five criteria. A similar pilot study was also performed in the next trial, again indicating very high inter-annotator agreement.

[63] When the confidence intervals of two systems do not overlap, the difference is statistically significant. When the intervals overlap, the difference may still be statistically significant; we performed paired two-tailed t-tests ($\alpha = 0.05$) in these cases to check for statistical significance.

[64] Consult `http://www.ebusiness-unibw.org/ontologies/consumerelectronics/v1`. The Consumer Electronics Ontology uses some concepts of the Good Relations Ontology; see `http://www.heppnetz.de/projects/goodrelations/`.

camcorders, and 58 printers, whose instances comply with the Consumer Electronics Ontology.[65]

We submitted the Consumer Electronics Ontology with the additional 60 individuals to the SWAT verbalizer, and we retained only the descriptions of the 60 individuals. As in the previous trial, we removed sentences expressing axioms that NaturalOWL does not consider. We also shortened some sentences to remove parts expressing very general-level information (e.g., "Olympus E-520 is a product or service model, and a digital camera" became "Olympus E-520 is a digital camera"). Lastly, we renamed the string values of some datatype properties to make the texts easier to understand (e.g., the measurement unit "CMT" became "cm"). An example of a resulting description follows. The OWL identifiers of a few individuals are in German, which is why the sentence about the energy source mentions "Akku", which is "battery" in German.

The Sony Cyber-shot DSC-T90 is a digital camera.
The Sony Cyber-shot DSC-T90 has as manufacturer Sony.
The Sony Cyber-shot DSC-T90 has as data interface type Usb2 0.
The Sony Cyber-shot DSC-T90 has as depth Depth.
  Depth has as unit of measurement cm.
  Depth has as value float 9.4.
The Sony Cyber-shot DSC-T90 has as digital zoom factor the Digital Zoom Factor.
  The Digital Zoom Factor has as value float 12.1.
The Sony Cyber-shot DSC-T90 has as display size Display.
  Display has as unit of measurement in.
  Display has as value float 3.0.
The Sony Cyber-shot DSC-T90 has as energy source Akku.
The Sony Cyber-shot DSC-T90 has as feature Video Recording, Microphone and the Automatic Picture Stabilizer.
The Sony Cyber-shot DSC-T90 has as focus size Focus Size.
  Focus Size has as max value float 140.0.
  Focus Size has as min value float 35.0.
The Sony Cyber-shot DSC-T90 has as height Height.
  Height has as unit of measurement cm.
  Height has as value float 5.7.
The Sony Cyber-shot DSC-T90 has as internal memory capacity the Internal Memory Capacity.
  The Internal Memory Capacity has as unit of measurement GB.
  The Internal Memory Capacity has as value float 11.0.
The Sony Cyber-shot DSC-T90 has as optical zoom factor Optical Zoom.
  Optical Zoom has as value float 4.0.
The Sony Cyber-shot DSC-T90 has as shutter lag Shutter Lag.
  Shutter Lag has as max value float 0.0010.
  Shutter Lag has as min value float 2.0.
  Shutter Lag has as unit of measurement sec.
The Sony Cyber-shot DSC-T90 has as weight Weight.
  Weight has as unit of measurement grm.
  Weight has as value float 128.0.
The Sony Cyber-shot DSC-T90 has as width Width.
  Width has as unit of measurement cm.
  Width has as value float 1.5.
The Sony Cyber-shot DSC-T90 has as compact flash false.
The Sony Cyber-shot DSC-T90 has as dimension 9.4 x 1.5 x 5.7.
The Sony Cyber-shot DSC-T90 has as tv connector true.
The Sony Cyber-shot DSC-T90 has as self timer true.

In this domain ontology, many properties have composite values, which are expressed by using auxiliary individuals. In the example above, a property (`hasDepth`) connects the digital camera being described to an auxiliary individual `Depth` (similar to the anonymous node `_:n` of the property concatenation price example of page 24), which is then connected via two other properties (`hasValueFloat` and `hasUnitOfMeasurement`) to the float value 9.4 and the unit of measurement (centimeters), respectively. To improve readability, we obtained the descriptions of the auxiliary individuals (e.g., `Depth`), which are entirely different entries in the glossary that the SWAT verbalizer produces, and we copied them immediately after the corresponding sentences that introduce the auxiliary individuals.[66] We also formatted each text as a list of sentences, with the copied sentences shown indented, again to improve readability.

We then generated texts for the 60 products by using NaturalOWL without domain-dependent generation resources, setting the maximum fact distance to one. Descriptions of auxiliary individuals were also generated, and they were copied immediately after the sentences introducing the auxiliary individuals, as with the texts of the SWAT verbalizer. The texts were again formatted as lists of sentences, with the sentences about the auxiliary individuals shown indented. An example of a resulting text follows.

[65] See `http://rdf4ecommerce.esolda.com/` for the dataset that we used. A list of similar datasets is available at `http://wiki.goodrelations-vocabulary.org/Datasets`.

[66] We also discarded sentences stating the datatypes of auxiliary individuals (e.g., "Depth is a quantitative value float").

Sony Cyber-shot DSC-T90 is Digital Camera.
Sony Cyber-shot DSC-T90 has manufacturer Sony.
Sony Cyber-shot DSC-T90 has feature Automatic Picture Stabilizer, Microphone and Video Recording.
Sony Cyber-shot DSC-T90 has weight Weight.
  Weight has unit of measurement grm.
  Weight has value float 128.0.
Sony Cyber-shot DSC-T90 has energy source Akku.
Sony Cyber-shot DSC-T90 has display size Display.
  Display has unit of measurement in.
  Display has value float 3.0.
Sony Cyber-shot DSC-T90 has optical zoom factor Optical Zoom.
  Optical Zoom has value float 4.0.
Sony Cyber-shot DSC-T90 has width Width.
  Width has unit of measurement cm.
  Width has value float 1.5.
Sony Cyber-shot DSC-T90 has shutter lag Shutter Lag.
  Shutter Lag has min value float 2.0.
  Shutter Lag has max value float 0.0010.
  Shutter Lag has unit of measurement sec.
Sony Cyber-shot DSC-T90 has height Height.
  Height has unit of measurement cm.
  Height has value float 5.7.
Sony Cyber-shot DSC-T90 has internal memory capacity Internal Memory Capacity.
  Internal Memory Capacity has unit of measurement GB.
  Internal Memory Capacity has value float 11.0.
Sony Cyber-shot DSC-T90 has focus size Focus Size.
  Focus Size has min value float 35.0.
  Focus Size has max value float 140.0.
Sony Cyber-shot DSC-T90 has digital zoom factor Digital Zoom Factor.
  Digital Zoom Factor has value float 12.1.
Sony Cyber-shot DSC-T90 has data interface type USB 2.0.
Sony Cyber-shot DSC-T90 has depth Depth.
  Depth has unit of measurement cm.
  Depth has value float 9.4.
Sony Cyber-shot DSC-T90 has self timer true.

