Natural Language Generation and Interfaces to Knowledge Bases

Claire Gardent[1], Eva Banik[2] and Laura Perez-Beltrachini[3]

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

26 June 2011, K-CAP 2011

Goals and Methods

Show how Natural Language Generation (NLG) can be used to build Natural Language Interfaces to Knowledge Bases

Demonstrate tools that exploit NLG to provide such interfaces

Why use NL to interact with KB ?

Users and domain experts have difficulty handling Description Logic (DL), OWL, RDF, etc.

Learning time

Problems with formulation:

- what is the appropriate Φ for expressing meaning M?

Problems with interpretation:

- what does Φ mean?

Problems with formulation and interpretation:

- does Φ capture the intended meaning ?

What is the DL formula for ...

How many patients between the ages of 40 and 60 when they were first diagnosed with lung cancers received radiotherapy and had a platelet count higher than 300 and a leukocytes count lower than 3?

Catalina Hallett, Richard Power, and Donia Scott.
Composing questions through conceptual authoring.
COMPUTATIONAL LINGUISTICS, 33:105–133, 2007.

What is the NL expression for ...

OWL:
class (MargheritaPizza partial
Pizza
restriction (hasTopping someValuesFrom Mozzarella)
restriction (hasTopping someValuesFrom Tomato))
Paraphrase:
Margherita pizzas have, *amongst other things*, *some* mozzarella topping and also *some* tomato topping.

Wrong or incomplete interpretations:

- All MargheritaPizza have Mozarella and Tomato
- Any pizzas having Mozarella and Tomato are MargheritaPizza
- MargheritaPizza has Mozarella and Tomato and nothing else

Owl Pizzas

Newcomers to Description Logics often ...

- expect classes to be disjoint by default
- mistakenly use universal rather existential restrictions
- mistakenly expects that "only" (allValuesFrom) implies "some" (someValuesFrom)

In general, they have difficulty understanding logical constructs

There is a need for a "pedantic but explicit paraphrase language".

Alan Rector, Nick Drummond, Matthew Horridge, Jeremy Rogers, Holger Knublauch, Robert Stevens, Hai Wang, and Chris Wroe.
Owl pizzas: Practical experience of teaching owl-dl: Common errors and common patterns.
In *14th International Conference on Knowledge Engineering and Knowledge Management*, pages 63–81, 2004.

NL versus Graphics

Natural Language is an interesting alternative to graphical interfaces (e.g., Protégé)

- No need for training: all users understand Natural Language
- Generation can provide multilingual KB interfaces
- Text can be better than graphics
  - High error rates by domain experts using graphical tools [Kim90]
  - Nested Conditional Structures easier to understand when text is used [Pet95]

Generation vs Parsing

Parsing: translates NL to a formal language e.g., DL

- ambiguous (free typing): the interpretation might not be what the user intended. Accurate Disambiguation is an open problem.
- restrictive (controlled) : the user must learn a "controlled natural language" (CNL)

Generation: verbalises DL. No ambiguity. No need to learn a language. The generator presents the possible extension of the current query.

More generally ...

Semantic Web technologies require interfaces through which knowledge can be viewed and edited without deep understanding of Description Logics

Natural Language Generation is an interesting alternative to ...

- graphical interfaces (e.g., Protégé)
- Natural Language Parsing (CNL initiative)

Tutorial Outline

Natural Language Generation (Eva Banik)

- What is NLG?
- The NLG pipeline

NLG and Natural Language Interfaces to Knowledge Bases (Claire Gardent and Laura Perez-Beltrachini)

- Verbalising Knowledge Bases
- Querying Knowledge Bases
- Authoring Knowledge Bases

Table of Contents

Table of Contents

Natural Language Generation Systems

- produce texts or speech in a human language
- input:
  - data-to-text generation:
    a non-linguistic representation of information
    - data: data about weather, medical information
    - knowledge base: ontology of diseases
  - text-to-text generation:
    a newspaper article to be summarized

The Task of Natural Language Generation

- to produce fluent, coherent natural language
- to convey *all* and *only* the information in the input
- to convey the information in a way that is easily understood, unambiguous and not misleading to the user

The task of Natural Language Generation

- The next train calls at Dundee and Perth. There are 6 trains a day from Aberdeen to Glasgow. The Caledonian Express departs at 10am. It is the Caledonian Express.

The task of Natural Language Generation

- The next train calls at Dundee and Perth. There are 6 trains a day from Aberdeen to Glasgow. The Caledonian Express departs at 10am. It is the Caledonian Express.
- There are six trains a day from Aberdeen to Glasgow. The next train is the Caledonian Express, which departs at 10am. It also calls at Dundee and Perth.

Two main stages in NLG

- Content selection: determine what to say

vs

- Content realization: determine how to say it

Content Selection: determine what to say

How do I get from Aberdeen to Glasgow?

Current time: Monday 09:40

Mon-Fri		t01	t02	t03	t04	t05	Caledonian Express
Aberdeen	d	0533	0633	0737	0842	0937	1000
Dundee	a	0651	0750	0853	0952	1052	1149
Perth	a	0714	0812	0915	1014	1114	1211
Glasgow	a	0834	0915	1014	1114	1215	1314

Content Selection: determine what to say

- Creates a set of messages from input data and other domain/background information
- Specific to the application domain
- Filter, summarize and process the input data
- Can be affected by a user model, history
- Can incorporate reasoning and planning algorithms

Content Selection: determine what to say

- Organize the input data into "messages":

```
Message-id:msg01
Relation:IDENTITY
Arguments:
 arg1:NEXT-TRAIN
 arg2:CALEDON-EXP
```

```
Message-id:msg02
Relation:DEPARTURE
Arguments:
 departing-entity:CALEDON-EXP
 departure-location:ABERDEEN
 departure-time:1000
```

```
Message-id:msg03
Relation:NUMBER-OF-TRAINS-IN-PERIOD
Arguments:
 source:ABERDEEN
 destination:GLASGOW
 number:6
 period:DAILY
```

Content Realization: The Pipeline Model

- The conversion of the input message into text
- Commonly done in a sequential manner, following a pipeline architecture
- Each module in the pipeline refines the representation passed on by the preceding module

Content Realization Pipeline

- Text Planning
- Microplanning
  - Sentence Aggregation
  - Lexicalization
  - Referring Expression Generation
- Surface Realization

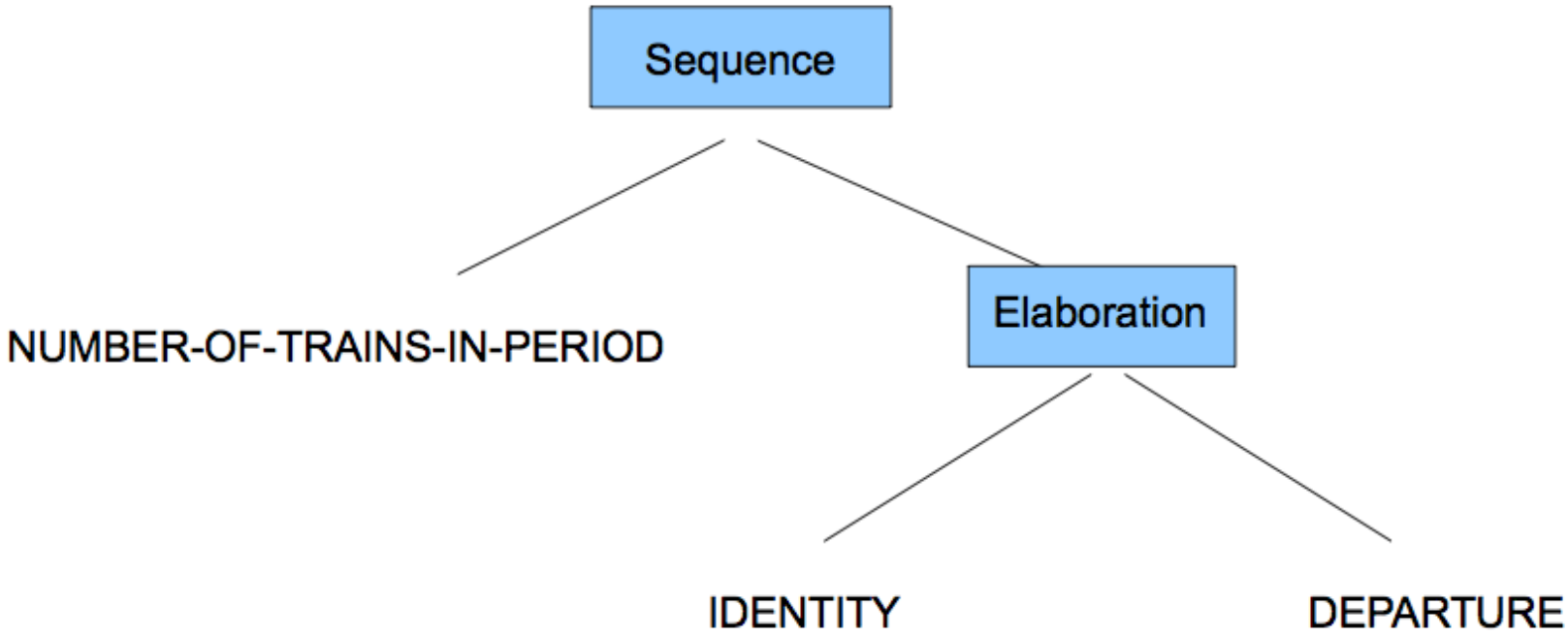
Text Planning

- Often interacts with content selection
- Imposes structure and ordering on a set of messages to form an outline for coherent text
- Text plan is a tree which corresponds to discourse structure of output text
- Leaves are sentence-sized chunks of input data
- Internal nodes are relations between sentences (discourse/ rhetorical relations, e.g., cause, contrast, sequence, elaboration)

Text Planning: Discourse Relations

- Discourse relations indicate how text fragments are related
- ELABORATION or EXEMPLIFICATION:
  - I like to collect old Fender guitars.
  - My favourite instrument is a 1951 Stratocaster.
- CONTRAST or EXCEPTION:
  - I like to collect old Fender guitars.
  - However, my favourite instrument is a 1991 Telecaster.

