

Natural Language Generation and Interfaces to Knowledge Bases

Demonstrations

Claire Gardent¹, Eva Banik² and Laura Perez-Beltrachini³

(1) CNRS/LORIA, Nancy, (2) Computational Linguistics Ltd, (3) Nancy University

26 June 2011, K-CAP 2011

Table of Contents

Example 1: Verbalisation of owl classes and properties, and sentence planning.....	4
Example 2: Semantic preserving rewriting of axioms.....	5
Class expression axioms (Axioms that allow relationships to be established between class expressions):.....	5
Property Axioms (Axioms that can be used to characterize and establish relationships between object property expressions):	6
Example 1: Lexicon automatically constructed. Lexical entries for class, individual and property names.....	7
Example 1: Getting frequency counts of ontology axiom patterns (using Axiom patterns (frequency counts)).....	7
Example 1: \Verbalisation of OWL axioms.	8
Example 1.1: Consensus model EquivalentTo as a Aristotelian definition.....	8
Example 1.2: Switching to technical English.....	8
Example 1: Grouping axioms (Power&Williams INLG 2010) . Aggregation grouping axioms of type SubClassOf	9
Example 2: The target text is an English glossary, axioms realised under different glossary tags	9
Example 1: Generating individual descriptions, example for exhibit1:amphora.....	10
Example 2: Annotations.....	11
Example 3: NLG tasks working example.....	14
Example 1: Initial query: general request.....	16
Example 2: Refinement process of the query. Query manipulation operations	17
Example 2.1: Substitute, add concept and relations and delete.....	17
Example 2.1.1: Substitute.....	17
Example 2.1.2: Add concepts.....	17
Example 2.1.2: Add relations.....	17
Example 2.2: Replacement of a complex concept description by an equivalent concept....	18
Example 3: Text highlighting gives visual hints of the underlying query tree. Conceptually aligned text.....	18
Example 4: Document planning carried out by the user.....	18
Example 5: Microplanning	18
Example 5.1: Concept templates: Noun-based or Adjective-based.....	19
Example 5.2: Relation templates. Different relation templates for different relation types.....	19
Example 6: Improving readability: aggregation and anaphoric references.....	20
Example 6.1: REG approach.....	20
Example 6.3: Aggregation.....	20
Example 1: Initial seed axiom.....	21
Example 2: Editing ontology lexicon: adding new terms for classes and properties.....	21
Example 3: Authoring the knowledge base and verbalisation of suggestions.	21
Example 3.1: Authoring the fist SubClassOf axiom.....	21
Example 3.2: Authoring another SubClassOf involving an ObjectSomeValuesFrom restriction.....	22
Example 4: Usage of reasoning services when building the knowledge base to verify acceptability constraints (consistency and informativeness).....	23

Example 5: Authoring a SubClassOf axiom involving an ObjectComplementOf class expression.....	23
Example 6: Authoring a SubClassOf axiom involving an ObjectUnionOf class expression....	23
Example 7: Looking at some of the underlying English grammar rules.....	23
Example 1: Lexicon edition, different lexicon/syntax for owl relations, hierarchy creation.....	24
Example 2: Difference between 'A is a ...' and 'Every is a'	24
Example 3: Authoring more complex constructions.....	24
Example 3.1: Aggregation.....	24
Example 4: Prediction of anaphoric references.....	25
Example 5: Take the generated sentences into Protégé to build the ontology.....	26

Demo1: Verbalising OWL in ACE

- Attempto research project from the University of Zurich that has developed the Attempto Controlled English (ACE) and related tools (<http://attempto.ifi.uzh.ch>) , one of them is the **OWL verbalizer**.
- ACE is a subset of English where ambiguity is ruled out and that can be converted into FOL.
- verbalisation of OWL in ACE should be **reversible**. i.e. one can convert the ACE representation back into OWL so that no loss in meaning occurs.
 - has a formal language semantics and comes with parsers that could convert back into OWL representation.
- Set of **construction rules** and set of **interpretation rules** --quantification, relative clause coordination, anaphoric references, among others .

verbalisation approach:

- OWL axiom constructors map to NL syntactic constructions.
- Tries to keep 1-sentence 1-axiom but for readability in some cases: aggregation and ordering (simple class descriptions mentioned first).
 - Leaves the structure of the input ontology as far as possible intact, making visible to OWL experts how constructs where mapped into ACE.
- OWL verbalisation concise and understandable, **provided certain naming conventions** for OWL individual, classes and relations.
- Verbalisation approach preserves the semantics of axioms. i.e. syntactically different axioms can be mapped to the same verbalisation given that they are semantically equivalent.

Examples:

- Following examples taken from ontologies: `people+pets.owl` and `family.swrl.owl`
- Using Protégé plugin or OWL verbalizer available in the ACE web site.

Example 1: Verbalisation of owl classes and properties, and sentence planning

- **Mapping terms to words, automatic building of lexical entries:**
 - individuals to proper nouns,
 - classes to nouns,
 - properties to transitive verbs in active voice and inverse properties to transitive verbs in passive voice.
 - class expressions (aka descriptions or in DL complex concepts) map to ACE complex noun phrases with 'and that', 'or that', 'that is/does not' and 'that'.
 - No verbalisation of data-valued properties
- **Mapping axioms to sentences:**
 - except *ClassAssertion*-axioms the other axioms map to *Every*-sentences with pattern *NounPhrase VerbPhrase*.
- Examples, in the Protégé plugin load the sample ontologies and look at the following cases:
 1. ACE lexicon tab, common nouns, proper nouns, and verbs.
 2. Complex noun-phrase example: `haulage_truck_driver`.
 3. `Fido is a dog . (Fido Type dog)`
 4. `Every truck is a vehicle. (truck SubClassOf vehicle)`
 5. `Every dog eats a bone. (dog SubClassOf eats some bone)`
 6. `Everything that is eaten by a sheep is a grass . (sheep SubClassOf eats only grass)`

7. No cat is a dog . (cat DisjointWith dog)

Example 2: Semantic preserving rewriting of axioms

- Semantic preserving rewriting of axioms. Axioms for which there is no unambiguous wording in Natural Language, e.g. ObjectPropertyRange how to verbalize 'range' for each particular case? Then replace OWL constructs with general ones like: SubClassOf, SubPropertyOf, ClassAssertion, etc.

Class expression axioms (Axioms that allow relationships to be established between class expressions):

1. SubClassOf

Ontology axiom:
Uncle SubClassOf Relative

Generated sentence:
Every Uncle is a Relative.

2. EquivalentTo

Ontology axiom:
white_van_man EquivalentTo man and (drives some (van and white_thing))

Rewriting rule for EquivalentTo(C1 ... Cn)
SubClassOf(C1 C2), SubClassOf(C2 C1), ...

