Available online at www.sciencedirect.com

ScienceDirect

Artificial Intelligence 172 (2008) 1285–1315

Artificial Intelligence

www.elsevier.com/locate/artint

Natural language directed inference from ontologies

Chris Mellish [*], Jeff Z. Pan

Department of Computing Science, University of Aberdeen, Aberdeen AB24 3UE, UK

Received 17 June 2007; received in revised form 21 January 2008; accepted 25 January 2008

Available online 4 March 2008

Abstract

This paper presents an investigation into the problem of *content determination* in natural language generation (NLG), using as an example the problem of determining what to say when asked "What is an *A*?", where *A* is a concept defined in an OWL ontology. It is shown that a naive approach to this problem, which just presents a set of the stated axioms, will often inadvertantly violate maxims of cooperative conversation. What is required instead is a kind of inference that generates logical conclusions of the axioms that are suitable for natural language presentation—*natural language directed inference* (NLDI). Although NLDI, in this case a kind of non-standard inference in description logics, is hard to formalise in general, for this problem we isolate a significant subproblem—that of enumerating subsumers of *A* that are suitable for natural language presentation. For this problem, which on the face of it appears intractable, we show how factors relevant to natural language presentation enable an optimised solution that is realistic in practice.

The paper makes a contribution to the increasingly important practical problem of explaining concepts in an ontology. It also makes a first step towards the development of *domain independent* principles for content determination.
© 2008 Elsevier B.V. All rights reserved.

Keywords: Ontologies; Natural language generation; Non-standard reasoning; Content determination

1. Content determination in natural language generation

Content determination in natural language generation (NLG) is the task of determining relevant material to be included in a natural language text. Usually this is taken to involve in addition some planning of the overall structure of the text, as this will affect whether particular combinations of content will be able to be coherently realised. Because the details of content determination depend on characteristics of the application domain and in general every NLG system has a different domain or goals, there has been little success in coming up with general models of the structure of this process. In particular, reference architectures for NLG [30,39]. have relatively little to say about it.

It is useful to distinguish between two broad classes of content determination problems. "Top–down" problems have specific goals in terms, for instance, of convincing or persuading the reader about something. These have typically been addressed through the framework of planning (e.g. [34]), where content is sought to fill in requirements of planning operators. Here the requirement to build a successful argument of some kind drives the process. On the other hand, "bottom–up" problems require the production of a more general expository or descriptive text that puts together

[*] Corresponding author.
E-mail addresses: cmellish@csd.abdn.ac.uk (C. Mellish), jpan@csd.abdn.ac.uk (J.Z. Pan).

0004-3702/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.artint.2008.01.003

information to satisfy more diffuse goals. For these problems text coherence is more important than which particular arguments or points are made. For instance, the ILEX project aimed to emulate a museum curator telling a good story to link together a sequence of selected exhibits. It was argued that an opportunistic approach to content determination was needed for this sort of application [29].

In this paper, we concentrate primarily on the "bottom–up" type of content determination problem. But what makes content determination hard in either case is largely the fact that two different "worlds" are involved—the background domain and the linguistic world. Content determination is selecting material from the (not necessarily very linguistic) domain model, e.g. facts, rules and numbers, in the hope that it will permit a coherent realisation as a text. In between the domain model and the set of possible texts sits the possibly non-trivial process of *realisation*,[1] which maps portions of content to possible natural language texts. The problem is that since realisation may be complex, it will be hard to judge which content will yield the most successful text. Therefore, although content determination must be largely a problem of domain reasoning, that reasoning must somehow be influenced by the linguistic possibilities and their relative quality.

Meteer [32] pointed out that this "generation gap" in the worst case will mean that content is formulated which is not expressible in language at all. This is also related to the "problem of logical form equivalence" [42] which arises because from a domain point of view two logically equivalent formulae are interchangeable and so it is a matter of chance which of the many logically equivalent formulae is given to a realiser. An optimal approach to realisation must therefore be able to choose between realisations corresponding to all formulae logically equivalent to its input.

The problems raised by Meteer and Shieber do not, however, always arise in practice. In many applications, the possible portions of content that might be selected are restricted enough and close enough to a linguistic representation that one can be sure that there will always be at least one realisation. Also realisation doesn't have to map onto all possible texts—one can artificially limit the extent to which realisation diverges from what is suggested by the surface form of the input. All NLG systems adopt these sorts of simplications, but they mean that the realism of the application or the expressivity of the output is compromised.