A Text Plan



There are 6 trains a day from Aberdeen to Glasgow.
The next train is the Caledonian Express.
It departs at 10am.

Microplanning

- Deciding what information appears on the leaves of the discourse plan tree
- Grouping sentences together if needed to avoid repetition
- Choosing the syntactic form of sentences
- Deciding where to use pronouns

Microplanning Subtasks

- Sentence Planning, Aggregation
- Lexicalization
- Referring Expression Generation

Sentence Aggregation

- Combines two or more messages together into one sentence to avoid repetition
- Takes a discourse plan and produces a new text plan whose leaves are combinations of messages
- Doesn't change the information content of the text but contributes to fluency and readability.

There are 6 trains a day each day from Aberdeen to Glasgow.
The next train is the Caledonian Express.
The Caledonian Express departs at 10am.

Types of Aggregation

- **Embedding**
  The next train, *which leaves 10 am,* is the Caledonian Express.
- **Ellipsis**
  The Caledonian Express leaves from Euston and ~~the Caledonian Express~~ terminates in Glasgow.
- **Set formation**
  The Caledonian Express calls at Dundee. The Caledonian Express calls at Perth. $\Rightarrow$
  The Caledonian Express calls at Dundee and Perth.

Where does aggregation take place?

- Aggregation of semantics: changes the input messages

```
Relation:  CALL-AT
    Arg1:  Caledon-Exp
    Arg2:  Perth + Dundee
```

- Aggregation of sentence plans: changes how individual messages are realized

```
Relation:  IDENTITY
    Arg1:  Next-train
    Arg2:  Caledon-Exp
  Syntax:  main clause
```

```
Relation:  DEPARTURE
    Arg1:  10am
    Arg2:  Caledon-Exp
  Syntax:  relative clause
```

Lexicalization

- Choosing words to express concepts and relations
- Different word choices can result in different style of text (e.g., formal/informal), added variety of texts, different levels of readability, or text in a different language

    - The Caledonian Express leaves Aberdeen at 10am.
    - The Caledonian Express departs from Aberdeen at 10 in the morning.

Referring Expression Generation

- Generate descriptions which enable the user to unambiguously identify the intended entity
- Choose the correct form of referring expression based on discourse context
- *The next train* is the *Caledonian Express*. *It* leaves at 10am. Many tourist guidebooks highly recommend *this train.*

Referring Expression Generation

- Types of referring expressions:
  - Definite descriptions (the train that leaves at 10am)
  - Names (Caledonian Express)
  - Pronouns (it)
- Referring expressions are important for text fluency:
  - It departs at 10am. The Caledonian Express is the next train.
  - The Caledonian Express departs at 10am. The next train is it.
- Choice of referring expressions can introduce ambiguity:
  - The Caledonian Express and the Glasgow express both go to Glasgow. It is the next train.

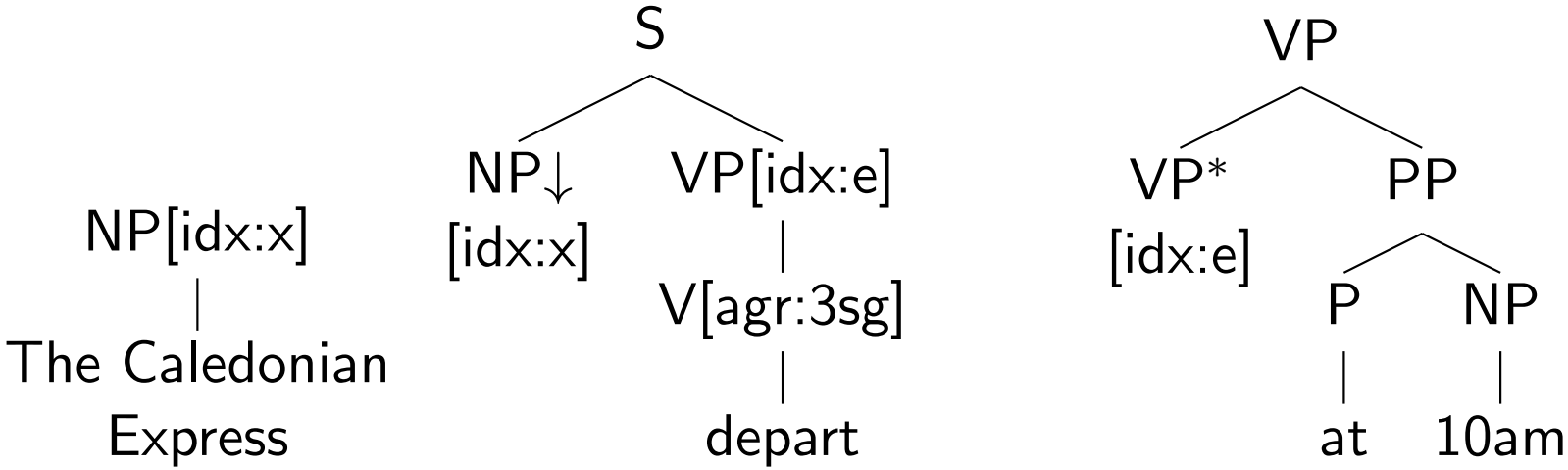
Surface Realization

- Encodes knowledge about the grammar of the target language
- Generates the correct grammatical form for words in sentences:
  - Subject-verb agreement
  - Correct auxiliary for present/past/future tense
  - Syntactically required pronominalization (generate correct forms for pronouns – him/her/himself etc)

Surface Realization

Arg1:	Relation:	Arg2:
Caledon-Exp	DEPARTURE	10am(e)