Verbalisation:
Every white_van_man is a man that drives a van that is a white_thing.
Every man that drives a van that is a white_thing is a white_van_man.

3. EquivalentTo with ObjectUnionOf , ObjectSomeValuesFrom constructors in the right hand side complex class expression.

Ontology axiom:
haulage_worker EquivalentTo works_for some (haulage_company or (part_of some haulage_company))

Verbalisation:
Every haulage_worker works_for something that is a haulage_company or that part_ofs a haulage_company. Everything that works_for something that is a haulage_company or that part_ofs a haulage_company is a haulage_worker.

4. EquivalentTo with ObjectIntersectionOf , ObjectSomeValuesFrom constructors in the right hand side complex class expression.

Ontology axiom:
mad_cow EquivalentTo cow and (eats some (brain and (part_of some sheep)))

Verbalisation:
Every mad_cow is a cow that eats a brain that part_ofs a sheep.

Every cow that eats a brain that part_ofs a sheep is a mad_cow .

Ontology axiom:

woman EquivalentTo adult and female and person

Verbalisation:

Every woman is an adult that is a female and that is a person.

Every adult that is a female and that is a person is a woman.

5. **EquivalentTo with ObjectUnionOf constructor in the right hand side complex class expression.**

Ontology axiom:

Relative EquivalentTo Aunt or Child or Nephew or Niece or Parent or Sibling or Uncle

Verbalisation:

Every Relative is something that is an Aunt or that is a Child or that is a Nephew or that is a Niece or that is a Parent or that is a Sibling or that is an Uncle. Everything that is an Aunt or that is a Child or that is a Nephew or that is a Niece or that is a Parent or that is a Sibling or that is an Uncle is a Relative.

Property Axioms (Axioms that can be used to characterize and establish relationships between object property expressions):

6. **FunctionalObjectProperty(R) SymmetricObjectProperty(R)**

(a) Ontology axiom:

Functional: hasConsort

Rewriting rule:

SubClassOf(owl:Thing ObjectMaxCardinality(1 R owl:Thing))

Verbalisation:

Everything hasConsorts at most 1 thing .

(b) Ontology axiom (OWL constructors that represent logical concepts which cannot be expressed in non-technical English):

Symmetric: hasConsort

Verbalisation:

If X hasConsorts Y then Y hasConsorts X .

(c) Ontology axiom:

Reflexive: hasConsort

Verbalisation:

Everything hasConsorts **itself** .

7. ObjectPropertyDomain(R C) ObjectPropertyRange(R C)

- (a) Ontology Axiom:
has_pet Domain person

Verbalisation:

Everything that has_pets something is a person .

- (b) Ontology Axiom:
reads Range publication

Verbalisation:

Everything that is readed by something is a publication .

NL description in OWL manual: The range of the *a:reads* property is the class *a:publication*.

8. InverseObjectProperties(R S)

- (a) Ontology Axiom:
eats InverseOf eaten_by

Verbalisation:

If X eatses Y then Y eaten_bies X. If X eaten_bies Y then Y eatses X .

Demo 2: Generating text from knowledge bases. The SWAT System

Build a lexicon from an OWL ontology

Example 1: Lexicon automatically constructed. Lexical entries for class, individual and property names.¹

```
lex(namedIndividual('Fido'), name, 'Fido', 'Fido').
lex(class('dog'), noun, dog, dogs).
lex(objectProperty('eats'), verb, eats, eat).
lex(objectProperty('eaten\_by'), verb, 'is eaten by', 'are eaten by').
```

SWAT OWL axiom pattern frequency tool

Example 1: Getting frequency counts of ontology axiom patterns (using Axiom patterns (frequency counts))

List of axioms abstract forms.

¹ Lexical entries are automatically derived from the entity's IRIs not from label annotations (the same as ACE and the rest of the verbalisation tools discussed in this paper).

OWL ontology into English sentences

Example 1: Verbalisation of OWL axioms.

Example 1.1: Consensus model EquivalentTo as a Aristotelian definition.

1. EquivalentTo(C1 ... Cn)

Ontology axiom:

white_van_man EquivalentTo man and (drives some (van and white_thing))

Verbalisation:

A white van man is defined as a man that drives a van that is a white thing.

Consensus model: observation about *information structure* is that designers tend to use EquivalentTo class definitions as a sentential description subject and predication about this subject. Then is natural to use the Aristotelian pattern for definitions 'is defined as'

Example 1.2: Switching to technical English

2. InverseObjectProperties(R S)

Ontology Axiom:

eats InverseOf eaten_by

Verbalisation:

X eats Y if and only if Y is eaten by X.

Note: difference with ACE verbalisation, in this case, only one sentence using 'if and only if' *Logical sophistication*, no other possible lexicalisation than technical english.

SWAT Alphabetical English glossary from class, individual and property definitions

- Organizes the information to be verbalized, generates hypertext (WilliamsThirdPower -ENLG2011).
 - Given an ontology, the tool produces as output a set of glossary entries (for classes, individuals and properties) in alphabetical order accompanied by descriptions in English sentences.
 - Each entry is subdivided into logically-based headings like 'Definition' and 'Examples'
 - Instead of verbalizing axioms one by one==> applies some structuring rules and aggregation (applicable to any ontology) to provide coherent descriptions.

Verbalisation approach

- NLG:
 - lexicon: inferring lexical entries for atomic entities from identifiers (special attention on POS and plural forms)

- content selection: grouping related axioms for the given entity (axioms defining the entity and axioms where the entity is used to construct other definitions).
- text planning: organizing selected content into glossary tags
- aggregation: grouping of axioms sharing a common pattern (to obtain more fluent descriptions)
- realization: sentence generation is accomplished by a generic grammar with rules for each different owl patterns and constructors.

Example 1: Grouping axioms (Power&Williams INLG 2010) . Aggregation grouping axioms of type SubClassOf

1. Aggregation at the left:

Underlying ontology axioms:

```
bicycle SubClassOf vehicle
van SubClassOf vehicle
coach SubClassOf vehicle
truck SubClassOf vehicle
```

SWAT verbalisation (using Alphabetical English glossary - class, individual and property definitions):

VEHICLE (class)

Examples The following are [vehicles](#): [vans](#), [trucks](#) and [coaches](#), and so on (5 items).

2. No aggregation in this case because the right-hand side of the subClassOf axioms does not follow the same pattern.

Underlying ontology axioms:

```
sheep SubClassOf eats only grass
sheep SubClassOf animal
```

SWAT verbalisation (Alphabetical English glossary - class, individual and property definitions):

SHEEP (class)

Typology A [sheep](#) is an [animal](#).

Description A [sheep eats](#) only a [grass](#).

SWAT verbalisation (English sentences (one sentence per OWL axiom)):

A sheep is an animal.