In this trial, we also wanted to consider a scenario where there are many individuals to be described (e.g., products sold by a reseller), the set of individuals to be described changes frequently (e.g., new products arrive) along with other individuals (e.g., a new manufacturer may be added), but nothing else in the domain ontology changes (e.g., the definitions of the classes, the available properties); in other words, only the assertional knowledge (ABox) changes. In this case, it may be impractical to update the domain-dependent generation resources whenever the individuals of the domain ontology change. Such an update might, for example, add particular user modeling annotations about the new individuals (e.g., to avoid saying that "The Sony Cyber-shot DSC-T90 is manufactured by Sony"), it might fine-tune the sentence plans to express more fluently information about the new individuals, but most importantly it could provide NL names for the new individuals; recall that the construction of NL names may be a significant part of the authoring effort.

Our hypothesis was that by considering a sample of individuals of the types to be described (a sample of cameras, camcorders, and printers in our case), it would be possible to construct domain-dependent generation resources (e.g., sections, ordering of sections and properties, sentence plans, NL names of classes) that would help NaturalOWL generate reasonably good descriptions of new individuals (products), without updating the domain-dependent generation resources, and by using the tokenized OWL identifiers or `rdfs:label` strings of new individuals, for which NL names would be unavailable.

To simulate a scenario of this kind, we randomly split the 60 products in two non-overlapping sets, the *development set* and the *test set*, each consisting of 10 digital cameras, 10 camcorders, and 10 printers. As in the previous trial, the second author constructed and refined NaturalOWL's domain dependent generation resources, this time by considering a version of the domain ontology that included the 30 products of the development set, but not the 30 products of the test set, and the texts generated for the products of the development set; this took approximately six days (for two languages). Texts for the 30 products of the test set were then also generated by using NaturalOWL and the domain dependent generation resources of the development set.

As in the previous trial, we defined only one user type, and we added user modeling annotations only to block sentences stating the obvious. The maximum messages per (aggregated) sentence was again set to three. We constructed domain-dependent generation resources for both English and Greek in this trial; the resources are summarized in Table 12. We created sentence plans only for the 42 properties of the domain ontology that were used in the development set (one sentence plan per property); the test set uses two additional properties, for which NaturalOWL's default sentence plans (for English and Greek) were used. We also assigned the 42 proper-

resources	English	Greek
sections	6	
property assignments to sections	42	
interest score assignments	12	
sentence plans	42	42
lexicon entries	19	19
natural language names	36	36

Table 12
Domain-dependent generation resources created for the **Consmer Electronics Ontology** (one user type).

ties to 5 sections, and we ordered the sections and their properties. We created NL names only when the automatically extracted ones were causing disfluencies in the development texts. Unlike the previous trial, the products to be described were not declared to be anonymous individuals, but the number of NL names that had to be provided was roughly the same as in the previous trial, since fewer automatically extracted names were causing disfluencies; in particular, all product names had reasonably good `rdfs:label` strings providing their English names. The largest part of the authoring effort was now devoted to sentence plans, which was partly due to the larger number of properties of the domain ontology, compared to the previous trial.

An example description of a product from the development set produced by using NaturalOWL with the domain-dependent generation resources follows. We formatted the sentences of each section as a separate paragraph, headed by the name of the section (e.g., "Other features:"); this formatting can be easily achieved, given that sections can be automatically marked up by NaturalOWL in the generated texts. The maximum fact distance was again set to one, but the sentence plans caused NaturalOWL to automatically retrieve additional message triples describing the auxiliary individuals at distance one; hence, we did not have to retrieve this information manually, unlike the indented sentences in the texts of the SWAT verbalizer and NaturalOWL without domain-dependent generation resources. [67]

Type: Sony Cyber-shot DSC-T90 is a digital camera.
Main features: It has a focal length range of 35.0 to 140.0 mm, a shutter lag of 2.0 to 0.0010 sec and an optical zoom factor of 4.0. It has a digital zoom factor of 12.1 and its display has a diagonal of 3.0 in.
Other features: It features an automatic picture stabilizer, a microphone, video recording and it has a self-timer.
Energy and environment: It uses batteries.
Connectivity, compatibility, memory: It supports USB 2.0 connections for data exchange and it has an internal memory of 11.0 GB.
Dimensions and weight: It is 5.7 cm high, 1.5 cm wide and 9.4 cm deep. It weighs 128.0 grm.

[67] See also the relevant footnote on page 24.

An example product description from the test set follows. "Super A 3 B" is an automatically obtained name (of a type of paper) that has been extracted from the corresponding OWL identifier.

Type: Canon PIXMA Pro9500 is a printer.
Main features: Its maximum black and white printing resolution is 4800.0 dots horizontally, 2400.0 dots vertically and it prints on at most Super A 3 B. It holds up to 150.0 sheets in the output tray.
Other features: It features color printing.
Energy and environment: It consumes 20.0 watt during use, 1.8 watt during standby and its sound emission is 34.0 db during use.
Connectivity, compatibility, memory: It does not have network ability, it supports the PictBridge standard, USB 2.0 connections for data exchange, and it is compatible with Mac OS X and Windows.
Dimensions and weight: It is 19.1 cm high, 35.4 cm wide and 66.0 cm deep. It weighs 14000.0 grm.
It has max color print speed normal Max Color Print Speed Normal.

The last sentence of the product description above expresses a message triple involving a property (`hasMaxColorPrintSpeedNormal`) of the domain ontology for which no sentence plan has been provided, because that property was not used in the development set. The default sentence plan has been used, but the O of the $\langle S, P, O \rangle$ message triple reported by the sentence is the auxiliary individual `MaxColorPrintSpeedNormal`. A manually crafted sentence plan would instruct NaturalOWL to use the value of another property of the domain ontology that leads from `MaxColorPrintSpeedNormal` to a numerical value expressing the printing speed (and similarly for the unit of measurement), and use that number (and unit) as the object of the sentence. The default sentence plan, however, instructs NaturalOWL to use the name of `MaxColorPrintSpeedNormal` itself as the object, leading to a nonsensical sentence, where a tokenized form of the OWL identifier `MaxColorPrintSpeedNormal` is used as the object.

Also, the sentence has been placed at the end of the text, because the property it expresses has not been assigned to any section. Problematic sentences like this were very rare in the texts that NaturalOWL generated for the test set using the domain-dependent generation resources of the development set, because the two additional properties of the test set were very rarely used. We also note that in the scenario that we are mostly interested in, only the individuals of the domain ontology would change; no new properties would be added, unlike the two additional properties of this trial's test set.