In this paper we consider the problem of determining content for natural language descriptions of parts of ontologies. The deficiencies of a naive method that ignores linguistic constraints motivates the notion of *natural language directed inference* (NLDI) for content determination. However it is unclear *a priori* whether NLDI can ever be implemented in a practical way. As an example of the kind of thing NLDI might be required to do, we introduce the problem of generating subsumers of a concept that are suitable for natural language presentation. We develop a reasoning algorithm for generating such subsumers, but the combinatorics are formidable. However, constraints about natural language appropriateness can be incorporated within the algorithm to render it feasible in practice. This suggests that it may be realistic to devise reasoning mechanisms for other aspects of NLDI, and that the natural language constraints may render feasible reasoning mechanisms that would otherwise have to be abandoned.

Our results have implications for the practical application of explaining concepts in ontologies. Our formulation and implementation of NLDI can also be seen as a first step towards the development of *domain independent* principles for content determination. This is valuable because often in NLG content is either determined in advance in a way that unrealistically anticipates linguistic realisation, or is determined using non-portable, application-dependent principles.

2. Presenting ontologies in natural language

The semantic web is a vision of the web of the future consisting of resources with machine-readable semantics [40]. It envisages the use of descriptions of resources connected into "semantic networks". *Ontologies* are used to describe constraints on the meanings of terms used in these networks and to provide a basis for inter-operability.

An *ontology* is, in general, a 'representation of a shared conceptualisation' of a specific domain [18,45]. It provides a shared and common *vocabulary*, including important concepts, properties and their definitions, and *constraints*, sometimes referred to as background assumptions regarding the intended meaning of the vocabulary. Ontologies are specified in (for many humans) complex formal languages, and knowledge engineers, domain experts and also casual users need good ways to compare ontologies and understand how to use them. As the logical structure of ontologies becomes richer, so it becomes increasingly hard to devise appropriate graphical means of presentation

[1] We use the term *realisation* here in a wider sense than is commonly used in NLG, intending it to cover all aspects of linguistic processing, including those required for multiple sentences.

DL Syntax	Semantics	In $\mathcal{ALEN}$?
A	$\mathsf{A}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$	✓
$\top$	$\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$	✓
$\bot$	$\bot^{\mathcal{I}} = \emptyset$	✓
$C_1 \sqcap C_2$	$(C_1 \sqcap C_2)^{\mathcal{I}} = C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$	✓
$C_1 \sqcup C_2$	$(C_1 \sqcup C_2)^{\mathcal{I}} = C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}}$	
$\neg C$	$(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$	✓ if C atomic
$\{\mathsf{o}_1\} \sqcup \{\mathsf{o}_2\}$	$(\{\mathsf{o}_1\} \sqcup \{\mathsf{o}_2\})^{\mathcal{I}} = \{\{\mathsf{o}_1\}^{\mathcal{I}}, \{\mathsf{o}_2\}^{\mathcal{I}}\}$	
$\exists R.C$	$(\exists R.C)^{\mathcal{I}} = \{x \mid \exists y.\langle x, y\rangle \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$	✓
$\forall R.C$	$(\forall R.C)^{\mathcal{I}} = \{x \mid \forall y.\langle x, y\rangle \in R^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$	✓
$\exists R\{\mathsf{o}\}$	$(\exists R\{\mathsf{o}\})^{\mathcal{I}} = \{x \mid \langle x, \mathsf{o}^{\mathcal{I}}\rangle \in R^{\mathcal{I}}\}$	
$\geqslant mR$	$(\geqslant mR)^{\mathcal{I}} = \{x \mid \sharp\{y.\langle x, y\rangle \in R^{\mathcal{I}}\} \geqslant m\}$	✓
$\leqslant mR$	$(\leqslant mR)^{\mathcal{I}} = \{x \mid \sharp\{y.\langle x, y\rangle \in R^{\mathcal{I}}\} \leqslant m\}$	✓
$= mR$	$(= mR)^{\mathcal{I}} = \{x \mid \sharp\{y.\langle x, y\rangle \in R^{\mathcal{I}}\} = m\}$	✓
R	$R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$	✓
R^-	$(R^-)^{\mathcal{I}} = \{\langle y, x\rangle \mid \langle x, y\rangle \in R^{\mathcal{I}}\}$	

Fig. 1. OWL concept and property descriptions ($\mathcal{ALEN}$ is the sublanguage of OWL discussed in Section 5).

that do not require special training on the part of the users. In this scenario, presentation in *natural language* is becoming increasingly attractive. Natural language has developed good ways of conveying complex logical structures and requires no special training.