A sheep eats only a grass.

Example 2: The target text is an English glossary, axioms realised under different glossary tags

(a) For class entries:

- **Examples.** *SubClassOf* axiom where sub-classes are verbalized as examples of the topic class. *ClassAssertion* axioms of the topic class are verbalized as examples.
- **Description.** *SubClassOf* where the non-atomic super-class is verbalized as a description of the topic class. *Range* and *Domain* property axioms where the topic class participates in.
- **Definition.** *EquivalentTo* axiom.

- **Distinctions.** *DisjointUnion*.
- **Typology.** *SubClassOf* axiom verbalizing the topic class's super-class with a copula sentence. (Note 'typology' verbalisation is not used in glossary for direct sub-classes of *Thing*).

Look at examples class animal, animal_lover and broadsheet. (people+petsB.owl)

(b) For individuals:

- **Description.** *ObjectPropertyAssertion* axioms (where the instance is either the domain or the range). *ClassAssertion* axioms with property restrictions.
- **Typology.** *ClassAssertion* axioms with atomic concepts.

Look at example individual Fido, Joe, Mick and Minnie. (people+petsB.owl)

(c) For properties:

- **Definition.** *ObjectPropertyRange*, *ObjectPropertyDomain*
- **Typology.** *SubObjectPropertyOf* the described property is a sub-property
- **Description.** *SubObjectPropertyOf* the described property is a super-property.
InverseObjectProperties
- **Examples.** *ObjectPropertyAssertion*.

Look at example relations had_father, has_mother, has_parent, has_as_pet (people+petsB.owl)

Demo 3: Generating text from knowledge bases. NaturalOWL

- <http://www.vimeo.com/801099>
- content selection: the system consults a model of the user and a representation of the interaction history, in order to select a subset of the information stored in its knowledge base
- text planning: this information is placed in a specific order, taking into account the entities and relationships mentioned in each piece of information.
- microplanning: the system's microplanner specifies the verbs and noun phrases, and how they should be 'aggregated' together, so that more than one idea can be expressed in a given sentence.

Example 1: Generating individual descriptions, example for exhibit1:amphora.

CLASS BROWSER
 For Project: ● mpiro
 Class Hierarchy

- complex-statue (1)
- imperial-portrait (2)
- kouros (1)
- portrait (2)
- ▼ ● vessel
 - ▼ ● amphora (1)
 - panathenaic-amphora (2)
 - aryballos (1)
 - ● cauldron

INSTANCE BROWSER
 For Class: ● amphora
 :NAME 🔍 🔍 🔍 🔍 🔍
 ◆ exhibit1

TEXT PREVIEW
 Language English User Type ExpertAdult Maximum Graph Distance In Content Selection 1

Preview
Reset interaction history

☐ Generate Comparisons
 ☐ Show Semantic and Syntactic Annotations

This is an amphora; it was painted by the Painter of Kleophrades and it was created during the archaic period. It dates from the early 5th century B.C. It depicts a warrior performing splachnoscopy before leaving for battle. Currently this amphora is exhibited in the Martin von Wagner Museum.

This is an amphora; it was painted by the Painter of Kleophrades and it was created during the archaic period. It dates from the early 5th century B.C. It depicts “a warrior performing splachnoscapy before leaving for battle”. Today this amphora is exhibited in the Martin von Wagner Museum.

Example 2: Annotations

Lexicon:

- Named atomic concept and individuals are associated to a lexical entry which describes the following information:
 - Singular and plural word forms
 - Gender, default number and whether the noun is countable or not.

CLASS BROWSER






For Project: ● mpiro

Class Hierarchy

- copy (9)
- creation-time-values (47)
- exhibit-form-stories (19)
- ▼ ● exhibit
 - amphora-handle (1)
 - coin (1)
 - decorative-artefacts (1)
 - Figurine (2)
 - Jewel
 - military
 - painting (1)
 - potsherd (1)
 - relief (5)
 - ▼ ● statue (2)
 - complex-statue (1)
 - imperial-portrait (2)
 - kouroi (1)
 - portrait (2)
 - ▼ ● vessel
 - ▼ ● amphora (1)
 - panathenaic-amphora (2)
 - aryballos (1)

INSTANCE BROWSER

For Class: ● amphora

:NAME     

◆ exhibit1

LEXICON BROWSER

Resource ● amphora For Language English ☐ Multilingual Personalized Canned Text

Singular form

Plural form

Natural gender

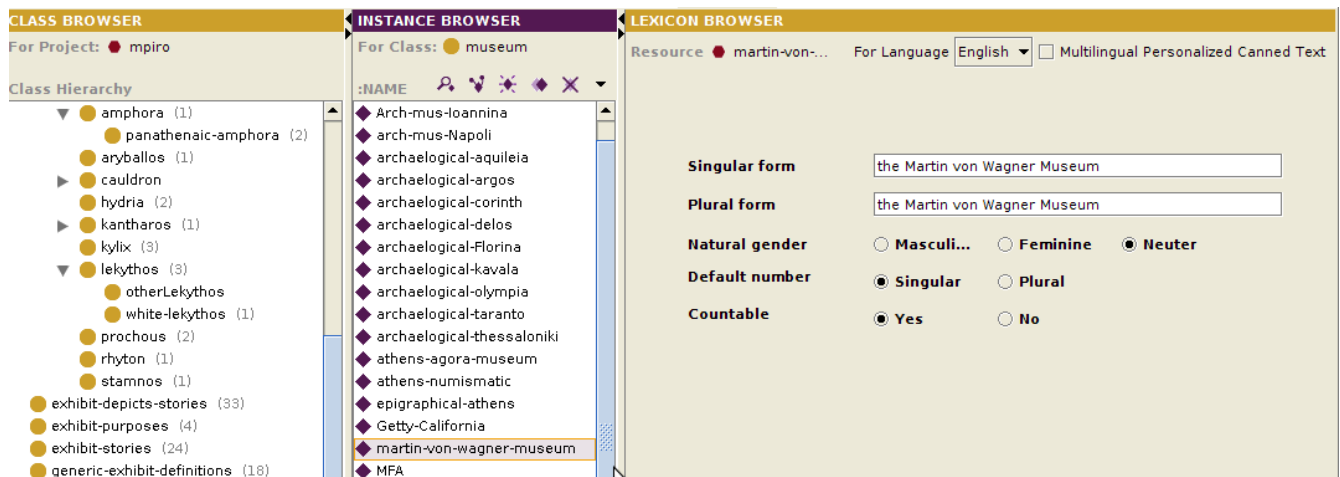
☐ Masculine
 ☐ Feminine
 ☒ Neuter

Default number

☒ Singular
 ☐ Plural

Countable

☒ Yes
 ☐ No



Microplanning and ordering: Templates

- Relations are associated with templates, e.g. **painted-by**
- Each template defines the syntactic structure of a sentence, i.e. a grammar rule with underspecified referring expressions.
- Subject and object slots from templates are filled with an automatically generated referring expression.
- Each template is composed by slots, each slot is of one of the following types: property owner, property value, string (canned text), verb, preposition and adverb. For the **painted-by** relation the template is defined:
 - slot0 = property owner. With type=auto this means that the filling np can be determined automatically, case=nominative.
 - Slot1 = verb. With tense=past, voice = passive, and both singular and plural past passive forms.
 - Slot2 = string. With value = by.
 - Slot3 = property value. With type = auto, case = accusative.
- For each microplanning template an order is specified which is then used to order properties' descriptions. Although is-relations are ordered first.