The 180 English texts that were generated by the three systems for the 30 development and the 30 test products were shown to the same 10 students of the first trial. The students were now told that the texts would be used in on-line descriptions of products in the Web site of a retailer, that the customers would also be able to see information about the products in the form of tables, but that the purpose of the study was to consider only the texts. Again, the OWL statements that the texts were generated from were not shown to the students, and the students did not know which system had generated each text. Each student was shown 18 randomly selected texts, 9 for products of the development set (three texts per system) and 9 for products of the test set (again three texts per system). Each student was shown all of his/her texts in random order, regardless of the system that had generated them. The students were asked to score the texts as in the previous trial.

criteria	SWAT	NaturalOWL −	NaturalOWL +
sentence fluency	1.97 ± 0.15	1.93 ± 0.27	**2.90 ± 0.08**
ref. expressions	1.10 ± 0.06	1.10 ± 0.11	**2.87 ± 0.08**
text structure	1.67 ± 0.15	1.33 ± 0.19	**2.97 ± 0.04**
clarity	1.97 ± 0.15	2.07 ± 0.26	**3.00 ± 0.00**
interest	1.77 ± 0.14	1.73 ± 0.29	**3.00 ± 0.00**

Table 13
Average results, with 95% confidence intervals, for **English** texts generated from **development** data of the **Consumer Electronics Ontology**. The texts were generated by the SWAT verbalizer and NaturalOWL with (+) and without (−) domain-dependent generation resources. For each criterion, the best scores are shown in bold; the confidence interval of the best score is also shown in bold if it does not overlap with the confidence intervals of the other scores of the criterion.

Table 13 shows the results for the English texts of the *development* set; when a confidence interval is shown as 0.00, this means that all the students gave the same score to all the texts of that particular system. As in the previous trial, the domain-dependent generation resources clearly help NaturalOWL produce much more fluent sentences, and much better referring expressions and sentence orderings. The text structure scores of the SWAT verbalizer and NaturalOWL without domain-dependent generation resources are now much lower than in the previous trial, because the texts of these systems jump from one topic to another making the texts look very incoherent (e.g., a sentence that expresses the width of a camera may be separated from a sentence expressing its height by an intervening sentence about its shutter lag), especially in this trial where the texts of the two systems are much longer. This incoherence may have also contributed to the much lower clarity scores of these two systems, compared to the previous trial. The interest scores of the two systems are also much lower than in the previous trial; this may be due to the verbosity of their texts, which is caused by their frequent references to auxiliary individuals in the second trial, combined with the lack (or very little use) of sentence aggregation and pronoun generation. By contrast, the clarity and interest of NaturalOWL with domain-dependent generation resources were judged to be perfect; the poor clarity and interest of the texts of the other two systems may have contributed to these perfect scores though. Again, the SWAT verbalizer obtained slightly better scores than NaturalOWL *without* domain-dependent generation resources, except for clarity, but the differences in the scores of the two systems are not statistically significant in any of the criteria.

Table 14 shows the results for the English texts of the *test* set. The results of the SWAT verbalizer and NaturalOWL without domain-dependent generation resources are very similar to those of Table 13, as one would expect. There was only a very marginal decrease in the scores of NaturalOWL with domain-dependent generation resources, compared to the scores of the same system for the development set (cf. Table 13). There is no statistically significant difference, however, between the corresponding cells of the two tables, for any of the three systems. These results support our hypothesis that by considering a sample of individuals of the types to be described one can construct domain-dependent generation resources that can be used to produce good texts for new individuals of the same types, provided that the rest of the domain ontology remains unchanged. The fact that all product names (but not other individuals) had `rdfs:label` strings providing their English names probably contributed to the high results of NaturalOWL with domain-dependent generation re-

criteria	SWAT	NaturalOWL −	NaturalOWL +
sentence fluency	2.03 ± 0.15	1.87 ± 0.15	**2.87 ± 0.08**
ref. expressions	1.10 ± 0.06	1.10 ± 0.06	**2.87 ± 0.08**
text structure	1.57 ± 0.13	1.37 ± 0.12	**2.93 ± 0.05**
clarity	2.07 ± 0.15	1.93 ± 0.15	**2.97 ± 0.04**
interest	1.83 ± 0.17	1.60 ± 0.14	**2.97 ± 0.04**

Table 14
Average results, with 95% confidence intervals, for **English** texts from **test** data of the **Consumer Electronics Ontology**. The texts were generated by the SWAT verbalizer and NaturalOWL with (+) and without (−) domain-dependent generation resources. For each criterion, the best scores are shown in bold; the confidence interval of the best score is also shown in bold if it does not overlap with the confidence intervals of the other scores of the criterion.

sources in the test set, but `rdfs:label` strings of this kind are common in OWL ontologies.

We then showed the 60 Greek texts that were generated by NaturalOWL with domain-dependent generation resources to the same 10 students, who were native Greek speakers; the SWAT verbalizer and NaturalOWL without domain-dependent generation resources cannot generate Greek texts from the Consumer Electronics ontology. Table 15 shows the results we obtained for the Greek texts of the development and test sets. There is no statistically significant difference from the corresponding results for English (cf. the last columns of Tables 13 and 14). There is also no statistically significant difference in the results for the Greek texts of the development and test sets (Table 15). We note, however, that it is common to use English names of consumer electronics products in Greek texts, which made using the English `rdfs:label` names of the products in Greek texts look acceptable. In other domains, for example about cultural heritage, it would be unacceptable to use English names of monuments, locations etc. in Greek texts; hence, one would have to provide Greek NL names for any new individuals added to the domain ontology.

5. Conclusions and future work

This article has provided a detailed description of NaturalOWL, an open-source NLG system that produces texts (in English and Greek) describing individuals or classes from OWL domain ontologies optionally associated with domain-dependent generation resources. Unlike simpler OWL verbalizers, which typically express a single axiom of the domain ontology at a time in controlled and often not entirely fluent natural language mostly for the benefit of domain experts, NaturalOWL aims to generate fluent, coherent, and interesting multi-sentence texts appropriate for end-users.

criteria	NaturalOWL +, development data	NaturalOWL +, test data
sentence fluency	2.87 ± 0.12	2.83 ± 0.09
ref. expressions	2.77 ± 0.20	2.80 ± 0.11
text structure	3.00 ± 0.00	3.00 ± 0.00
clarity	3.00 ± 0.00	2.93 ± 0.05
interest	2.97 ± 0.06	3.00 ± 0.00

Table 15
Avg. results, with 95% confidence intervals, for **Greek** texts generated from **development** and **test** data of the **Consumer Electronics Ontology** by NaturalOWL with (+) and without (−) domain-dependent generation resources.

We discussed NaturalOWL's processing stages, the optional domain-dependent generation resources of each stage, why the resources are useful, and how they can also be represented in OWL. We also presented trials we performed to measure the effort that is required to configure NaturalOWL for new domain ontologies, and the quality of the resulting texts. The trials confirmed our hypothesis that the domain-dependent generation resources help NaturalOWL produce texts of significantly better quality, compared to the texts produced without the domain-dependent generation resources, and clearly better than the texts of a simpler OWL verbalizer.

To the best of our knowledge, this is the first detailed description of a complete, general-purpose, NLG system for OWL domain ontologies. It is also the first study of the effect of domain-dependent generation resources on the texts of a system that can describe OWL classes and individuals both with and without domain-dependent generation resources. This article has also shown how the domain-dependent generation resources of NaturalOWL can themselves be represented in OWL, which allows easily publishing them on the Web, reusing them, checking them for inconsistencies etc., as with other OWL ontologies. The trials that we performed are also novel, in that they considered existing OWL domain ontologies, created independently by others.