In this paper we concentrate on ontologies written in OWL, the web ontology language [27], as this is the nearest to an existing standard for an ontology definition language. In particular, we focus on OWL DL, which is the most expressive variant of OWL with decidable reasoning properties. Fig. 1 shows the ways of constructing concepts and properties in OWL DL, using the standard description logic syntax used in this paper. A denotes an atomic (named) concept, R an atomic (named) property, o a name of an individual and C a simple or complex concept.[2] Semantics is defined in terms of interpretations $\mathcal{I}$, each with an interpretation function $.^{\mathcal{I}}$ and a set $\Delta^{\mathcal{I}}$ of individuals. The figure shows what the interpretation of each construct in a given $\mathcal{I}$ is. The main idea is that complex concepts are built from named concepts/properties using constructors such as $\sqcup$, $\exists$, $\forall$ etc. with defined semantics. Thus for instance *Person* $\sqcap$ ($\exists$*child.Male*) represents those things which are instances of the concept *Person* for whom there is a *child* to whom *Male* applies. Ontologies are defined via axioms that relate (simple or complex) concepts. These place constraints on what can be an interpretation $\mathcal{I}$, e.g.

$$C_1 \sqsubseteq C_2 \ (C_2 \text{ subsumes } C_1) \quad \text{iff} \quad C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}} \text{ in all interpretations } \mathcal{I}$$

$$C_1 \equiv C_2 \ (C_1 \text{ is equivalent to } C_2) \quad \text{iff} \quad C_1^{\mathcal{I}} = C_2^{\mathcal{I}} \text{ in all interpretations } \mathcal{I}$$

Various questions can then be answered by inference, e.g. for some axioms $\mathcal{T}$

$$\mathcal{T} \models C_1 \sqsubseteq C_2?$$

(i.e. "does $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$ in all interpretations $\mathcal{I}$ of $\mathcal{T}$"?) Answering questions of this particular kind is the task of the *subsumption* reasoning service. Other reasoning services standardly implemented include *satisfiability* checking and *classification*, which puts concept names in their proper place in a taxonomic hierarchy (according to subsumption).

A common requirement from someone using an unfamiliar ontology is to get information about a concept mentioned in the ontology. This requirement would be most naturally stated in natural language as a question "What is an A?", where A (the *target*) is an atomic (named) concept in the ontology. Indeed, in a small survey of questions posted to FAQs about terminology, we found that 73% of questions were of this simple form. Although it would be beneficial to have more information about why someone was asking such a question, nevertheless it is likely that, especially

[2] OWL DL actually places some extra restrictions on the use of the constructs shown, though these are generally of no significance here (as we are working with ontologies that are already defined). In addition, it distinguishes between *object properties* and *datatype properties*. Here we treat datatype properties as object properties, datatypes as concepts and data values as names of individuals. This means that we lose the ability to do any specific datatype-related inferences.

in situations with casual users of ontologies, user questions will often bring no more information. Therefore this is a plausible, if demanding, scenario for investigating the generation of natural language output.

In this paper, we describe aspects of an NLG system designed to construct a natural language answer to "What is an A?" with no information available apart from the ontology. In particular, the system is limited to presenting only what the ontology (directly or indirectly) implies about A (whether or not this reflects any truth in the real world). There are many linguistic issues that are raised by such an application, but in this paper we focus on content determination and how it might be supported by appropriate reasoning. We assume that the question will if necessary lead to a longer dialogue between person and machine where, for instance, subsequently the person asks "What is a B?", for some B, perhaps mentioned in the answer to the first question.

3. Naive content selection from ontologies

A first attempt at the task of answering "What is an A?" might somehow render in natural language the set of ontology axioms that mention A. The designer of an ontology has chosen one of many possible logically equivalent ways to axiomatise their domain, and this is important information. Therefore it is plausible to take this as the basis for selecting content. One approach to selecting axioms might therefore be based on the content selection method used in the ILEX system [35]. The axioms can be seen as forming a graph, where each axiom is connected to the concepts it mentions (and where there may also be other links for relations between axioms). In this graph, routes between axioms correspond to different possible transitions in a coherent text—a text proceeds from one sentence to another by exploiting shared entities or by virtue of a specified relation between the sentences. Selecting the axioms to express in the answer involves a best-first search for axioms, starting at the entity A. Each axiom is evaluated according to a score taking into account how close it is to the concept A, how intrinsically interesting, important and understandable it is, and how few times it has already been presented.