The template for **painted-by** in xml format:

```
<owl:EnglishMicroplans rdf:parseType="Collection">
  <owl:Microplan rdf:about="http://www.aueb.gr/users/ion/mpiro.owl#painted-by-templ1-en">
    <owl:MicroplanName>templ1</owl:MicroplanName>
    <owl:Used>true</owl:Used>
    <owl:AggrAllowed>true</owl:AggrAllowed>
    <owl:Slots rdf:parseType="Collection">
      <owl:Owner>
        <owl:case>nominative</owl:case>
        <owl:RETYPE>RE_AUTO</owl:RETYPE>
      </owl:Owner>
      <owl:Verb>
        <owl:voice>passive</owl:voice>
        <owl:tense>past</owl:tense>
        <owl:Val xml:lang="en">was painted</owl:Val>
        <owl:pluralVal xml:lang="en">were painted</owl:pluralVal>
      </owl:Verb>
      <owl:Text>
        <owl:Val xml:lang="en">by</owl:Val>
      </owl:Text>
      <owl:Filler>
        <owl:case>accusative</owl:case>
        <owl:RETYPE>RE_AUTO</owl:RETYPE>
      </owl:Filler>
    </owl:Slots>
  </owl:Microplan>
</owl:EnglishMicroplans>
```

- Similar templates are associated to relations **creation-period** (*was created*), **creation-time** (*dates*), **current-location** (*exhibited*) and **exhibit-depicts** (*depicts*).
 - The microplanning template for the relation **current-location** contains a first slot (slot0) of type

string with value 'Today'.

- The microplanning template associated to the relation **exhibit-depicts** contains three slots with respective types: property owner, verb, property value. However, the relation's range is the class **exhibit-depicts-stories**, and each instance of this class has an associated canned text which is used to fill in the property value slot. In the example generated text above:

```
<owl:Val rdf:ID="exhibit1dpk-CT2">
<owl:Val xml:lang="el">απεικονίζει έναν πολεμιστή να εκτελεί σπλαγχνοσκοπία πριν φύγει για τη μάχη</owl:Val>
<owl:Val xml:lang="en">a warrior performing splachnoscopy before leaving for battle</owl:Val>
<owl:forUserType rdf:resource="http://www.aueb.gr/users/ion/owl/UserModelling#ExpertAdult"/>
<owl:forUserType rdf:resource="http://www.aueb.gr/users/ion/owl/UserModelling#ExpertAdultG"/>
<owl:FOCUS_LOST>true</owl:FOCUS_LOST>
<owl:FillerAggrAllowed>false</owl:FillerAggrAllowed>
</owl:Val CannedText>
```

User modeling

- Different type of users (stereotypes)
- users' interests scores for every instance
- interaction history (assimilation metric)

USER TYPES

Project mpiro

UserTypes

- AdultG
- Child
- Adult**
- ChildG
- ExpertAdultG
- ExpertAdult

Adult

Facts per page

Maximum facts per sentence

Language

INTEREST AND REPETITIONS OF PROPERTIES (RIGHT-CLICK ON A PROPERTY T...

Resource amphora

Property	Filler
http://www.aueb.gr/users/ion/mpirowl#painting-style	Filler
http://www.aueb.gr/users/ion/mpirowl#donate-by	Filler
http://www.aueb.gr/users/ion/mpirowl#exhibit-story	Filler
http://www.aueb.gr/users/ion/mpirowl#coord	Filler
http://www.aueb.gr/users/ion/mpirowl#objectProperty_2	Filler
http://www.aueb.gr/users/ion/mpirowl#made-of	Filler
http://www.aueb.gr/users/ion/mpirowl#currentLocation	Filler

Interest-Repetitions for < *,
http://www.aueb.gr/users/ion/mpirowl#creation-time, * >

User Type	Interest	Repetitions
AdultG	6	1
Child	6	1
Adult	6	1
ChildG	6	1
ExpertAdultG	6	1
ExpertAdult	6	1

Other annotations

- controlling the length of the output text: Semantic distance of the selected facts to the instance being described.
- generation of comparisons, usage of interaction history.

Example 3: NLG tasks working example

Following example illustrates the workings of the NLG architecture in Natural OWL (1) changes in content selection according to user model, interaction history, generation of comparisons and graph distance parameter, (2) document plan (3) and working of aggregation and REG:

- Preview text for **exhibit1** instance of class **amphora**, user=expertAdult, Maximum graph distance=1.
This is an amphora; it was painted by the Painter of Kleophrades and it was created during the archaic period. It dates from the early 5th century B.C. It depicts a warrior performing splachnoscropy before leaving for battle. Currently this amphora is exhibited in the Martin von Wagner Museum.
 - Templates for current-location and painted-by, ordering annotation for painted-by=1 and ordering annotation for current-location=10 then the document planner places 'painted by' before in the text plan.
 - Ordering with respect to microplans
- Look at semantic annotations on generated text, Assim=0.
- Re-click on preview ==> the Assim annotation changes to 1.
- select child user stereotype, observe text content change, eg. decorated with and explanation about splachnoscropy
*This is an amphora, **decorated** with the red-figure technique. It was painted by the Painter of Kleophrades and it was created during the archaic period. It dates from the early 5th century B.C. It depicts a warrior performing splachnoscropy before leaving for battle. **Splachnoscropy is the study of animal entrails, through which people tried to predict the future. It was one of the most common divination methods used in the archaic period.** Currently this amphora is exhibited in the Martin von Wagner Museum.*
- Pick up **exhibit48** instance of class **prochous**, back to user stereotype=expertAdult, with the 'generate comparisons' option enabled, preview text:
*This is a prochous; it was found in another Vitsa. **Like the amphora that you saw earlier,** it was created during the archaic period. It is made of bronze and today it is exhibited in the Archaeological Museum of Ioannina.*
- The same as before but without 'generate comparisons' enabled:
This is a prochous; it was found in Vitsa and it was created during the archaic period. It is made of bronze and today it is exhibited in the Archaeological Museum of Ioannina.
- Select user stereotype child and preview text for both graph distance 1 and 2:
This is a prochous, found in Vitsa. It was created during the archaic period and it is made of bronze. Currently it is exhibited in the Archaeological Museum of Ioannina.