NaturalOWL was partly developed to demonstrate the benefits of using NLG on the Semantic Web. A key benefit is that it becomes possible to publish information on the Web in the form of OWL domain ontologies, and rely on NLG to automatically produce personalized texts in multiple languages from the ontologies, thus making the information easily accessible not only to computer programs and domain experts, but also end-users. We hope that NaturalOWL will contribute towards a wider adoption of NLG on the Semantic Web and also towards a gradual standardization of domain-dependent generation resources, allowing resources to be reused more easily across NLG systems. We welcome cooperations to add to NaturalOWL support for additional languages and more eleborate NLG components. To that end we have suggested several possible extensions, along with pointers to prominent related work, and we provide a collaborative source code repository for NaturalOWL.

We are currently working towards a global optimization NLG approach, in order to replace the pipeline architecture of NaturalOWL with an architecture that will consider all the NLG processing stages in parallel, in order to avoid greedy stage-specific decisions. It would also be particularly interesting to explore how some of the domain-dependent generation resources could be obtained automatically from appropriate corpora.

References

[1] E. Althaus, N. Karamanis, A. Koller, Computing locally coherent discourses, in: 42nd Annual Meeting of ACL, Barcelona, Spain, 2004.

[2] I. Androutsopoulos, S. Kallonis, V. Karkaletsis, Exploiting OWL ontologies in the multilingual generation of object descriptions, in: 10th European Workshop on NLG, Aberdeen, UK, 2005.

[3] I. Androutsopoulos, P. Malakasiotis, A survey of paraphrasing and textual entailment methods, Journal of Artificial Intelligence Research 38 (2010) 135–187.

[4] I. Androutsopoulos, J. Oberlander, V. Karkaletsis, Source authoring for multilingual generation of personalised object descriptions, Nat. Language Engineering 13 (3) (2007) 191–233.

[5] G. Angeli, P. Liang, D. Klein, A simple domain-independent probabilistic approach to generation, in: Conf. on Empirical Methods in Nat. Language Processing, Cambridge, MA, 2010.

[6] G. Antoniou, F. van Harmelen, A Semantic Web primer, 2nd ed., MIT Press, 2008.

[7] C. Areces, A. Koller, K. Striegnitz, Referring expressions as formulas of description logic, in: 5th Int. Nat. Lang. Generation Conf. Salt Fork, OH, 2008.

[8] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, P. Patel-Schneider (eds.), The description logic handbook: theory, implementation and application, Cambridge University Press, 2002.

[9] R. Barzilay, N. Elhadad, K. McKeown, Inferring strategies for sentence ordering in multidocument news summarization, Journal of Artificial Intelligence Research 17 (2002) 35–55.

[10] R. Barzilay, M. Lapata, Collective content selection for concept-to-text generation, in: Human Lang. Technology Conf. and Conf. on Empirical Methods in Nat. Language Processing, Vancouver, British Columbia, Canada, 2005.

[11] R. Barzilay, M. Lapata, Aggregation via set partitioning for natural language generation, in: Human Lang. Technology Conf. of the North American Chapter of ACL, New York, NY, 2006.

[12] R. Barzilay, M. Lapata, Modeling local coherence: An entity-based approach, Comput. Linguistics 34 (1) (2008) 1–34.

[13] R. Barzilay, L. Lee, Catching the drift: Probabilistic content models, with applications to generation and summarization, in: 43rd Annual Meeting of ACL, Ann Arbor, MI, 2004.

[14] J. Bateman, Upper modelling: A general organisation of knowledge for nat. lang. processing, in: 5th Int. Workshop on NLG, Dawson, PA, 1990.

[15] J. Bateman, Enabling technology for multilingual nat. lang. generation: the KPML development environment, Nat. Lang. Engineering 3 (1) (1997) 15–56.

[16] J. Bateman, M. Zock, Natural language generation, in: R. Mitkov (ed.), The Oxford Handbook of Comput. Linguistics, chap. 15, Oxford University Press, 2003, pp. 284–304.

[17] A. Belz, Automatic generation of weather forecast texts using comprehensive probabilistic generation-space models, Nat. Lang. Eng. 14 (4) (2008) 431–455.

[18] A. Belz, E. Kow, J. Viethen, A. Gatt, Generating referring expressions in context: the GREC task evaluation challenges, in: E. Krahmer, M. Theune (eds.), Empirical Methods in Nat. Lang. Generation, Springer-Verlag, 2010, pp. 294–327.

[19] R. Bernardi, D. Calvanese, C. Thorne, Lite natural language, in: 7th Int. Workshop on Comput. Semantics, Tilburg, The Netherlands, 2007.

[20] T. Berners-Lee, J. Hendler, O. Lassila, The Semantic Web, Scientific American (2001) 34–43.

[21] A. Bernstein, E. Kaufmann, GINO – a guided input natural language ontology editor, in: 5th Int. Semantic Web Conf. Athens, GA, 2006.

[22] A. Bernstein, E. Kaufmann, C. Kaiser, C. Kiefer, Ginseng: A guided input natural language search engine for querying ontologies, in: Jena User Conf. Bristol, UK, 2006.

[23] D. Bilidas, M. Theologou, V. Karkaletsis, Enriching OWL ontologies with linguistic and user-related annotations: the ELEON system, in: 19th IEEE Int. Conf. on Tools with Artificial Intelligence, vol. 2, Patras, Greece, 2007.

[24] K. Bontcheva, Open-source tools for creation, maintenance, and storage of lexical resources for lang. generation from ontologies, in: 4th Conf. on Lang. Resources and Evaluation, Lisbon, Portugal, 2004.

[25] K. Bontcheva, Generating tailored textual summaries from ontologies, in: 2nd European Semantic Web Conf. Heraklion, Greece, 2005.

[26] K. Bontcheva, H. Cunningham, The Semantic Web: a new opportunity and challenge for human language

technology, in: Workshop on Human Lang. Technology for the Semantic Web and Web Services, 2nd Int. Semantic Web Conf. Sanibel Island, FL, 2003.

[27] K. Bontcheva, V. Tablan, D. Maynard, H. Cunningham, Evolving GATE to meet new challenges in language engineering, Nat. Language Engineering 10 (3/4) (2004) 349–373.

[28] K. Bontcheva, Y. Wilks, Automatic report generation from ontologies: the MIAKT approach, in: 9th Int. Conf. on Applications of Nat. Language to Information Systems, Manchester, UK, 2004.

[29] S. Brin, L. Page, The anatomy of a large-scale hypertextual Web search engine, Computer Networks and ISDN Systems 30 (1-7) (1998) 107–117.

[30] C. Brun, M. Dyteman, V. Lux, Document structure and multilingual authoring, in: 1st Int. Nat. Lang. Generation Conf. Mitzpe Ramon, Israel, 2000.

[31] S. Busemann, H. Horacek, A flexible shallow approach to text generation, in: 9th Int. Workshop on Nat. Lang. Generation, New Brunswick, NJ, 1999.

[32] H. Chen, S. Branavan, R. Barzilay, D. Karger, Content modeling using latent permutations, Journal of Artificial Intelligence Research 36 (2009) 129–163.