Surface Realization

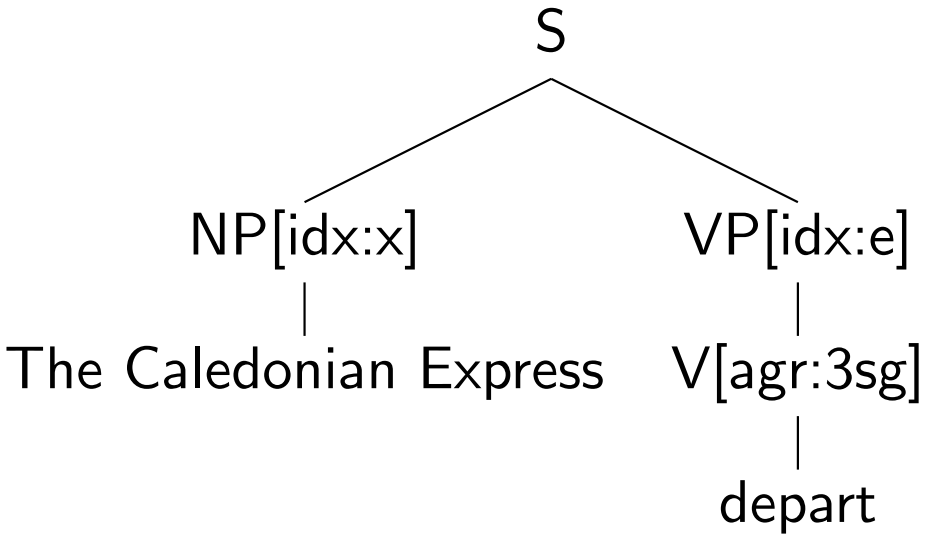


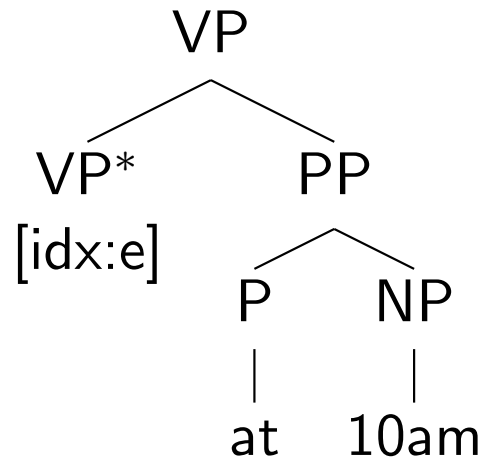
`Arg1:`
`Caledon-Exp`

`Relation:`
DEPARTURE

`Arg2:`
`10am(e)`

Surface Realization



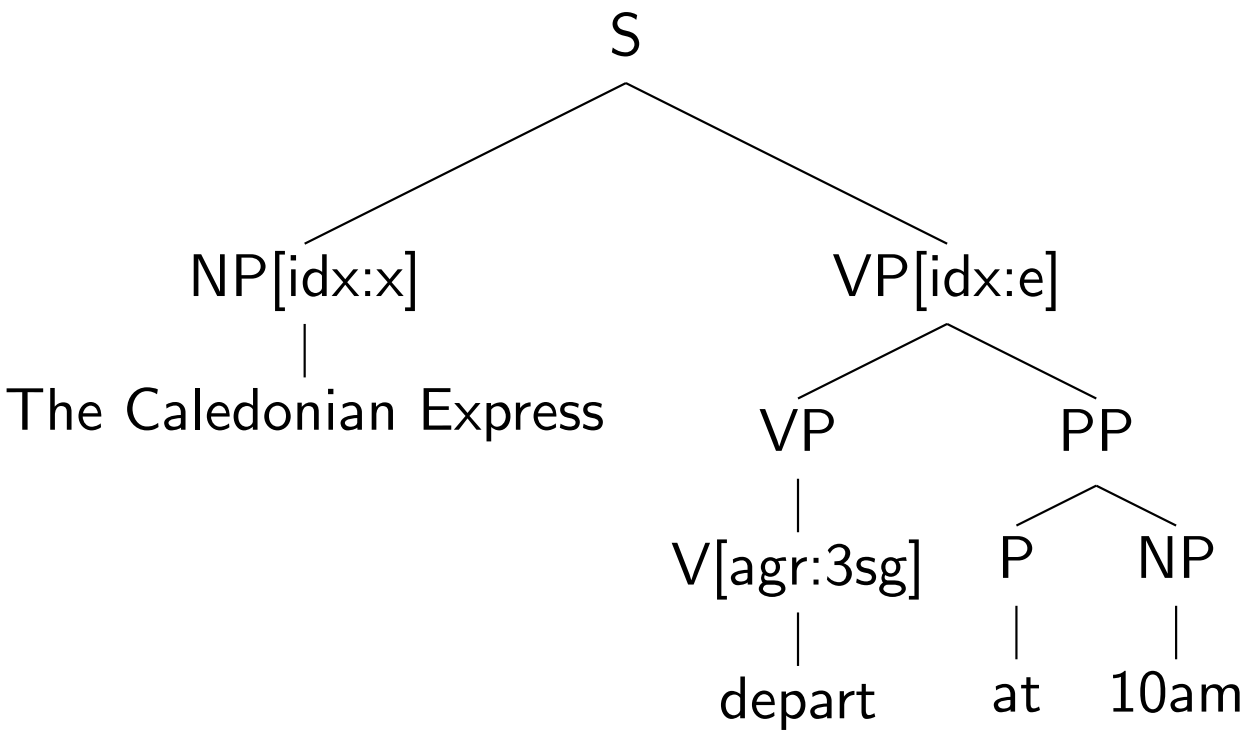


`Arg1:`
`Caledon-Exp`

`Relation:`
DEPARTURE

`Arg2:`
`10am(e)`

Surface Realization



The Caledonian Express departs at 10am.

`Arg1:`	`Relation:`	`Arg2:`
`Caledon-Exp`	DEPARTURE	`10am(e)`

Overview

Content Selection

Message01:IDENTITY

Message02:DEPARTURE

Message03:NUM-OF-TRAINS

Content Realization

Text Planning → Discourse Tree

Microplanning:→ Sentence Plan

Sentence Aggregation

Lexicalization

Referring Expression Generation

Surface Realization → Output Text

Further Reading

- Ehud Reiter and Robert Dale (2000): Building Natural Language Generation Systems. *Cambridge University Press.*
- Ehud Reiter and Robert Dale (1997): Building applied natural language generation systems. In: *Journal of Natural Language Engineering*, 3:1 pp.57–87.
- Ehud Reiter (1994): Has a consensus NL generation architecture appeared, and is it psychologically plausible? In: *Proceedings of the 7th. International Workshop on Natural Language generation,* pp 163–170.
- Mike Reape and Chris Mellish (1999): Just what *is* aggregation anyway? In: *Proceedings of the 7th European Workshop on Natural Language Generation*, pp 20–29.

Table of Contents

Verbalising Ontologies: What is it ? Why is it useful?

Ontology verbalisers are used to

- document knowledge bases (cf. Protege Plugin)
- produce reports from ontologies (e.g., concept definitions, individual descriptions)
- provide multilingual descriptions of an ontology

Ontology verbalisers facilitate Man-Machine Interaction

- they make ontology accessible to non experts
- they avoid misunderstandings resulting from a poor understanding of the meaning of DL expressions (Interpretation issue)

Verbalising Ontologies: Is it possible? How difficult is it?

Is there a natural, "simple" mapping between the syntax of KR languages and the syntax of Natural Language ?

Is there is a "simple" mapping between the terms of DL and the words of NL?

Defining the target syntax

A Controlled Natural Language (CNL) is a well-defined subset of English restricted wrt both syntax and lexicon

The OWL1-1 task force aims to link OWL to a CNL

OWL CNLs include ACE (Attempto Controlled English), PENG (processable English), SOS (Sydney OWL Syntax), Rabbit, CLOnE [ST04, KF07].

Defining the syntax mapping (ACE CNL)

OWL properties and classes	Examples of corresponding ACE verbs and noun phrases
Named property	*Transitive verb, e.g.* like
InverseObjectProperty(R)	*Passive verb, e.g.* is liked by
Named class	*Common noun, e.g.* cat
owl:Thing	something; thing
ObjectComplementOf(C)	something that is not a car; something that does not like a cat
ObjectIntersectionOf($C_1 \ldots C_n$)	something that is not a cat and that owns a car and that ...
ObjectUnionOf($C_1 \ldots C_n$)	something that is a cat or that is a camel or that ...
ObjectOneOf(a)	*Proper name, e.g.* John; something that is John
ObjectSomeValuesFrom(R C)	something that likes a cat
ObjectExistsSelf(R)	something that likes itself
ObjectMinCardinality(n R C)	something that owns at least 2 cars
ObjectMaxCardinality(n R C)	something that owns at most 2 cars
ObjectExactCardinality(n R C)	something that owns exactly 2 cars

Applying the mapping (Verbalising)

$\mathsf{CAT} \sqcap \neg\exists\ \mathsf{like}.(\mathsf{DOG} \sqcap (\exists\ \mathsf{attack}.\mathsf{MAILMAN} \sqcup \mathsf{FIDO}))$

$\mathsf{CAT} \sqcap$	a cat
$\neg\exists\ \mathsf{like}.(\mathsf{DOG} \sqcap$	that does not like a dog
$(\exists\ \mathsf{attack}.\mathsf{MAILMAN} \sqcup$	that attacks a mailman or
$\mathsf{FIDO}))$	that is Fido

The "Consensus Model"

The consensus model [Pow10] assumes that there is a simple deterministic mapping between DL and CNL and more specifically, that:

- Atomic terms (individuals, classes, properties) map to words
- Axioms map to sentences (one sentence per axiom)

Does the Consensus Model hold in practice ?

Theoretically, KR languages do not guaranty that the Consensus Model holds because:

- Terms are unstandardised and thus do not necessarily map to words
- OWL classes and axioms can be arbitrarily complex and therefore do not necessarily map to sentences

Do existing ontologies validate the consensus model ?

Mapping terms to words: Is it always possible to find a suitable lexical entry for atomic terms?

[MS05, Pow10] show that out of 882 ontology files coded in OWL (111Mb):

- only 14% of the class names contain no recognised word (using WordNet as a lexicon)
- 72% of the class names ended with recognised nouns
- 30% of the class names consisted entirely of noun strings

Most atomic terms can be assigned a CNL expression by mapping their components to the corresponding words

Mapping terms to words: How complex are terms?

[MS05, Pow10] show that out of 48 ontologies totalling around 45 000 axioms and 25 000 atomic terms, the number of words contained in an atomic terms varies between one and 4 (*Beaujolais Région, ABI graph plot, etc.*).

Most atomic terms can be lexicalised

Mapping axioms to sentences: Is it always possible to describe the content of an axiom by a sentence?

[Pow10] examine the axiom patterns in the TONES[1] ontology repository (214 files containing up to 100726 axioms) and show that most axioms follow a simple logical pattern

Pattern	Frequency	Percentage
$C_A \sqsubseteq C_A$	18961	42.3%
$C_A \sqcap C_A \sqsubseteq \bot$	8225	18.3%
$C_A \sqsubseteq \exists P_A.C_A$	6211	13.9%
$[I, I] \in P_A$	4383	9.8%
$[I, L] \in D_A$	1851	4.1%
$I \in C_A$	1786	4.0%
$C_A \equiv C_A \sqcap \exists P_A.C_A$	500	1.1%
Other	2869	6.4%
Total	44786	100%

Most axioms can be described by a CNL sentence

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

Summing Up

The restricted syntax of DL makes it possible to define a mapping from DL to a CNL

In theory though, because terms are unstandardised and axioms may be arbitrarily complex, there is no garantee that using this mapping will yield understandable text

In practice however, terms mostly can be mapped to words or phrases and axioms to sentences because (i) terms (or their labels) contain words and (ii) most axioms are simple. Although, as the OWL ACE verbaliser demo will show, the generated text is not always well formed.