This is a prochous. A prochous is a vessel that was used to pour liquids. It has one handle, a large body, and a nib-like neck. This particular prochous was found in Vitsa and it was created during the archaic period. [The archaic period was the period during which the Greek ancient city-states developed and it covers the time between 700 and 480 B.C.] This prochous is made of bronze and currently it is exhibited in the Archaeological Museum of Ioannina.
- Make graph distance equal to 2, overview of interest annotations for **exhibit-description** relation. Then, preview text for both Adult and ExpertAdult:
*This is a prochous. **A prochous is a vessel that was used to pour liquids. It has one handle, a large body, and a nib-like neck. This particular prochous was found in Vitsa and it was created during the archaic period. The archaic period was the period during which the Greek ancient city-states developed and it covers the time between 700 and 480 B.C.** This prochous is made of bronze and currently it is exhibited in the Archaeological Museum of Ioannina.*

This is a prochous; it was found in Vitsa and it was created during the archaic period. It is made of bronze and today it is exhibited in the Archaeological Museum of Ioannina.

Demo4: Quelo

Three software components in Quelo:

- the query logic
- user interface
- NLG component

Quelo query model:

- Query (logic) language: atomic concept instantiation, limited existential restriction (roles or object relations) and conjunction (any list of atomic concept instantiation or roles that must hold at the same time).
- Quelo query model (a data structure for representing tree-shaped conjunctive DL query):
 - a labeled directed n-ary tree
 - nodes labeled with sequence of node labels (each node label refers to an atomic concept of the ontology)
 - edges labeled with an edge label (each edge label refers to a relation of the ontology)

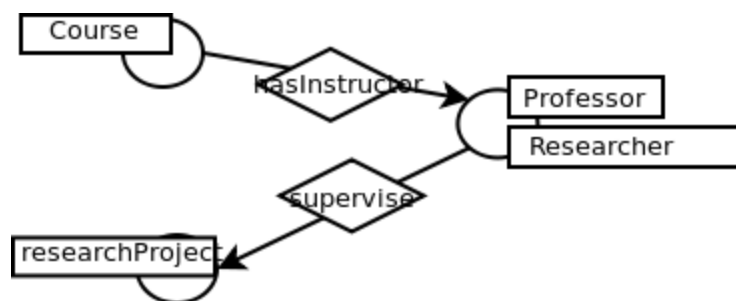
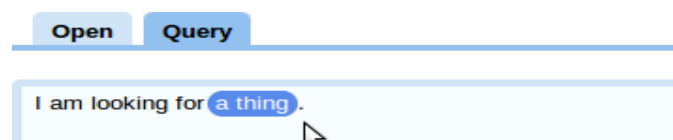


Illustration 1.

- A function roll-up associates each node of the Quelo query to a tree shaped conjunctive DL query. (The query encoded by the tree is a formula –obtained by roll-up – of the root node).
- Editing operations defined in the underlying DL query. Four operations for the query model:
 - addCompatible(node, concept)
 - addProperty(node, relation, concept)
 - substitute(selection, concept)
 - delete(selection)
- The query is presented in English text and these operations are realised as insertion, deletion or substitution of portions of text.

Example 1: Initial query: general request

- Initial request is expressed with a start sentence that contextualizes the query and the verbalisation of the element that subsumes all the concepts, Thing:

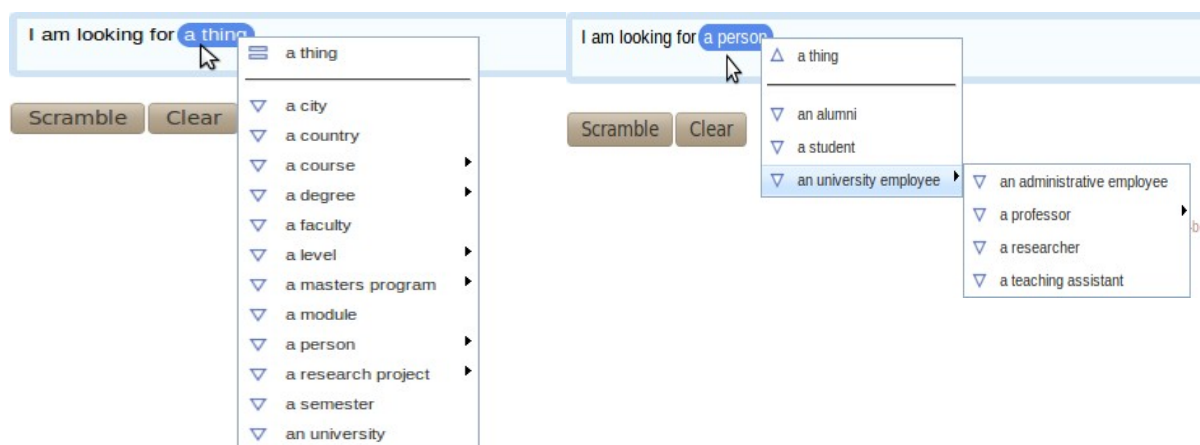


Example 2: Refinement process of the query. Query manipulation operations

- **Replace text:** substitute the focused node by a super or sub concept or an equivalent concept, **insert text:** add compatible concept or relation, **delete text:** delete query operations . The suggestions for these operations are provided by a reasoning service running over the query and ontology.
- The query logic component provides the content for the popup menus which is verbalized by the NLG component.

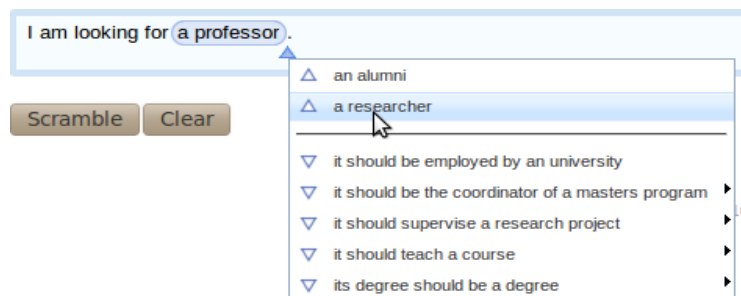
Example 2.1: Substitute, add concept and relations and delete.