[33] H. Cheng, C. Mellish, Capturing the interaction between aggregation and text planning in two generation systems, in: 1st Int. Conf. on Nat. Lang. Generation, Mitzpe Ramon, Israel, 2000.

[34] P. Cimiano, P. Buitelaar, J. McCrae, M. Sintek, LexInfo: A declarative model for the lexicon-ontology interface, Web Semantics 9 (1) (2011) 29–51.

[35] H. Clark, Bridging, in: P. Johnson-Laird, P. Wason (eds.), Thinking: Readings in Cognitive Science, Cambridge University Press, 1977, pp. 9–27.

[36] A. Cregan, R. Schwitter, T. Meyer, Sydney OWL syntax – towards a controlled natural language syntax for OWL 1.1, in: OWL Experiences and Directions Workshop, Innsbruck, Austria, 2007.

[37] R. Dale, Generating Referring Expressions: Constructing Descriptions in a Domain of Objects and Processes, MIT Press, 1992.

[38] R. Dale, S. Green, M. Milosavljevic, C. Paris, C. Verspoor, S. Williams, Dynamic document delivery: generating natural language texts on demand, in: 9th Int. Conf. and Workshop on Database and Expert Systems Applications, Vienna, Austria, 1998.

[39] R. Dale, E. Reiter, Computational interpretations of the Gricean maxims in the generation of referring expressions, Cognitive Science 18 (1995) 233–263.

[40] R. Dale, J. Viethen, Attribute-centric referring expression generation, in: E. Krahmer, M. Theune (eds.), Empirical Methods in Nat. Lang. Generation, Springer, 2010, pp. 163–179.

[41] H. Dalianis, Aggregation in nat. lang. generation, Comput. Intelligence 15 (4) (1999) 384–414.

[42] D. Dannells, Applying semantic frame theory to automate natural language template generation from ontology statements, in: 6th Int. Nat. Lang. Generation Conf. Trim, Co. Meath, Ireland, 2010.

[43] D. Dannels, Generating tailored texts for museum exhibits, in: Workshop on Language Technology for Cultural Heritage Data of the Language Resources and Evaluation Conf. Marrakech, Morocco, 2008.

[44] B. Davis, A. Iqbal, A. Funk, V. Tablan, K. Bontcheva, H. Cunningham, S. Handschuh, Roundtrip ontology authoring, in: 7th Int. Conf. on the Semantic Web, Karlsruhe, Germany, 2008.

[45] S. Demir, S. Carberry, K. McCoy, A discourse-aware graph-based content-selection framework, in: 6th Int. NLG Conf. Trim, Co. Meath, Ireland, 2010.

[46] R. Denaux, V. Dimitrova, A. Cohn, C. Dolbear, G. Hart, Rabbit to OWL: Ontology authoring with a CNL-based tool, in: N. Fuchs (ed.), Controlled Nat. Language, vol. 5972 of Lecture Notes in Computer Science, Springer, 2010, pp. 246–264.

[47] R. Denaux, C. Dolbear, G. Hart, V. Dimitrova, A. Cohn, Supporting domain experts to construct conceptual ontologies: A holistic approach, Web Semantics 9 (2) (2011) 113–127.

[48] B. Di Eugenio, D. Fossati, D. Yu, S. Haller, M. Glass, Aggregation improves learning: Experiments in natural language generation for intelligent tutoring systems, in: 43rd Annual Meeting of ACL, Ann Arbor, MI, 2005.

[49] A. Dimitromanolaki, I. Androutsopoulos, Learning to order facts for discourse planning in natural language generation, in: 9th European Workshop on Nat. Lang. Generation, 10th Conf. of the European Chapter of ACL, Budapest, Hungary, 2003.

[50] P. Duboue, K. McKeown, Empirically estimating order constraints for content planning in generation, in: 39th Annual Meeting of ACL, Toulouse, France, 2001.

[51] P. Duboue, K. McKeown, Content planner construction via evolutionary algorithms and a corpus-based fitness function, in: 2nd Int. Nat. Lang. Generation Conf. Harriman, NY, 2002.

[52] P. Duboue, K. McKeown, Statistical acquisition of content selection rules for natural language generation, in: Conf. on Empirical Methods in Nat. Language Processing, Sapporo, Japan, 2003.

[53] M. Dzbor, E. Motta, C. Buil, J. Gomez, O. Goerlitz, H. Lewen, Developing ontologies in OWL: An observational study, in: OWL Experiences and Directions Workshop, Athens, GA, 2006.

[54] M. Elhadad, J. Robin, SURGE: A reusable comprehensive syntactic realization component, in: 8th Int. Nat. Lang. Generation Workshop, Herstmonceux Castle, Sussex, UK, 1996.

[55] M. Elsner, J. Austerweil, E. Charniak, A unified local and global model for discourse coherence, in: Human Lang. Technologies Conf. of the North American Chapter of ACL, Rochester, New York, 2007.

[56] C. Fellbaum (ed.), WordNet: an electronic lexical database, MIT Press, 1998.

[57] G. Fliedl, C. Kop, J. Vohringer, From OWL class and property labels to human understandable natural language, in: 12th Int. Conf. on Applications of Nat. Language to Information Systems, Paris, France, 2007.

[58] N. Fuchs, K. Kaljurand, T. Kuhn, Attempto Controlled English for knowledge representation, in: C. Baroglio, P. Bonatti, J. Maluszynski, M. Marchiori, A. Polleres, S. Schaffert (eds.), Reasoning Web, Springer-Verlag, 2008, pp. 104–124.

[59] A. Funk, V. Tablan, K. Bontcheva, H. Cunningham, B. Davis, S. Handschuh, CLOnE: Controlled language

for ontology editing, in: 6th Int. Semantic Web and 2nd Asian Semantic Web Conf. Busan, Korea, 2007.

[60] D. Galanis, I. Androutsopoulos, Generating multilingual descriptions from linguistically annotated OWL ontologies: the NaturalOWL system, in: 11th European Workshop on Nat. Lang. Generation, Schloss Dagstuhl, Germany, 2007.

[61] D. Galanis, G. Karakatsiotis, G. Lampouras, I. Androutsopoulos, An open-source natural language generator for OWL ontologies and its use in Protégé and Second Life, in: 12th Conf. of the European Chapter of ACL (demos), Athens, Greece, 2009.

[62] A. Gatt, A. Belz, Introducing shared tasks to NLG: the TUNA shared task evaluation challenges, in: E. Krahmer, M. Theune (eds.), Empirical Methods in NLG, Springer-Verlag, 2010, pp. 264–293.

[63] A. Gatt, E. Reiter, SimpleNLG: A realisation engine for practical applications, in: 12th European Workshop on NLG, Athens, Greece, 2009.

[64] J. Gracia, E. Montiel-Ponsoda, P. Cimiano, A. Gomez-Perez, P. Buitelaar, J. McCrae, Challenges for the multilingual Web of Data, Web Semantics 11 (2012) 63–71.

[65] B. Grau, I. Horrocks, B. Motik, B. Parsia, P. Patel-Schneider, U. Sattler, OWL 2: The next step for OWL, Web Semantics 6 (2008) 309–322.