Demo 1: the OWL ACE Verbaliser

The OWL ACE Verbaliser implements the consensus model to verbalise OWL axioms

It is available as a plug in for Protégé

Demo:
`http://attempto.ifi.uzh.ch/site/docs/owl_to_ace.html`
The demos shows simple cases with correct output but also problems related to morphology ("readsed", "driveses") and to the verbalisation of properties of properties, of cardinality, of equivalence and of complex axioms.

Producing text rather than sentences

Generating Text from Knowledge Bases

Sometimes, generating one sentence per axiom is enough e.g., to "read" an axiom. Other times however it is not sufficient and generating text is required e.g.,

- to produce short, readable documentation of an ontology [WP10, Pow11]
- to produce a coherent description for an individual (verbalising A-Box content) e.g.,

The MIAKT system generates reports aimed at medical professionals from a medical ontology describing patient information [Bon05]
The NaturalOWL system produces descriptions of instances (e.g., museum exhibits) and classes from OWL DL ontologies [GA07]

Verbalising sets of axioms

Verbalising one sentence per axiom will often result in disorganised lists including inefficient repetitions rather than text. E.g.,

$$CAT \sqsubseteq ANIMAL$$
$$DOG \sqsubseteq ANIMAL$$
$$HORSE \sqsubseteq ANIMAL$$
$$RABBIT \sqsubseteq ANIMAL$$

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

Aggregation can help minimise redundancy and repetition.

[WP10] show that all axioms patterns in EL++ can be *aggregated* without further domain knowledge .

Example

$CAT \sqsubseteq ANIMAL$	$\textsc{subClassOf}(\textsc{class}(\textsc{cat}),\textsc{class}(\textsc{animal}))$
	Every cat is an animal.
$DOG \sqsubseteq ANIMAL$	$\textsc{subClassOf}(\textsc{class}(\textsc{dog}),\textsc{class}(\textsc{animal}))$
	Every dog is an animal.
$HORSE \sqsubseteq ANIMAL$	$\textsc{subClassOf}(\textsc{class}(\textsc{horse}),\textsc{class}(\textsc{animal}))$
	Every horse is an animal.
$RABBIT \sqsubseteq ANIMAL$	$\textsc{subClassOf}(\textsc{class}(\textsc{rabbit}),\textsc{class}(\textsc{animal}))$
	Every rabbit is an animal.

Aggregation:
$\textsc{subClassOf}([\textsc{class}(\textsc{cat}),\textsc{class}(\textsc{dog}),\textsc{class}(\textsc{horse}),$
$\textsc{class}(\textsc{rabbit})],\textsc{class}(\textsc{animal}))$

Realisation: The following are kinds of animals: a cat, a dog, a horse and a rabbit.

In real life

Given the axiom pattern $A \sqsubseteq B$, $A \sqsubseteq \mathrm{C}$, $B \sqsubseteq C$ and $C \sqsubseteq D$, three aggregation operations are possible :

left-hand-side merge (L)	$[A, B] \sqsubseteq C$
right-hand-side merge (R)	$A \sqsubseteq [B, C]$
chaining (C)	$A \sqsubseteq B \sqsubseteq C \sqsubseteq D$

4 axiom patterns account for 96% of all patterns found in around 35000 axioms namely: $A \sqsubseteq B$ (51%), $A \sqsubseteq \exists P.B$ (33%), $[a, b] \in P$ (8%), $a \in A$ (4%).

Only some axiom pattern/aggregation operation combinations make sense linguistically. The safest are those grouping axioms with the same pattern.

The generation procedure

Aggregation: axiom patterns are aggregated.

Surface realisation: single and aggregated axioms are turned into sentences using a lexicon to map terms into words and a grammar to map axioms into sentences

- Lexicon: built automatically from the identifier names and labels.
  Classes as nouns, properties as verbs with valency two and individuals as proper names.
- Grammar: rules for realising single and aggregated axioms.
  For single axioms, reuse the proposal from the OWN CNL task force [ST04]

Example verbalisations

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

Testing aggregation

Applied to a sample of around 50 ontologies.

Aggregation reduces the number of generated sentences and increases sentence length

Unit	Original	Aggregated	Reduction
Sentences	35542	11948	66%
Words	320603	264461	18%

Remarks

- Some sentences are very long (800 instances of class Island) and would be better expressed using a table
- Some axioms participate in several merges thereby yielding some redundancies e.g., *The following are men: Fred, ...* and *Fred is all of the following: a man, ...*

Summing Up

Aggregation can help minimise redundancy and repetitions

The approach proposed in [WP10] is domain independent and can be applied to any knowledge base because it is based on the logical structure of axioms

Structuring the presentation of Knowledge Bases

[Pow11] investigates how discourse relations can be used to produce *coherent text* i.e., text that makes explicit the rhetorical/discourse relations between statements e.g.,

Every corgi is an animal. Every corgi is a dog. $\Rightarrow$

- Every corgi is all of the following: an animal and a dog. [Pow11]
- Every corgi is an animal and more specifically, a dog. [WP10]

Structuring the presentation of Knowledge Bases

The SWAT system has different presentation strategies depending on the axiom types:

- Equivalence axioms translates to definitions
  E.g., *A grownup is defined as an adult that is a person.*
- Property axioms translates to descriptions
  E.g., *Joe has as pet Fido*
- Subclass axioms translate to examples or to "Typology"
  E.g., *A giraffe is an animal*
- Class assertions translate to examples (class entry) or to typology statements (individual entry)
  E.g., *Minnie is an elderly*

Demo 2: the SWAT Verbaliser

`http://swat.open.ac.uk/tools/`

The SWAT verbaliser provides the following functionalities for OWL ontologies:

- Simple statistics on the ontology (e.g., axiom pattern frequency)
- Lexicon construction from an ontology
- Translation to Prolog
- On sentence per axiom translation
- Ontology documentation

Describing individuals: the MIAKT system

Automatic generation of reports from ontologies

Given an ontology describing the breast cancer domain encoded in DAML+OIL and a case description (patient information, medical procedures, mammograms, etc.) encoded in RDF, the MIAKT system generates a textual description of each case.