As an example, consider a real fuel cell ontology whose axioms include the following as the only ones that mention the concept *MEA*:[3]

$$FuelCell \sqsubseteq (\forall contains.MEA)$$

$$MEA \sqsubseteq Actuality$$

$$MEA \sqsubseteq (\exists contains.Electrolyte)$$

$$MEA \sqsubseteq (\exists contains.Anode)$$

$$MEA \sqsubseteq (\exists contains.Cathode)$$

$$MEA \sqsubseteq (\leqslant 1 contains)$$

$$domain(contains, (FuelCell \sqcup MEA \sqcup Electrode \sqcup Catalyst))$$

In response to the question "What is a MEA?", the content determination approach just described, although it would be able to include information about other related concepts, would retrieve these axioms as the only things that could be said directly about a *MEA*. These might be realised as something like the following in natural language:

> "A FuelCell can only contain a MEA. A MEA is a kind of Actuality. It contains an Electrolyte. It contains an Anode. It contains a Cathode. It contains at most 1 thing. Only a MEA, an Electrode, a Catalyst or a FuelCell can contain something."

This and other examples demonstrate a number of deficiencies from which the naive content determination approach suffers. Many of these are associated with violations of the conversational maxims formulated by Grice [17]. According to Grice, normal cooperative conversation is assumed to be subject to a set of maxims for the speaker:

Quality: Do not say what you believe to be false, and do not say that for which you lack adequate evidence.
Quantity: Make your contribution as informative as is required, and no more informative than this.

[3] More details of this ontology are given in Appendix A.

Relevance: Make your contributions relevant.
Manner: Avoid obscurity and ambiguity; be brief and orderly.

These maxims represent desirable cooperative behaviour. In general, hearers will do their best to interpret things in such a way as is consistent with the maxims being followed, drawing *conversational implicatures* as necessary, in order to support this. Although in many ways Grice's maxims are simply "common sense", it is a kind of "common sense" that is distinctive to (natural language) communication with humans, rather than applying to logical reasoning in general. They also give us a way of talking about a number of problems with the above example:

1. *Problems with Quantity*: Often it may be better to present one or more consequences of an axiom (given the theory) together with it or instead of it. For instance, in the above, instead of presenting

   $$MEA \sqsubseteq (\leqslant 1 contains)$$

   (which the ontology writer put down), it would be better to present

   $$MEA \sqsubseteq (= 1 contains)$$

   which follows from this together with $MEA \sqsubseteq (\exists contains.Anode)$. In this example, most hearers, on being presented with "a MEA contains at most one thing", will naturally assume that a MEA could contain nothing, for otherwise the maxim of quantity is being violated. This is an instance of a *scalar implicature* [16,20].
   In the example, "It contains an Electrolyte. It contains an Anode. It contains a Cathode" is also misleading (suggesting that it might contain three things), because a MEA must contain a single thing which is all of these:[4]

   $$MEA \sqsubseteq (= 1 contains) \sqcap (\exists contains.Electrolyte \sqcap Anode \sqcap Cathode)$$

   Again the problem is that conveying just part of the information can be misleading. Some of the principles at work here, in deciding what can be presented without leading to misunderstanding, may be similar to those encountered in cooperative question answering [13].
2. *Problems with Relevance*: An axiom may be expressed in a way that, when realised, places inappropriate emphasis on entities. For instance, an axiom $X \sqsubseteq Y$ could be realised by "An X is a kind of Y", whereas the equivalent $Y \sqsupseteq X$ could be realised by "Y's include X's". The latter would be much better than the former at a point in a text that is discussing the properties of Y. The first and last axioms in the above example are not appropriate in a part of the text that is focussing on *MEA*, though possibly some of their logical consequences might be. As another example, instead of expressing $X \equiv Y \sqcup Z \sqcup \ldots$ one might choose to express the weaker axiom $Y \sqsubseteq X$ in a text about Y.
3. *Problems with Manner*: The axioms may not package the available information appropriately. From a logical point of view there is no natural limit to the complexity of an axiom. An axiom may be too complex to be realised in a single sentence and may not be easily decomposable into sentence-sized chunks. In this case, it might be appropriate to present a "weaker" statement.
   On the other hand, the axioms may give rise to sentences that are short and repetitive. In the example, we used three sentences to realise the axioms:

   $$MEA \sqsubseteq (\exists contains.Electrolyte)$$
   $$MEA \sqsubseteq (\exists contains.Anode)$$
   $$MEA \sqsubseteq (\exists contains.Cathode)$$