[66] B. Grosz, A. Joshi, S. Weinstein, Centering: a framework for modelling the local coherence of discourse, Comput. Linguistics 21 (2) (1995) 203–225.

[67] Y. Guo, H. Wang, J. van Genabith, Dependency-based n-gram models for general purpose sentence realisation, Nat. Language Engineering 17 (2010) 455–483.

[68] V. Haarslev, R. Moller, RACER system description, in: 1st Int. Joint Conf. on Automated Reasoning, Siena, Italy, 2001.

[69] C. Halaschek-Wiener, J. Golbeck, B. Parsia, V. Kolovski, J. Hendler, Image browsing and natural language paraphrases of semantic web annotations, in: 1st Workshop on Semantic Interoperability in the European Digital Library at the 5th European Semantic Web Conf. Tenerife, Spain, 2008.

[70] C. Hallett, D. Scott, R. Power, Composing questions through conceptual authoring, Comput. Linguistics 33 (2007) 105–133.

[71] M. Halliday, Introduction to Functional Grammar, 2nd ed., Edward Arnold, 1994.

[72] M. Halliday, R. Hassan, Cohesion in English, Longman, 1976.

[73] K. Harbusch, G. Kempen, Generating clausal coordinate ellipsis multilingually: A uniform approach based on postediting, in: 12th European Workshop on Nat. Lang. Generation, Athens, Greece, 2009.

[74] G. Hart, M. Johnson, C. Dolbear, Rabbit: Developing a control natural language for authoring ontologies, in: 5th European Semantic Web Conf. Tenerife, Canary Islands, Spain, 2008.

[75] A. Hartley, C. Paris, Multilingual document production – from support for translating to support for authoring, Machine Translation 12 (1–2) (1997) 109–129.

[76] A. Hartley, D. Scott, J. Bateman, D. Dochev, AGILE – a system for multilingual generation of technical instructions, in: 8th Machine Translation Summit, Santiago de Compostella, Spain, 2001.

[77] M. Hearst, TextTiling: Segmenting text into multi-paragraph subtopic passages, Comput. Linguistics 23 (1) (1997) 33–64.

[78] D. Hewlett, A. Kalyanpur, V. Kolovski, C. Halaschek-Wiener, Effective NL paraphrasing of ontologies on the Semantic Web, in: Workshop on End-User Semantic Web Interaction, 4th Int. Semantic Web Conf. Galway, Ireland, 2005.

[79] H. Horacek, A best-first search algorithm for generating referring expressions, in: 10th Conf. of the European Chapter of ACL, Budapest, Hungary, 2003.

[80] E. Hovy, Automatic discourse generation using discourse structure relations, Artificial Intelligence 63 (1–2) (1993) 341–385.

[81] A. Isard, Choosing the best comparison under the circumstances, in: Int. Workshop on Personalization Enhanced Access to Cultural Heritage, 11th Int. Conf. on User Modeling, Corfu, Greece, 2007.

[82] A. Isard, J. Oberlander, I. Androutsopoulos, C. Matheson, Speaking the users' languages, IEEE Intelligent Systems 18 (1) (2003) 40–45.

[83] K. Kaljurand, Attempto controlled english as a Semantic Web language, Ph.D. thesis, Faculty of Mathematics and Computer Science, University of Tartu, Estonia (2007).

[84] K. Kaljurand, N. Fuchs, Verbalizing OWL in Attempto Controlled English, in: 3rd Int. Workshop on OWL: Experiences and Directions, Innsbruck, Austria, 2007.

[85] G. Karakatsiotis, Automatic generation of comparisons in a natural language generation system, MSc thesis, Department of Informatics, Athens University of Economics and Business, in Greek (2007).

[86] G. Karakatsiotis, D. Galanis, G. Lampouras, I. Androutsopoulos, NaturalOWL: Generating texts from OWL ontologies in Protégé and in Second Life, in: 18th European Conf. on Artificial Intelligence (demos), Patras, Greece, 2008.

[87] N. Karamanis, C. Mellish, M. Poesio, J. Oberlander, Evaluating centering for information ordering using corpora, Comput. Linguistics 35 (1) (2009) 29–46.

[88] A. Karasimos, A. Isard, Multi-lingual evaluation of a natural language generation system, in: 4th Int. Conf. on Language Resources and Evaluation, Lisbon, Portugal, 2004.

[89] R. Kasper, R. Whitney, SPL: A sentence plan language for text generation, Tech. rep., Information Sciences Institute, University of Southern California (1989).

[90] E. Kaufmann, A. Bernstein, Evaluating the usability of natural language query languages and interfaces to Semantic Web knowledge bases, Web Semantics 8 (2010) 377–393.

[91] C. Kelly, A. Copestake, N. Karamanis, Investigating content selection for language generation using machine learning, in: 12th European Workshop on Nat. Lang. Generation, Athens, Greece, 2010.

[92] R. Kibble, R. Power, Optimizing referential coherence in text generation, Comp. Ling. 30 (4) (2004) 401–416.

[93] J. Kim, R. Mooney, Generative alignment and semantic parsing for learning from ambiguous supervision, in: 23rd Int. Conf. on Comput. Ling., Beijing, China, 2010.

[94] W. Kintsch, J. Keenan, Reading rate and retention as a function of the number of propositions in the text base of sentences, Cognitive Psychology 5 (1973) 257–274.

[95] W. Kintsch, T. van Dijk, Towards a model of text comprehension and production, Psychological Review 85 (1978) 363–394.

[96] S. Konstantopoulos, I. Androutsopoulos, H. Baltzakis, V. Karkaletsis, C. Matheson, A. Tegos, P. Trahanias, INDIGO: Interaction with personality and dialogue enabled robots, in: 18th European Conf. on Artificial Intelligence, (demos), Patras, Greece, 2008.

[97] S. Konstantopoulos, A. Tegos, D. Bilidas, I. Androutsopoulos, G. Lampouras, P. Malakasiotis, C. Matheson, O. Deroo, Adaptive natural language interaction, in: 12th Conf. of the European Chapter of ACL (demos), Athens, Greece, 2009.

[98] I. Konstas, M. Lapata, Concept-to-text generation via discriminative reranking, in: 50th Annual Meeting of the ACL, Jeju Island, Korea, 2012.

[99] I. Konstas, M. Lapata, Unsupervised concept-to-text generation with hypergraphs, in: Human Lang. Technology Conf. of the North American Chapter of ACL, Montréal, Canada, 2012.

[100] E. Krahmer, M. Theune (eds.), Empirical Methods in Nat. Lang. Generation, Springer, 2010.

[101] E. Krahmer, K. van Deemter, Computational generation of referring expressions: A survey, Comput. Linguistics 38 (1) (2012) 173–218.

[102] E. Krahmer, S. van Erk, A. Verleg, Graph-based generation of referring expressions, Comput. Linguistics 29 (1) (2003) 53–72.

[103] T. Kuhn, R. Schwitter, Writing support for controlled natural languages, in: Australasian Lang. Technology Association Workshop, Hobart, Australia, 2008.