The MIAKT NLG Pipeline

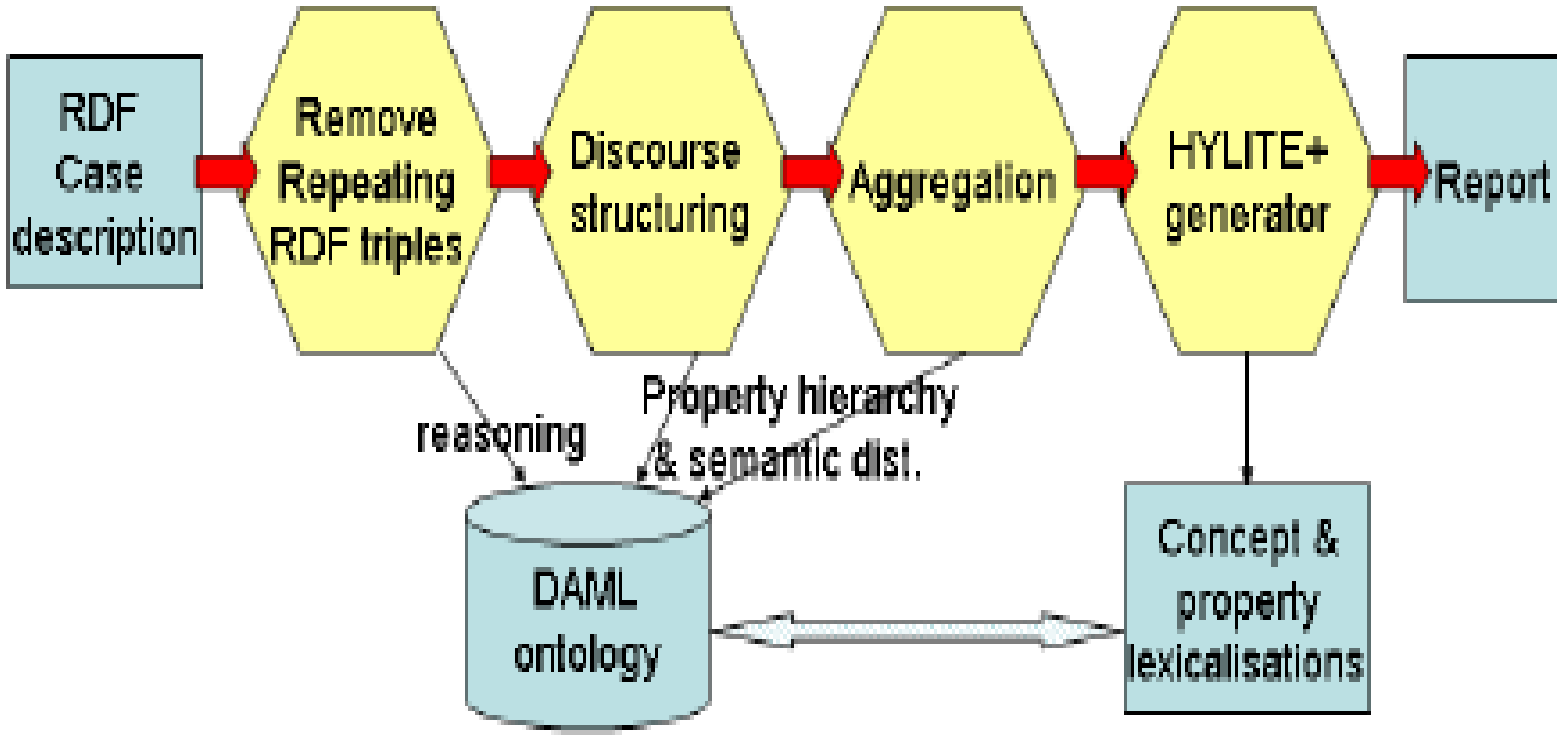
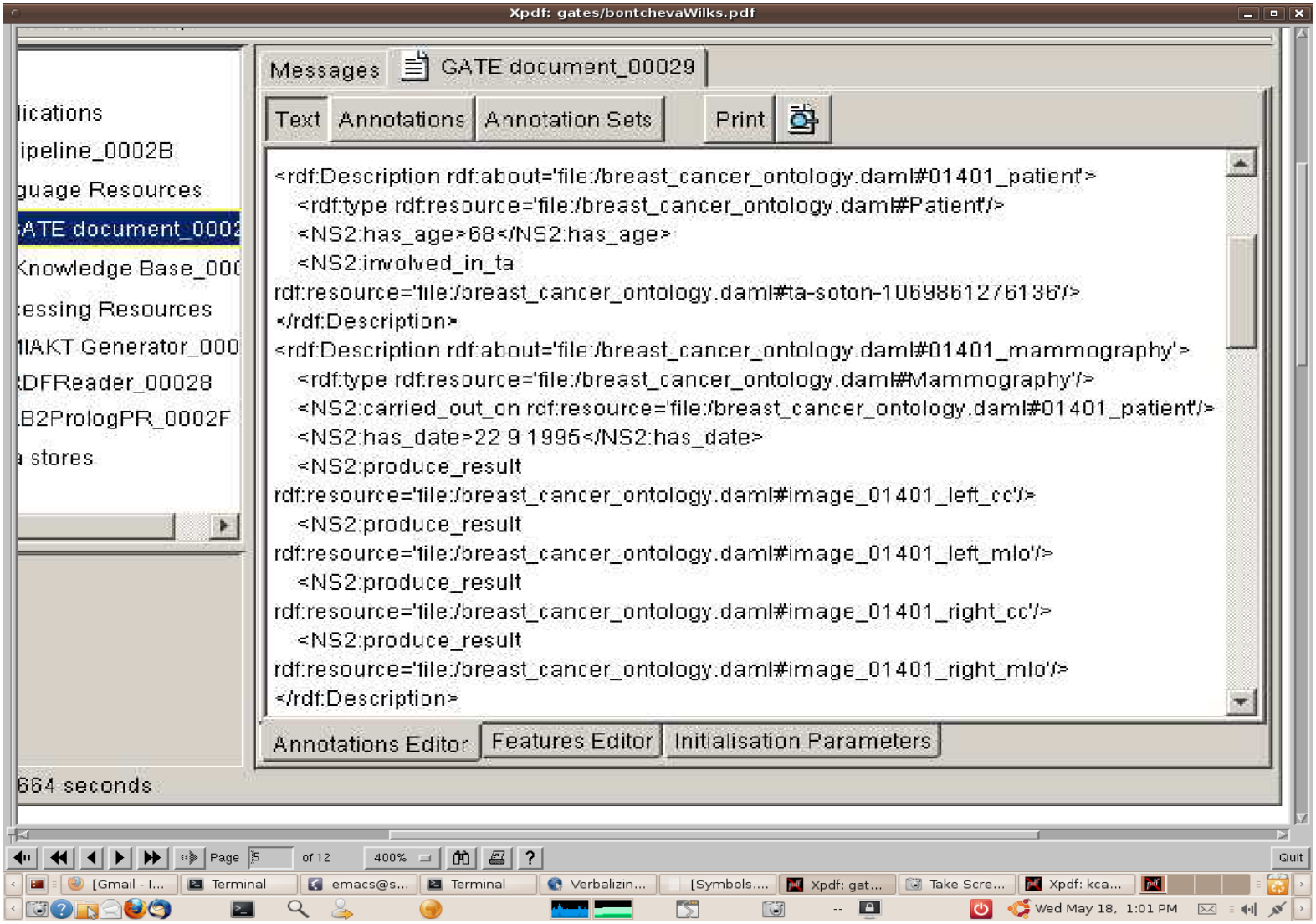
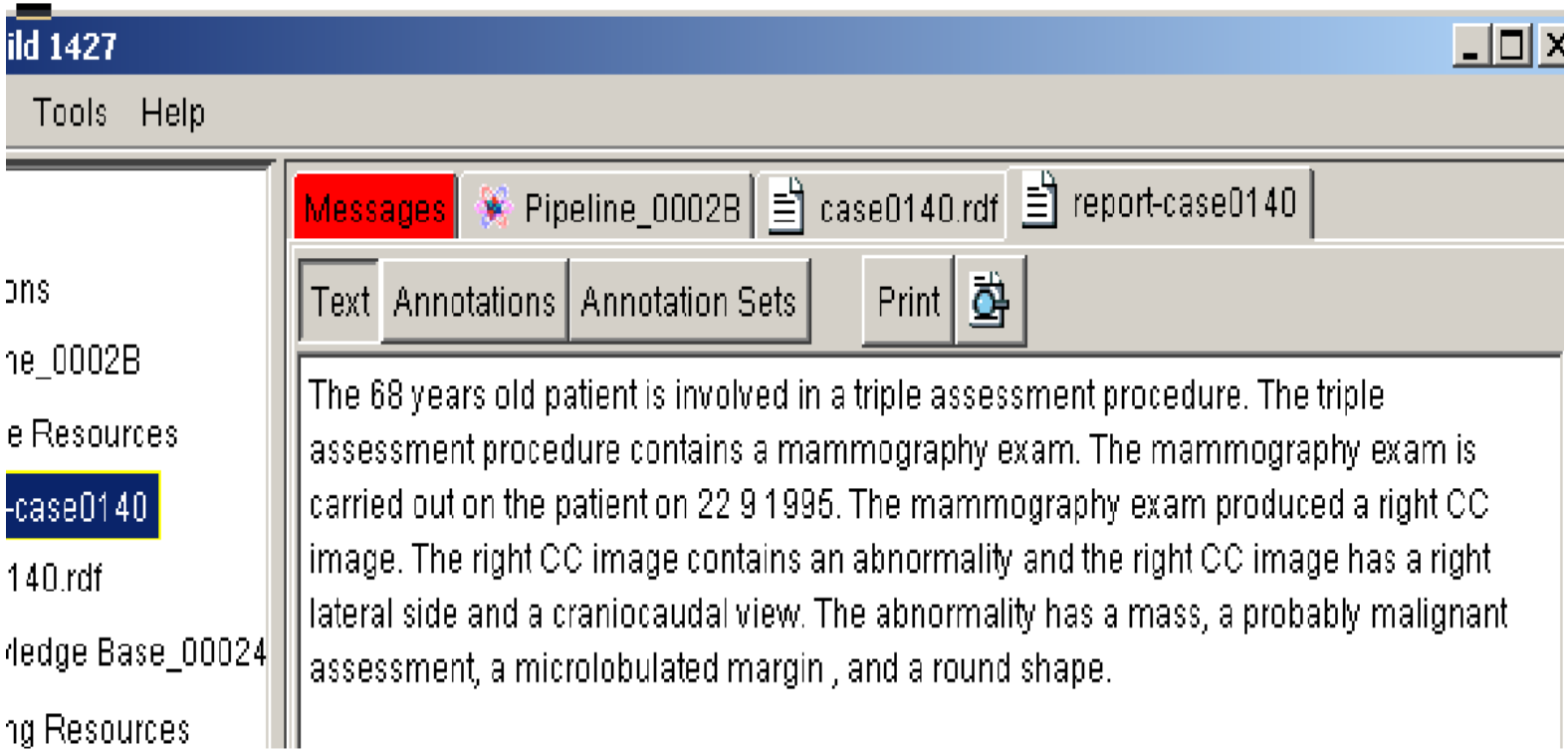


Fig. 1. The MIAKT Generator

Example MIAKT input



Example MIAKT output



Removing redundancies

RDF case descriptions are created by medical professionals. They sometimes contain repetitive information due to the use of inverse relations e.g.,

```
involved_in_ta(01401_patient, ta-106986127613 6)
involved_patient(ta-106986127613 6, 01401_patient)
```

Removing implied information i.e., information which logically follows from other facts could also be done by using a reasoner and axioms from the ontology.