Example 2.1.1: Substitute



Example 2.1.2: Add concepts

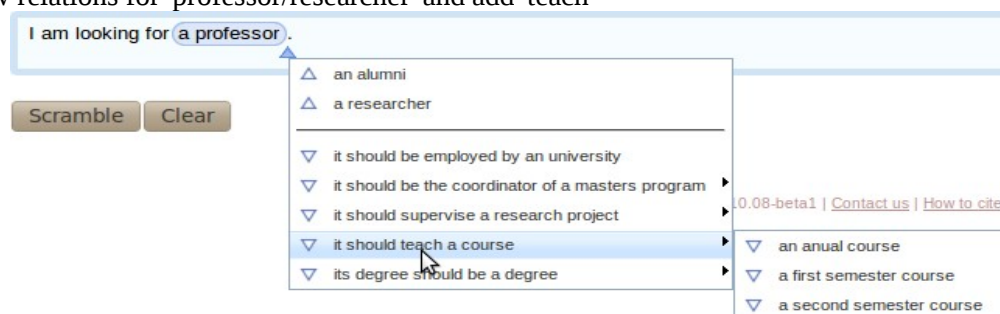
1. Show compatible concepts for 'professor' and 'researcher'



Example 2.1.2: Add relations

By clicking in the + menu we get all satisfiable properties of the instance according to the KB and the current build query.

2. Show relations for 'professor/researcher' and add 'teach'



Example 2.2: Replacement of a complex concept description by an equivalent concept.

3. I am looking for a person. It should be enrolled at an university. == I am looking for a student

Example 3: Text highlighting gives visual hints of the underlying query tree. Conceptually aligned text.

- Discontinuous highlighted text corresponds to different parts of the underlying query tree.
- Hovering on a text span associated to a node : highlights the text span corresponding to the subquery tree at that node.
- Hovering on a text span associated to an edge highlights the subquery from that edge.
- Each node of the underlying query tree has associated an add button. Hovering on the add button highlights the spans of text associated to the node owning the button.
- By clicking on a text span associated to an edge makes the edge sticky.
- Double clicking one entity in a NP highlights all NPs in the generated text which correspond to the same node in the query.

Example 4: Document planning carried out by the user

- Text planning: The underlying query tree determines the order (or the user choices when constructing the query). In terms of RST can be seen as a chain of 'elaboration' relations.
 - The document planner traverses the Quelo query.
- Document plan is a list of messages, each message comes from either a node label or an edge.

Open

Query

I am looking for a masters program . Its coordinator should be a coordinator . The masters program should include a module and it should be given in a faculty .

Open

Query

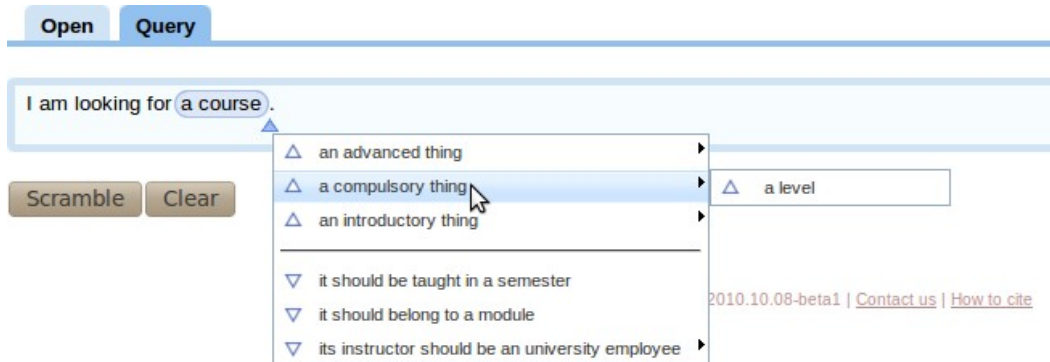
I am looking for a masters program . It should include a module and it should be given in a faculty . Its coordinator should be a professor .

Example 5: Microplanning

- Templates are mapped onto Abstract linguistic Specifications
- Tree lexicaliser generates a clause for each label in nodes and for each label of each edge outgoing a node X and entering a node Y.
- When lexicalising an edge label (edge leaving from X arriving on Y) the lexicaliser will apply the template associated to the edge label.
 - If the template is a verb-base template the subject NP is filled in with the referring NP associated to the label of node X, and the object NP is filled in with the introductory NP associated to the label of node Y.
 - If the template is a noun-based template the subject NP is constructed from the noun in the relation id which includes another NP that is filled with the NP associated to label of node X. The clause uses a copula and the subject complement is the introductory NP associated to the label of node Y.
 - Semanticisation: the microplanner also links syntactic elements to the elements of the underlying Quelo query, this provides:

- Information to the referring expression generation.
- Information to the aggregation module.
- Links between text portions and query elements.

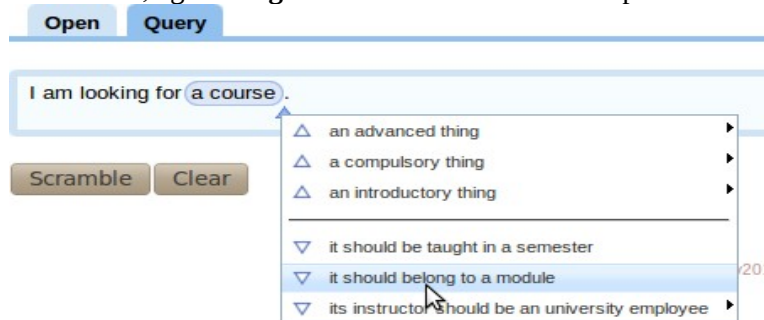
Example 5.1: Concept templates: Noun-based or Adjective-based



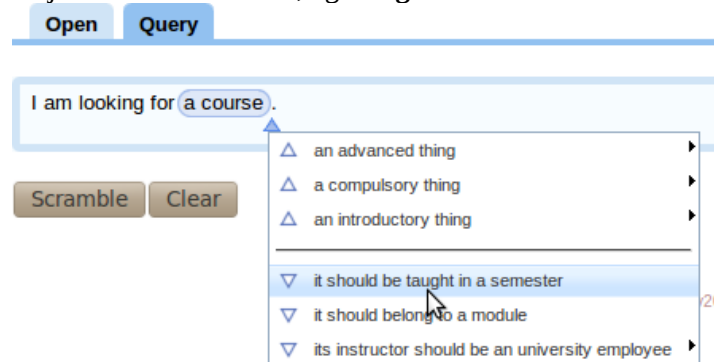
Example 5.2: Relation templates. Different relation templates for different relation types.

- Relations represented by transitive adjectives, nouns or verbs.