[104] I. Langkilde, Forest based statistical sentence generation, in: 1st Conf. of the North American Chapter of ACL, Seattle, WA, 2000.

[105] I. Langkilde, K. Knight, Generation that exploits corpus-based statistical knowledge, in: 36th Annual Meeting of ACL and the 17th Int. Conf. on Comput. Linguistics, vol. 1, Montreal, Quebec, Canada, 1998.

[106] I. Langkilde-Geary, An empirical verification of coverage and correctness for a general-purpose sentence generator, in: 2nd Int. Nat. Lang. Generation Workshop, Harriman, NY, 2002.

[107] B. Lavoie, O. Rambow, A fast and portable realizer for text generation systems, in: 5th Conf. on Applied Nat. Language Processing, Washington DC, 1997.

[108] P. Liang, M. Jordan, D. Klein, Learning semantic correspondences with less supervision, in: 47th Annual Meeting of ACL and 4th Conf. of AFNLP, Suntec, Singapore, 2009.

[109] S. Liang, D. Scott, R. Stevens, A. Rector, Unlocking medical ontologies for non-ontology experts, in: 10th Workshop on Biomedical Nat. Language Processing, Portland, OR, 2011.

[110] S. Liang, R. Stevens, D. Scott, A. Rector, Automatic verbalisation of SNOMED classes using OntoVerbal, in: 13th Conf. on AI in Medicine, Bled, Slovenia, 2011.

[111] B. Lonneker-Rodman, C. Baker, The FrameNet model and its applications, Nat. Language Engineering 15 (3) (2009) 414–453.

[112] N. Madnani, B. Dorr, Generating phrasal and sentential paraphrases: A survey of data-driven methods, Comput. Linguistics 36 (3) (2010) 341–387.

[113] W. Mann, S. Thompson, Rhetorical structure theory: A theory of text organization, Text 8 (3) (1998) 243–281.

[114] T. Marciniak, M. Strube, Beyond the pipeline: Discrete optimization in NLP, in: 9th Conf. on Comput. Nat. Language Learning, Ann Arbor, MI, 2005.

[115] M. Marge, A. Isard, J. Moore, Creation of a new domain and evaluation of comparison generation in a natural language generation system, in: 5th Int. Nat. Lang. Generation Conf. Salt Fork, OH, 2008.

[116] K. McKeown, Text generation, Cambridge University Press, 1985.

[117] S. McRoy, S. Channarukul, S. Ali, An augmented template-based approach to text realization, Nat. Language Engineering 9 (4) (2003) 381–420.

[118] A. Melengoglou, Multilingual aggregation in the M-PIRO system, Master's thesis, School of Informatics, University of Edinburgh, UK (2002).

[119] C. Mellish, Using Semantic Web technology to support NLG – case study: OWL finds RAGS, in: 6th Int. NLG Conf. Trim, Co. Meath, Ireland, 2010.

[120] C. Mellish, A. Knott, J. Oberlander, M. O'Donnell, Experiments using stochastic search for text planning, in: Proceesings of the 9th Int. Workshop on NLG, Niagara-on-the-Lake, Ontario, Canada, 1998.

[121] C. Mellish, J. Pan, Finding subsumers for natural language presentation, in: Int. Workshop on Description Logics, Windermere, England, 2006.

[122] C. Mellish, J. Pan, Nat. lang. directed inference from ontologies, Artificial Intelligence 172 (2008) 1285–1315.

[123] C. Mellish, D. Scott, L. Cahill, D. Paiva, R. Evans, M. Reape, A reference architecture for nat. lang. generation systems, Nat. Language Engineering 12 (2006) 1–34.

[124] C. Mellish, X. Sun, Natural language directed inference in the presentation of ontologies, in: 10th European Workshop on NLG, Aberdeen, UK, 2005.

[125] C. Mellish, X. Sun, The Semantic Web as a linguistic resource: opportunities for nat. lang. generation, Knowledge Based Systems 19 (2006) 298–303.

[126] M. Milosavljevic, The automatic generation of comparison in descriptions of entities, Ph.D. thesis, Department of Computing, Macquarie University, Australia (1999).

[127] E. Montiel-Ponsoda, G. A. de Cea, A. Gomez-Perez, W. Peters, Enriching ontologies with multilingual information, Nat. Lang. Eng. 17 (2010) 283–309.

[128] J. Moore, C. Paris, Planning text for advisory dialogues: Capturing intentional and rhetorical information, Comp. Ling. 19 (4) (1993) 651–694.

[129] B. Motik, R. Shearer, I. Horrocks, Optimized reasoning in descr. logics using hypertableaux, in: 21st Int. Conf. on Automated Deduction, Bremen, Germany, 2007.

[130] A. Nenkova, J. Chae, A. Louis, E. Pitler, Structural features for predicting the linguistic quality of text, in: E. Krahmer, M. Theune (eds.), Empirical Methods in Nat. Lang. Generation, Springer, 2010, pp. 222–241.

[131] J. Oberlander, G. Karakatsiotis, A. Isard, I. Androutsopoulos, Building an adaptive museum gallery in Second Life, in: Museums and the Web, Montreal, Quebec, Canada, 2008.

[132] M. O'Donnell, C. Mellish, J. Oberlander, A. Knott, ILEX: an architecture for a dynamic hypertext

generation system, Nat. Language Engineering 7 (3) (2001) 225–250.

[133] I. Paraboni, K. van Deemter, J. Masthoff, Generating referring expressions: making referents easy to identify, Comput. Linguistics 33 (2).

[134] C. Paris, N. Colineau, A. Lampert, K. V. Linden, Discourse planning for information composition and delivery: A reusable platform, Nat. Language Engineering 16 (1) (2009) 61–98.

[135] M. Poesio, R. Stevenson, B. Di Eugenio, Centering: A parameter theory and its instantiations, Comput. Linguistics 30 (3) (2004) 309–363.

[136] R. Power, Towards a generation-based semantic web authoring tool, in: 12th European Workshop on Nat. Lang. Generation, Athens, Greece, 2009.

[137] R. Power, Complexity assumptions in ontology verbalisation, in: 48th Annual Meeting of ACL (short papers), Uppsala, Sweden, 2010.

[138] R. Power, Deriving rhetorical relationships from semantic content, in: 13th European Workshop on Nat. Lang. Generation, Nancy, France, 2011.

[139] R. Power, D. Scott, Multilingual authoring using feedback texts, in: 17th Int. Conf. on Comput. Linguistics and the 36th Annual Meeting of ACL, Montreal, Canada, 1998.

[140] R. Power, R. Stevens, D. Scott, A. Rector, Editing OWL through generated CNL, in: Workshop on Controlled Nat. Lang., Marettimo Island, Italy, 2009.

[141] R. Power, A. Third, Expressing OWL axioms by English sentences: Dubious in theory, feasible in practice, in: 23rd Int. Conf. on Comput. Linguistics (posters), Beijing, China, 2010.

[142] A. Ratnaparkhi, Trainable methods for surface natural language generation, in: 1st Conf. of the North American Chapter of ACL, Seattle, WA, 2000.