Discourse planning

The input (RDF triples) is unordered. The output text however typically starts off by describing the patient then move on to the medical procedures and their findings.

Discourse patterns are applied recursively by the discourse structuring algorithm.

```
Describe-Patient ->
   Patient-Attributes,
   Describe-Procedures1

Patient-Attributes ->
% collects all properties that are sub-properties
% of the attribute-property
   [attribute(Patient, Attribute)],
   Patient-Attributes *
```

Aggregation

At the discourse level, merge adjacent triples which share the 1st argument and have the same property name.

```
ATTR(Abnormality: 01401_abnormality, Mass: 01401_mass)
ATTR(Abnormality: 01401_abnormality, Margin: inst_margin_microlob)
ATTR(Abnormality: 01401_abnormality, Shape: inst_shape_round)
ATTR(Abnormality: 01401_abnormality, Diagnose: inst_ass_prob_malig)
```

The abnormality has a mass. The abnormality has a microlobulated margin. The abnormality has a round shape. The abnormality has a probably malignant assessment.

$\Rightarrow$ *The abnormality has a mass, a probably malignant assessment, a microlobulated margin and a round shape.*

Describing entities and classes: NaturalOWL

NLG engine that produces descriptions of entities (museum exhibits) and classes (types of exhibits) in English and Greek from OWL DL ontologies

Ontologies must be annotated with linguistic and user modeling annotations

A protégé plug-in can be used to create these annotations and to generate previews of the resulting texts by invoking the generation engine

The NaturalOWL annotations

In NaturalOWL, classes and properties are annotated with various types of information

- words and phrases which indicates how to verbalise classes
- micro-plans indicating how to verbalise properties
- a partial order on properties used in document planning to order facts and produce a coherent text
- interest scores indicating how interesting a given fact is to each user type
- parameters that control the length of the output text

Why annotate ontologies with lexical annotations?

- Terms must be mapped to both English and Greek
- Words are usually ambiguous
- Automatic translation will yield all possible translations of all possible meanings

NaturalOWL bypasses these issues by annotating classes and properties with appropriate words/phrases.

NaturalOWL architecture

Document planning: selects the logical facts (OWL triples) that will be conveyed to the user

Document structuring: orders the selected facts so as to produce a coherent, fluent text

Micro-planning; each selected fact is turned into a sentence using the micro-plan and the words/phrases annotating the classes and properties present in that fact

Document Planning in NaturalOWL

1. First select facts that are directly relevant to the instance being described
   - Takes into account interest scores of selected facts and
   - a dynamically updated user model showing what has already been conveyed to the user: fact that reports (dis)similarity to previously mentioned entity may be included to output comparisons
2. a distance parameter can be set which permits including facts that are further away in a graph representation of the ontology
3. the selected facts are ordered using a manually specified partial order on facts

Microplanning in NaturalOWL

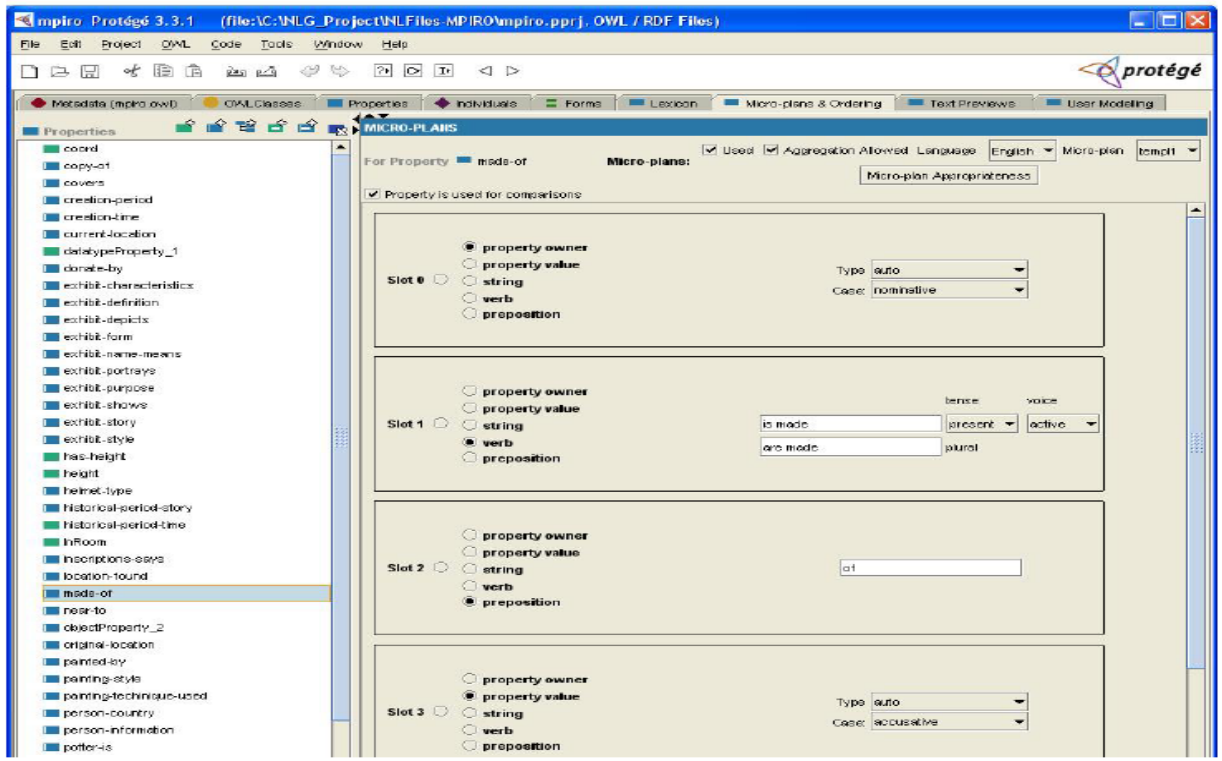
- Each selected fact is turned into a sentence using the microplan annotating this fact
- A microplan is a pattern that leaves referring expressions underspecified
- Referring expressions are generated taking into acount the context of each sentence to avoid repetition and ambiguity
- Aggregation rules combine the resulting sentences into longer ones

Example microplan

To express a fact that involves the made-of property:

- concatenate an automatically generated referring expression (e.g., name, pronoun, definite noun phrase) in nominative case for the owner of the fact (semantic subject of the triple),
- the verb form "is made" (or "are made", if the subject is in plural),
- the preposition "of",
- and another automatically generated referring expression in accusative case for the filler of the property

Using the Protégé plug-in to specify a microplan



Demo 3: NaturalOWL

The NaturalOWL NLG system can be downloaded at
`http://www.aueb.gr/users/ion/publications.html`

The demo illustrates the annotation graphical interface for enriching the ontology with lexicalisation, microplans and other parameters. We also show how different parameter setting impact the generated text.

Table of Contents

Querying Ontologies

NLG can be used to support the user in the task of formulating a query over a knowledge base.

- NL suggestions for queries are **generated** from the ontology
- The user can edit (add, delete, substitute) this suggestion by modifying the NL query generated by the system
- Modifying the NL query modifies the DL query
- Conceptual Authoring [HPS07] is used to link text and KB content so that modifying the text means modifying the query