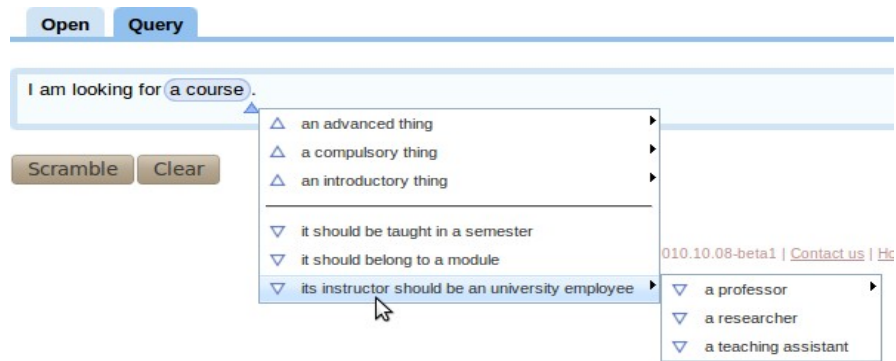
1. Verb based relation, eg. **belongsTo**. 'verb-based relation template'.



2. Transitive adjective based relation, eg. **taughtIn**. 'verb-based relation template *should be*'.



3. Noun based relation, eg. **instructor_of** or **has_instructor**. 'noun-based relation template'. The subject of the added clause (sentence) is not associated to the node of the leaving edge but to the NP concerning the noun contributed by the relation.

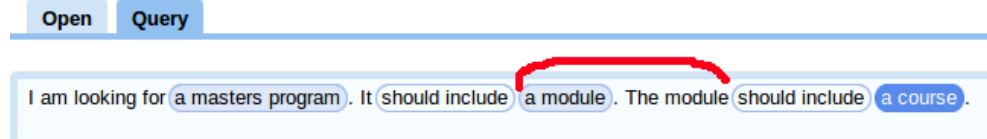


Example 6: Improving readability: aggregation and anaphoric references

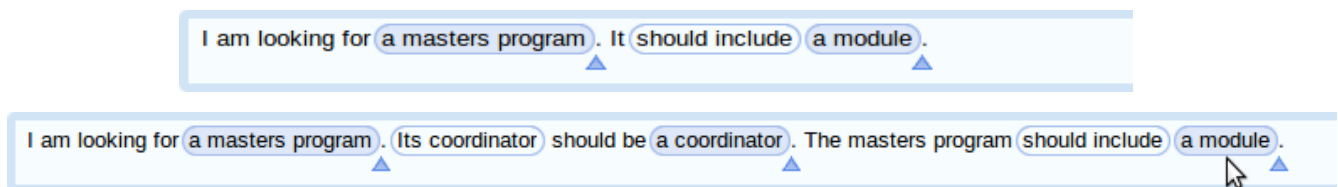
Example 6.1: REG approach

- First reference NP and subsequent references: usage of the semantic information for anaphoric references.
- NPs in noun-based template relations. NP that appears within a prepositional phrase attached to another noun phrase.
- NPs in noun-based template relations. Subject NP, generation of definite/indefinite determiner *deduced from functional restriction in the relation*. In noun-based relations such as has_model the subject of the added clause is the noun introduced by the relation. If the relation is functional the determiner used to introduce the np is definite (*the*) if the relation is not functional the determiner is indefinite (*a, some* depending if it is countable or mass noun).

Example 6.1.1: introductory NP and subsequent NPs

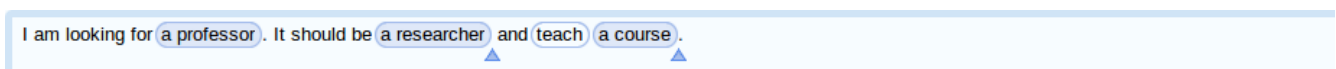


Example 6.1.2: pronouns transformed to definite NP when adding information with other subject NP in the middle.



Example 6.3: Aggregation

- Analyse the sequence of clauses obtained from a Quelo query.
- Conditions for aggregation:
 - The subjects of the subsequent clauses refer to the same node.
 - Join adjacent clauses with the same subject and same verb voice:
- 1. same subject and same verb voice:



2. coordination but not aggregated clause because of different verb voice

I am looking for a teaching assistant . It should teach a course and it should be employed by an university .

3. clause with passive verb follows a clause with a copula

I am looking for a coordinator . It should be a researcher and should be employed by an university .

4. aggregated clauses all same subject and verbs voice

I am looking for a professor . It should be the coordinator of a masters program , supervise a masters thesis and teach a course .

Demo 5: Power's prototype of Authoring Tool (prolog)

Example 1: Initial seed axiom.

1: Every thing/1 is a thing/2. (trivial axiom)

And the lexicon only contains the word *Thing*, aka **open-ended** interface meaning that the lexicon is not defined.

Example 2: Editing ontology lexicon: adding new terms for classes and properties.

(new terms in the knowledge base must be mapped to syntactic categories constrained by the CNL)

```
term(animal,noun,class) .  
term(dog,noun,class) .  
term(pet,noun,class).  
term(bone,noun,class).  
term(eat, verb, property).
```

Example 3: Authoring the knowledge base and verbalisation of suggestions.

Example 3.1: Authoring the fist SubClassOf axiom

Substitutions menu, calculated from the ontology (by reasoning service) expressed in English phrases according to the context of suggestion:

1: Every thing/1 is a thing/2. (seed axiom edition of the sub-class)

- 1: Every non/1-thing/3.
- 2: Everything that is a thing/3 and/1 a thing/4.
- 3: Everything that is a thing/3 or/1 a thing/4.
- 4: Every animal/1.
- 5: Every bone/1.
- 6: Every dog/1.
- 7: Everything that eats/1 one or more things/3.
- 8: Every pet/1.
- 9: Every thing/1.

1: Every dog/1 is a thing/2. (edition of the super-class)

- 1: A non/2-thing/3.
- 2: A thing/3 and/2 a thing/4.
- 3: A thing/3 or/2 a thing/4.
- 4: An animal/2.
- 5: A bone/2.
- 6: A dog/2.
- 7: Something that eats/2 one or more things/3. (Note: recursion on the axioms definitions)
- 8: A pet/2.
- 9: A thing/2.

1: Every dog/1 is a animal/2.

Example 3.2: Authoring another SubClassOf involving an *ObjectSomeValuesFrom* restriction.

- 2: Every thing/1 is a thing/2.
- 2: Every dog/1 is a thing/2. (editing by selecting a span denoting a class and replacing it by another term)
- 2: Every dog/1 eats/2 one or more things/3.
- 2: Every dog/1 eats/2 one or more bones/3.

- Verbalisation of the authored knowledge base axioms:

- 1: Every dog/1 is an animal/2.
- 2: Every dog/1 eats/2 one or more bones/3.

- Knowledge base generated to far:

(.tm file)

```
term(animal, noun, class).
term(bone, noun, class).
term(dog, noun, class).
term(eat, verb, property).
term(pet, noun, class).
term(thing, noun, class).
```

(.ax file)

```
subClassOf(1:term(dog, noun, class), 2:term(animal, noun, class)).
subClassOf(1:term(dog, noun, class), 2:exists(term(eat, verb, property), 3:term(bone, noun, class))).
```

Example 4: Usage of reasoning services when building the knowledge base to verify acceptability constraints (consistency and informativeness).