[143] A. Rector, N. Drummond, M. Horridge, J. Rogers, H. Knublauch, R. Stevens, H. Wang, C. Wroe, OWL pizzas: Practical experience of teaching OWL-DL: Common errors and common patterns, in: 14th Int. Conf. on Knowledge Engineering and Knowledge Management, Northamptonshire, UK, 2004.

[144] E. Reiter, NLG vs. templates, in: 5th European Workshop on Nat. Lang. Generation, Leiden, The Netherlands, 1995.

[145] E. Reiter, R. Dale, Building Natural Language Generation systems, Cambridge University Press, 2000.

[146] E. Reiter, R. Robertson, L. Osman, Knowledge acquisition for natural language generation, in: 1st Int. Conf. on NLG, Mitzpe Ramon, Israel, 2000.

[147] Y. Ren, K. van Deemter, J. Pan, Charting the potential of description logic for the generation of referring expressions, in: 6th Int. Nat. Lang. Generation Conf. Trim, Co. Meath, Ireland, 2010.

[148] C. Sauper, R. Barzilay, Automatically generating Wikipedia articles: A structure-aware approach, in: Joint Conf. of the 47th Annual Meeting of ACL and the 4th Int. Joint Conf. on Nat. Language Processing of the AFNLP, Suntec, Singapore, 2009.

[149] N. Schutte, Generating nat. language descriptions of ontology concepts, in: 12th European Workshop on Nat. Lang. Generation, Athens, Greece, 2009.

[150] R. Schwitter, English as a formal specification language, in: 13th Int. Workshop on Database and Expert Systems Applications, Aix-en-Provence, France, 2002.

[151] R. Schwitter, Controlled nat. languages for knowledge representation, in: 23rd Int. Conf. on Comput. Linguistics (posters), Beijing, China, 2010.

[152] R. Schwitter, Creating and querying formal ontologies via controlled nat. language, Applied Artificial Intelligence 24 (2010) 149–174.

[153] R. Schwitter, K. Kaljurand, A. Cregan, C. Dolbear, G. Hart, A comparison of three controlled nat. languages for OWL 1.1, in: 4th OWL Experiences and Directions Workshop, Washington DC, 2008.

[154] R. Schwitter, M. Tilbrook, Controlled natural language meets the Semantic Web, in: Australasian Language Technology Workshop, Macquarie University, Sydney, Australia, 2004.

[155] N. Shadbolt, T. Berners-Lee, W. Hall, The Semantic Web revisited, IEEE Intell. Systems 21 (2006) 96–101.

[156] E. Sirin, B. Parsia, B. Grau, A. Kalyanpur, Y. Katz, Pellet: A practical OWL-DL reasoner, Web Semantics 5 (2) (2007) 51–53.

[157] R. Stevens, J. Malone, S. Williams, R. Power, Automating class definitions from OWL to English, in: Bio-Ontologies: Semantic Applications in Life Sciences SIG at the 18th Annual Int. Conf. on Intelligent Systems for Molecular Biology, Boston, MA, 2010.

[158] R. Stevens, J. Malone, S. Williams, R. Power, A. Third, Automatic generation of textual class definitions from OWL to English, Biomedical Semantics 2 (S 2:S5).

[159] X. Sun, C. Mellish, Domain independent sentence generation from RDF representations for the Semantic Web, in: Combined Workshop on Language-Enabled Educational Technology and Development and Evaluation of Robust Spoken Dialogue Systems, European Conf. on AI, Riva del Garda, Italy, 2006.

[160] X. Sun, C. Mellish, An experiment on free generation from single RDF triples, in: 11th European Workshop on NLG, Schloss Dagstuhl, Germany, 2007.

[161] V. Tablan, T. Polajnar, H. Cunningham, K. Bontcheva, User-friendly ontology authoring using a controlled language, in: 5th Int. Conf. on Language Resources and Evaluation, Genoa, Italy, 2006.

[162] M. Theune, E. Klabbers, J. De Pijper, E. Krahmer, J. Odijk, From data to speech: a general approach, Nat. Language Engineering 7 (1) (2001) 47–86.

[163] D. Tsarkov, I. Horrocks, FaCT++ description logic reasoner: System description, in: 3rd Int. Joint Conf. on Automated Reasoning, Seattle, WA, 2006.

[164] K. Van Deemter, Generating referring expressions: Boolean extensions of the Incremental Algorithm, Comput. Linguistics 28 (1) (2002) 37–52.

[165] K. van Deemter, Generating referring expressions that involve gradable properties, Comp. Ling. 32 (2).

[166] K. van Deemter, E. Krahmer, M. Theune, Real versus template-based natural language generation: a false opposition?, Comput. Linguistics 31 (1) (2005) 15–24.

[167] K. Van Deemter, R. Power, High-level authoring of illustrated documents, Nat. Language Engineering 9 (2) (2003) 101–126.

[168] S. Varges, C. Mellish, Instance-based natural language generation, Nat. Lang. Eng. 16 (3) (2010) 309–346.

[169] D. Vogiatzis, D. Galanis, V. Karkaletsis, I. Androutsopoulos, C. Spyropoulos, A conversant

robotic guide to art collections, in: 2nd Workshop on Language Technology for Cultural Heritage Data, Language Resources and Evaluation Conf. 2008.

[170] M. Walker, A. Joshi, E. Prince (eds.), Centering Theory in Discourse, Oxford University Press, 1998.

[171] M. Walker, O. Rambow, M. Rogati, Spot: A trainable sentence planner, in: 2nd Annual Meeting of the North American Chapter of ACL, Pittsburgh, PA, 2001.

[172] S. Wan, M. Dras, R. Dale, C. Paris, Spanning tree approaches for statistical sentence generation, in: E. Krahmer, M. Theune (eds.), Empirical Methods in Nat. Lang. Generation, Springer, 2010, pp. 13–44.

[173] M. White, CCG chart realization from disjunctive inputs, in: 4th Int. Nat. Lang. Generation Conf. Sydney, Australia, 2006.

[174] G. Wilcock, Integrating natural language generation with XML web technology, in: 10th Conf. of the European Chapter of ACL (Conf. Companion), Budapest, Hungary, 2003.

[175] G. Wilcock, Talking OWLs: towards an ontology verbalizer, in: Workshop on Human Lang. Technology for the Semantic Web, 2nd Int. Semantic Web Conf. Sanibel Island, FL, 2003.

[176] G. Wilcock, An overview of shallow XML-based natural language generation, in: 2nd Baltic Conf. on Human Lang. Technologies, Tallinn, Estonia, 2005.

[177] S. Williams, R. Power, Grouping axioms for more coherent ontology descriptions, in: 6th Int. Nat. Lang. Generation Conf. Trim, Co. Meath, Ireland, 2010.

[178] S. Williams, A. Third, R. Power, Levels of organization in ontology verbalization, in: 13th European Workshop on Nat. Lang. Generation, Nancy, France, 2011.

[179] J. Yu, E. Reiter, H. J., C. Mellish, Choosing the content of textual summaries of large time-series data sets, Nat. Language Engineering 13 (1) (2006) 25–49.