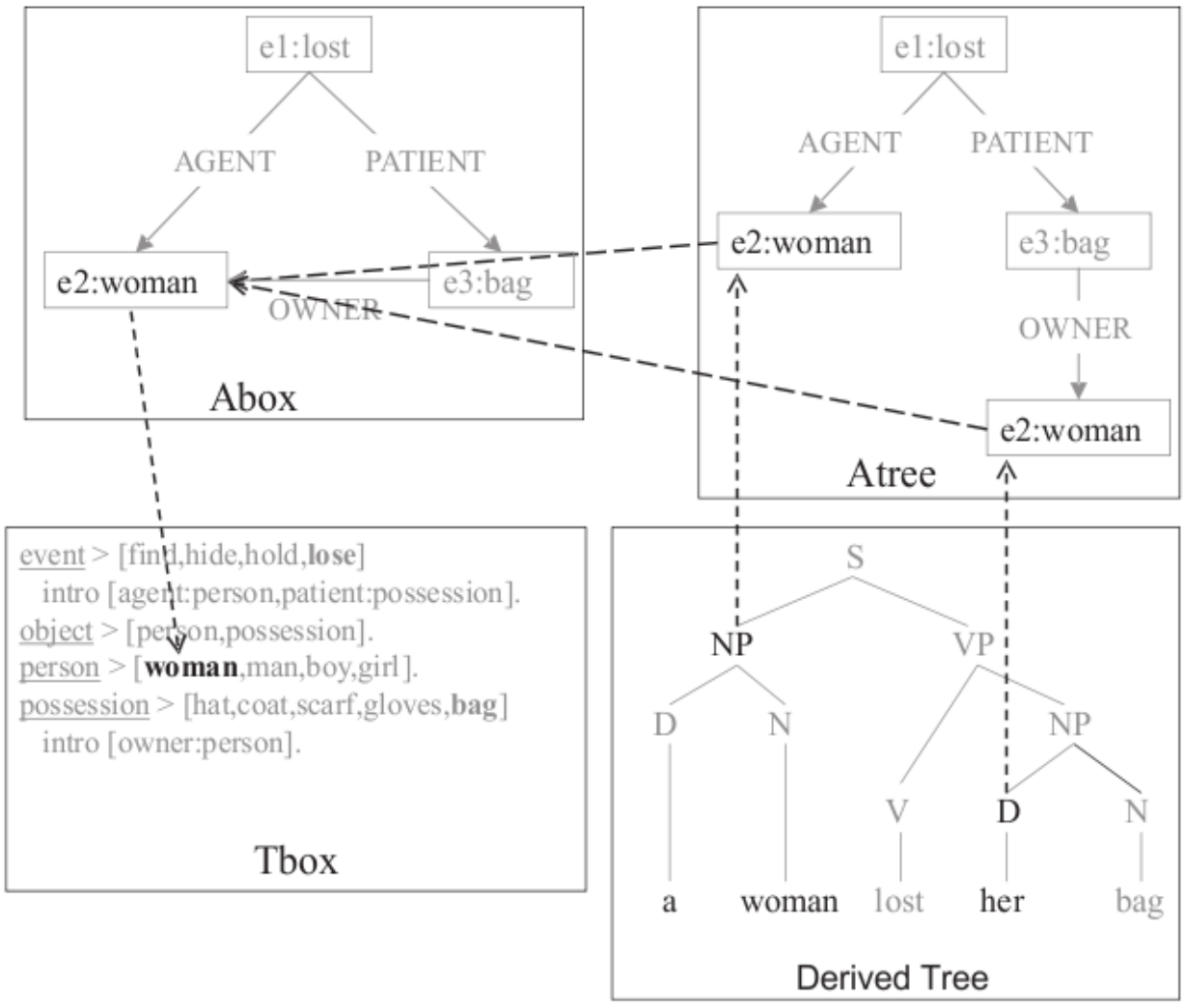
Conceptually aligned text

Conceptual authoring relies on conceptually aligned text [HP08]
i.e., on a systematic bidirectional linking between text and ontology

[HP08]'s WYSIWYM system
(`http://mcs.open.ac.uk/nlg/old_projects/wysiwym/`)
posits 3 related structures:

- A graph based representation of a set of DL assertions (A-box)
- An A-tree, which describes a syntactic realisation of the A-box
- A syntactic tree whose yield provides the output sentence

A-Box, A-Tree and Syntactic Tree



Constructing the A-Tree

Mappings: Mappings are defined between concepts and A-Tree fragments (nodes and immediate dependents). The Atree provides a syntactic context used to constrain the mapping search for each child

Search: Generation searches and applies mappings top-down from the root of the Abox.

Determinism: if multiple mappings are possible, the first one is chosen. If no mapping is found, the system fails

Completeness: a static test can be run to determine if any mappings are unused or missing.

Example Mapping

Concept	C938B actor(person) actee(person) target(fact)
Mapping	frame concept=C938B bindings=SUB,DOB,CL_COM gr_key=Tnx0Vnxs2, lemma=remind pos=verb

Use “remind” verb with 3 arguments where the first argument is a nominal subject (realising the actor), the second a nominal object (realising the actee) and the third, a clause realising the target fact.

Querying with QUELO

- The user defines a query by editing a sentence generated by Quelo.
- She can add, delete, substitute or weaken any of the text snippets highlighted in that sentence by the system
- Each text snippet is related both to the formula and to the Quelo query (as in the conceptual authoring approach)
- Snippets available for edition are filtered using automated reasoning: they must be consistent and informative wrt the query and the ontology.

QUELO's query model

Quelo's query model is a data structure linking tree-shaped conjunctive DL queries and text.

$Woman \sqcap Singer \sqcap \exists parent.(Actor) \sqcap \exists spouse.(Man \sqcap \exists parent.(Singer))$

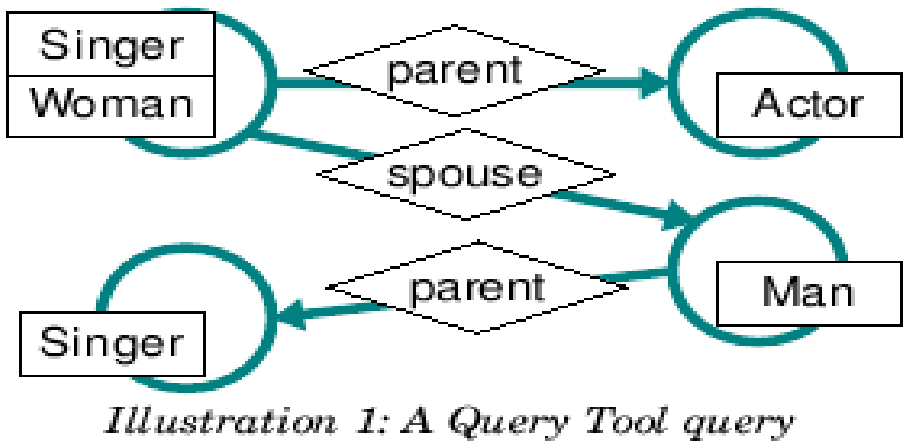


Illustration 1: A Query Tool query

Figure: The Query Model

I am looking for a female singer mother of an actor married to the father of a singer

QUELO's operations on the query model

ADDCOMPATIBLE(NODE,CONCEPT): adds CONCEPT to NODE provided CONCEPT does not generalise, specialise, is equivalent to or incompatible with the concept associated with node

ADDPROPERTY(NODE,RELATION,CONCEPT): adds a new edge to a node labeled withRELATION and pointing to a node labeled withCONCEPT. Only add edges that are compatible with the current query.

SUBSTITUTE(SELECTION,CONCEPT): if SELECTION is a node label, replaces it with CONCEPT. Else, replaces the node or the entire substree with the concept. The available concepts must generalise, specialise or be equivalent to the substituted concept.

DELETE(SELECTION): deletes SELECTION (under certain conditions)

QUELO's NLG system

Document planning (content selection and ordering) is carried by the user.

Microplanning includes

- Lexicalisation
- Surface Realisation
- Aggregation
- Referring Expression Generation

Lexicalisation and Surface Realisation

For each node,

- Nodes are mapped onto NPs
- Edges are mapped onto clause templates
- Lexicalisation varies depending on whether the concept is noun- or adjective-based and whether a property is verb- (*sell* or noun- (*mother*) based
  - I am looking for a shop selling shirts
  - The mother of the actor should be ...

Aggregation in QUELO

Joins adjacent clauses with the same subject and the same voice (active, passive) into a single sentence

(1) The firm **should** manufacture goods.
The firm **should** provide services.
The firm should manufacture goods and provide services.

If two adjacent clause share a verb group, then delete this verb group in the second sentence.

(2) The actor **should be** a singer.
The father **should be** a boy.
The actor should be a singer and the father a boy.

Referring Expressions in QUELO

Replace noun phrases used as subsequent references with more appropriate expressions
Subsequent references to the same node are realized as definite descriptions or pronouns.
The actor should be married to **a singer**. **The singer** should be Muslim. The actor should live in an Indian city.

Referring Expressions in QUELO

When a subsequent mention appears within a prepositional phrase, the prepositional phrase is replaced by a possessive determiner.

(3) The mother of the singer should be an Indian.
$\Rightarrow$ His mother should be an Indian.