Choose an operation (h=help): c. [c Check consistency]

Unsatisfiable: []

Choose an operation (h=help): c/2. [c/A Check consistency for axiom A]

Axiom 2 is informative (other values redundant or inconsistent)

Example 5: Authoring a SubClassOf axiom involving an *ObjectComplementOf* class expression

4: Every thing/1 is a thing/2.

4: Every cat/1 is a thing/2.

4: Every cat/1 is a non/2-thing/3.

4: Every cat/1 is a non/2-dog/3.

Authored KB axiom:

subClassOf(1:term(cat, noun, class), 2:complementOf(3:term(dog, noun, class))).

Example 6: Authoring a SubClassOf axiom involving an *ObjectUnionOf* class expression

5: Every thing/1 is a thing/2.

5: Everything that is a thing/3 or/1 a thing/4 is a thing/2.

5: Everything that is a cat/3 or/1 a thing/4 is a thing/2.

5: Everything that is a cat/3 or/1 a dog/4 is a thing/2.

5: Everything that is a cat/3 or/1 a dog/4 is an animal/2.

Authored KB axiom:

subClassOf(1:unionOf(3:term(cat, noun, class), 4:term(dog, noun, class)), 2:term(animal, noun, class)).

Verify consistency and informativeness in this axiom:

Choose an operation (h=help): c/5.

Axiom 5 is redundant

Example 7: Looking at some of the underlying English grammar rules

**s(subClassOf(Class1, Class2)) -->
np(sub, Num, Class1),
vp(Num, Class2).**

**s(subClassOf(Class1, Class2)) -->
np(sub, Num, Class1),**

copular(Num),
np(pred, Num, Class2).

np(sub, sing, Id:unionOf(Class1, Class2)) -->
[everything], [that], vp(sing, Id:unionOf(Class1, Class2)).

Demo 6: The ACE editor

- shows all possible continuations of a partial text on the basis of a given grammar
- deterministic resolution of anaphoric references
- verb phrases close at their end all 'every' scopes that are opened within them.

Example 1: Lexicon edition, different lexicon/syntax for owl relations, hierarchy creation

There is an available lexicon (pre-defined function words and open class words which can be extended as needed). This open class words can then be used to create the ontology and assertions.

Peter is a pilot. (adding Peter)

Peter is a clever pilot. (modify to this to enter the adjective clever and note how it is modeled as an owl class)

Every pilot is a person. (hierarchy creation)

Every airline is located-in a city. (transitive adjective)

Every airline is located-in a city of a country. (city of is taken as an inverse property)

Every clever pilot holds a prize. (transitive verb)

Which airline holds a prize?

Example 2: Difference between 'A is a ...' and 'Every is a'

Every country is an area.

A country is an area.

Example 3: Authoring more complex constructions

No pilot owns an aircraft.

Every pilot is registered-at exactly 1 airline.

Every pilot is hired by an airline. (The lexicon contains each tr. verb and its participle form, automatically when used they are added as ObjectInverseOf(ObjectProperty(:hire)))

If X hires Y then X is a company. (authoring property axioms)

Example 3.1: Aggregation

Every airline is something that is a company or is an institution. (No aggregation in this case, when typing the sentence: "Every airline is something that is a company or an institution ..." at this moment the sentence is not possible to be closed by a punctuation full stop in the editor. Instead we should enter: "Every airline is something that is a company or is an institution. ...")

Every airline is something that is a company or is an institution and is located-in a city. (association in *ObjectUnionOf* and *ObjectIntersectionOf*)

Every airline is a company or is an institution and owns something that is an aircraft.

- something maps to Thing and the sentence to the following *SubClassOf* axiom:
SubClassOf(Class(:airline) ObjectUnionOf(Class(:company)
ObjectIntersectionOf(Class(:institution) ObjectSomeValuesFrom(ObjectProperty(:own)
ObjectIntersectionOf(Class(owl:Thing) Class(:aircraft))))))
 - using “ , “ corresponds to different associativity of the union and intersections, but is not available in the editor, however the text is generated when entering the class expression in Protégé. “Every airline is something that own an aircraft , and that is a company or that is an institution .”
- “Every airline is a company or is an institution that owns something that is an aircraft.” usage of 'that' or 'and' generates the same content.

Example 4: Prediction of anaphoric references

If [a pilot]_R buys an aircraft then [he]_R is rich. If a pilot is rich then he buys an aircraft. (Note also that this is the way of editing *EquivalentTo* as both *SubClassOf* s, *Tbox* edition)

[A pilot]_N is a person. [He]_N drives [a car]_R. [The car]_R is [a vehicle]_S. [It]_S is owned by an airline. (Abox edition)

Every pilot drives [an aircraft]_R from every company and does not buy [it]_R. (Note referring expression suggested for company because it is under the closed scope of every)

Note: this last example is just to show reference and every scope but does not translate to owl, it seems that prepositions are not translated. The same for intransitive verbs does not seem to be translated into owl.

Example 5: Take the generated sentences into Protégé to build the ontology.

The screenshot displays the Protégé ontology editor interface. The top menu bar includes 'ICE View', 'Active Ontology', 'Entities', 'Classes', 'Object Properties', 'Data Properties', 'Individuals', 'OWL Viz', 'DL Query', and 'OntoGraf'. The left pane shows the 'Class hierarchy' for 'airline', with a tree structure starting from 'Thing' and branching into 'country', 'airline', 'company', 'institution', 'person', 'pilot', 'star_pilot', 'prize', 'vehicle', 'aircraft', and 'car'. The right pane shows the 'Annotations' and 'Description' for the 'airline' class. The 'Annotations' tab is active, showing two annotations: 'CN_pl' with the value 'airlines' and 'CN_sg' with the value 'airline'. The 'Description' tab shows the class description: 'company or (institution and (own some (aircraft and Thing)))' and 'located-in some (Thing and (inverse (city) some country))'. The 'Members' tab shows three individuals: 'Ind263053881900447943', 'Ind333236833108906', and 'Ind481518972378393392'.

Class hierarchy: airline

- Thing
 - country
 - airline
 - company
 - institution
 - person
 - pilot
 - star_pilot
 - prize
 - vehicle
 - aircraft
 - car

Annotations: airline

Annotations

- CN_pl "airlines"^^PlainLiteral
- CN_sg "airline"^^PlainLiteral

Description: airline

Equivalent classes

Superclasses

- company or (institution and (own some (aircraft and Thing)))
- located-in some (Thing and (inverse (city) some country))

Inherited anonymous classes

Members

- Ind263053881900447943
- Ind333236833108906
- Ind481518972378393392