Ambiguity remains an issue:

(4) The girl$_1$ should live with a woman$_2$. She$_?$ should like strawberries

Demo 4: QUELO

`http://krdbapp.inf.unibz.it:8080/quelo/`, QUELO [EFT10]: A query tool integrating a conceptual authoring approach and restricting query extension so that they are logically consistent with the context and with the ontology

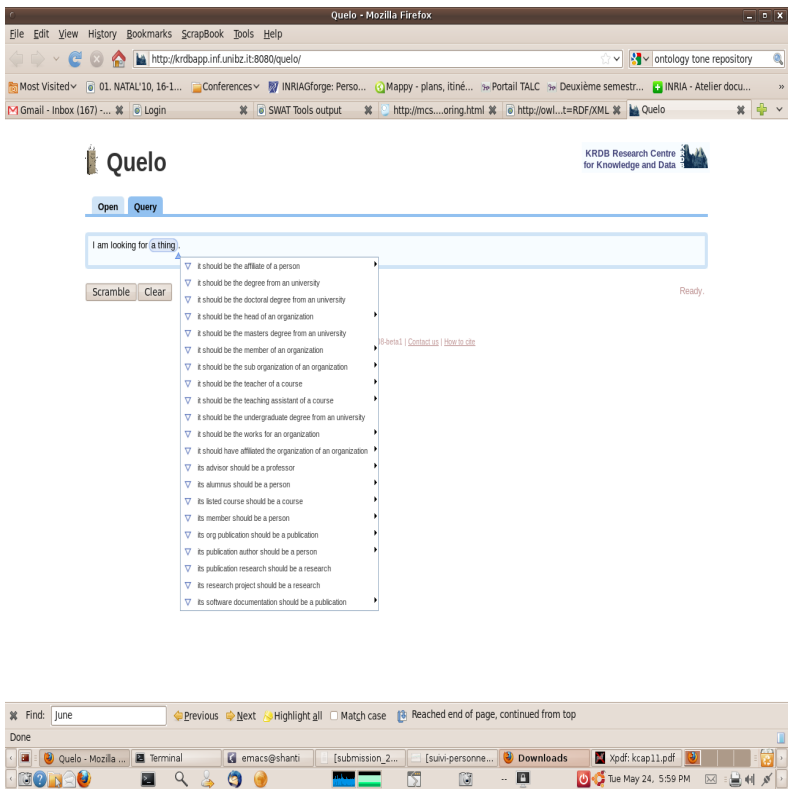


Table of Contents

Authoring Knowledge Bases

[Pow09] presents a Prolog program which allows a KB to be built up frm scratch using NL only o

Simple DL:

- One kind of statement: $C \sqsubseteq D$
- 4 Class constructors: $A, \top, \exists R.C, \{a\}$
- One property constructor: property inversion

Authoring Knowledge Bases

NL suggestions for ontology extension are generated using a very basic sentence realiser:

- Lexicon
  - Individuals realised as proper names
  - Atomic Classes realised as count nouns
  - Properties realised either by a transitive verb or a count noun
  - The name of every atomic term is identical to the lemma of the corresponding word
- Generic grammar for realising axioms and complex class descriptions

The Authoring Process

Initially, the KB contains the axiom $\top \sqsubseteq \top$.

The program generate a sentence from this axiom and provides the user with a list of editing options:

```
1: Every thing/1 is a thing/2.
t   Add a new term
a   Add a new axiom
A/C Edit class C in axiom A
A/d Delete axiom A
```

The user chooses an option and the resulting editing operation both updates the KB and triggers the generator which verbalises the update.

Example Session

```
1: Every thing/1 is a thing/2.
t   Add a new term
a   Add a new axiom
A/C Edit class C in axiom A
A/d Delete axiom A
```

The user adds for each term a triplet (word,syntactic category,logical type). E.g.,

```
t (pet,noun,class)
t (animal,noun,class)
t (own,verb,property)
```

Example Session (Ct'd)

1/1 Edit Class 1 in axiom 1

The program shows a list of possible substitutions:

```
1 Every pet
2 Every animal
3 Everything that owns one or more things
4 Everything owned by one or more things
```

User selection: 1
The KB is updated and the edited axiom is verbalised:

```
1. Every pet/1 is a thing/2
```

Example Session (Ct'd)

User selection: 1/2

System suggestions (modified to suit the current syntactic context):

```
1 a pet
2 an animal
3 owns one or more things
4 is owned by one or more things
```

User selection: 2

The KB is updated and the edited axiom is verbalised

```
1. Every pet/1 is an animal/2
```

Example Session (Ct'd)

```
1: Every pet/1 is an animal/2.
t   Add a new term
a   Add a new axiom
A/C Edit class C in axiom A
A/d Delete axiom A
```

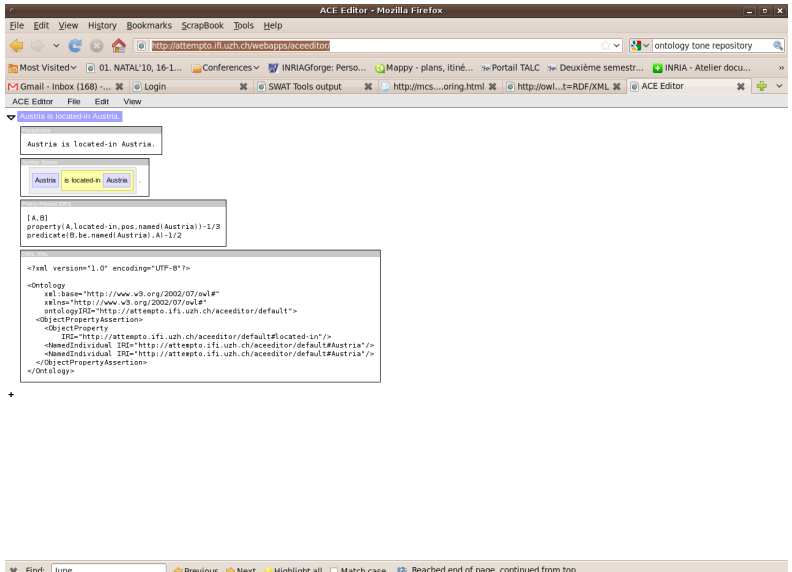
User selection: a

```
1: Every pet/1 is an animal/2.
2. Every thing/1 is a thing/2.
```

Demos 5 and 6: Authoring Ontologies

Demo 5: Power's Authoring Prolog Program. A prototype that shows how to use NLG to author an ontology

Demo 6: The ACE editor `http://attempto.ifi.uzh.ch/webapps/aceeditor/`, [Kuh10]: CNL sentences are parsed and translated into logical form (FOL, OWL, etc.). The resulting formulae are stored thus creating a knowledge base.



Acknowledgments

Thanks to Enrico Franconi (Quelo) and Richard Power (SWAT, The Authoring Tool) for providing us with demo software.

Thanks to Vinay Chaudry and to the K-CAP organisers for giving us the opportunity to give this tutorial.

Thanks!


Kalina Bontcheva.
Generating tailored textual summaries from ontologies.
In *ESWC*, pages 531–545, 2005.


P. Guagliardo E. Franconi and M. Trevisan.
An intelligent query interface based on ontology navigation.
In *Workshop on Visual Interfaces to the Social and Semantic Web*, Hong Kong, 2010.


D. Galanis and I. Androutsopoulos.
Generating multilingual descriptions from linguistically annotated owl ontologies: the natural owl system.
pages 143–146, Schloss Dagstuhl, Germany, 2007.


D. Hardcastle and R. Power.
Generating conceptually aligned text.
Technical report, The Open University, 2008.


Catalina Hallett, Richard Power, and Donia Scott.
Composing questions through conceptual authoring.
COMPUTATIONAL LINGUISTICS, 33:105–133, 2007.


Kaarel Kaljurand and Norbert Fuchs.
Verbalizing owl in attempto controlled english.
In *Proceedings of the Third International Workshop on OWL: Experiences and Directions OWLED*, 2007.


Y. Kim.
Effects of conceptual data modelling formalisms on user validation and analyst modelling of information requirements.
PhD thesis, University of Minesota, 1990.


Tobias Kuhn.
Codeco: A grammar notation for controlled natural language in predictive editors.
In Michael Rosner and Norbert E. Fuchs, editors, *Pre-Proceedings of the Second Workshop on Controlled Natural Languages (CNL 2010)*, volume 622 of *CEUR Workshop Proceedings*. CEUR-WS, 2010.


C. Mellish and X. Sun.
The semantic web as a linguistic resource: Opportunities for natural language generation.

Knowledge-Based Systems, 2005.


Marian Petre.
Why looking isn't always seeing: Readership skills and graphical programming.
Communications of the ACM, 1995.


R. Power.
Towards a generation-based semantic web authoring tool.
In *ENLG*, 2009.


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


R. Power.
Coherence relations in ontologies.
Technical report, The Open University, 2011.


Ehud Reiter and Robert Dale.
Building applied natural language generation systems.

Journal of Natural Language Engineering, 3(1):57–87, 1997.


E. Reiter and R. Dale.
Building Natural Language Generation Systems.
Cambridge University Press, Cambridge, 2000.


Alan Rector, Nick Drummond, Matthew Horridge, Jeremy Rogers, Holger Knublauch, Robert Stevens, Hai Wang, and Chris Wroe.
Owl pizzas: Practical experience of teaching owl-dl: Common errors and common patterns.
In *14th International Conference on Knowledge Engineering and Knowledge Management*, pages 63–81, 2004.


Ehud Reiter.
Has a consensus NL generation architecture appeared, and is it psychologically plausible?
In David McDonald and Marie Meteer, editors, *Proceedings of the 7th. International Workshop on Natural Language generation (INLGW '94)*, pages 163–170, Kennebunkport, Maine, 1994.


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


X. Sun and C. Mellish.
Domain independent sentence generation from rdf representations for the semantic web.
In *ECAI06 Combined Workshop on Language-Enabled Educational Technology and Development and Evaluation of Robust Spoken Dialogue Systems*, Riva del Garda, Italy, 2006.


R. Schwitter and M. Tilbrook.
Controlled natural language meets the semantic web.
In *Proceedings of the Australasian Language Technology Workshop*, pages 55–62, Macquarie University, 2004.


S. Williams and R. Power.
Grouping axioms for more coherent ontology descriptions.
In *INLG*, 2010.