   A structurally-driven "aggregation" process taking place in realisation [41] could in fact realise this combination as "A MEA contains an Electrolyte, an Anode and a Cathode", but in this case it would be preferable to present the consequence:

   $$MEA \sqsubseteq (\exists contains.Electrolyte \sqcap Anode \sqcap Cathode)$$

   realised as "A MEA contains something that is an Electrolyte, an Anode and a Cathode", which can only be obtained by deeper domain reasoning.

[4] The same problem exists if we realise these three axioms as a single sentence: *It contains an Electrolyte, an Anode and a Cathode*.

4. Natural language directed inference

The only way to overcome these sorts of limitations is to enable content determination to select material in more ways than just choosing an axiom. It must always choose to express something that is true, given the logical theory (maxim of Quality), and content determination will therefore be a form of *inference*. In general, we could consider using any logical consequence of the axioms. However, not all logical consequences are equally good. In previous work [31], we argued that the propositions that are presented should, as well as being sound, contribute information relevant to the goal of the text, be not very different from the original axioms (and so capture some of the intent behind those axioms), have appropriate linguistic complexity when realised, satisfy linguistic coherence constraints, not have already been expressed (and not be tautologies), be complete, to the extent that they don't support false implicatures and be in accord with user model preferences. We called the kind of inference required to find such logical consequences *natural language directed inference* (NLDI). It is a kind of forwards inference with goals specific to its use for natural language generation.

In the above work, we implemented a version of NLDI from ontologies as an *ad hoc* and incomplete mechanism for generating multiple logical consequences, whose realisations were then evaluated according to linguistic criteria. It would, however, be more efficient to have linguistic principles to some extent *guiding* the reasoning process, rather than just choosing from its results. It would also make sense to exploit existing mainstream work on automated reasoning for ontologies. Standard reasoning services for DLs [21], however, require detailed specification of the reasoning goals (e.g. the subsumption service needs to be told exactly which two concepts are to be tested). Since NLDI is a kind of data-driven reasoning with goals that cannot be stated precisely in logical terms ("give me logical consequences of the axioms which are worth presenting"), standard DL reasoning services cannot be used immediately to implement NLDI. The challenge is then to exploit these efficiently implemented services in more complex ways to make NLDI possible. But *a priori* it is very unclear whether a practical version of NLDI can possibly be implemented.

One standard way of organising bottom–up content determination in NLG is to make use of *schemas* [28], patterns of content organisation observed in human-written texts. A schema can be thought of as a non-deterministic recursive transition network, where each atomic transition involves including (non-deterministically from the options) a piece of content with a specific "rhetorical predicate". Each possible traversal of the network then corresponds to a particular sequence of rhetorical predicates (with chosen instances), to be realised in the order selected. Although it may not be easy to find a schema for every situation, schemas can be seen as a good way of implementing NLDI, by reducing the goal of generating a whole text to goals involving the retrieval of instances of rhetorical predicates. McKeown addressed the description of database structure, an application not unlike ours, and so one might expect similar rhetorical predicates to be useful for explaining ontologies. Her rhetorical predicates included the following:

identification: An aircraft carrier is a surface ship with a displacement between 78,000 and 80,800 and a length between 1039 and 1063.
attributive: A torpedo has an underwater target location.
equivalent: Wines described as 'great' are fine wines from an especially good village.
constituency: Modern torpedoes are of 2 general types.
comparison: The electric powered models are similar.

In DL terms, the first three correspond to concept subsumptions $A \sqsubseteq \alpha$ wrt axioms $\mathcal{T}$, with $A = \mathit{AircraftCarrier}$, *Torpedo* and *GreatWine*.

For McKeown, the instances of the rhetorical predicates were explicitly encoded in the input, but unfortunately not all (or even any) of such subsumptions need necessarily appear directly in an ontology. For instance, in the fuel cell ontology, subsumers of MEA include concepts such as:

$$(\forall \mathit{contains}.\mathit{Electrolyte})$$

$$(\exists \mathit{contains}.((\forall \mathit{contains}.\mathit{Catalyst}) \sqcap \mathit{Electrolyte}))$$

$$(\exists \mathit{contains}.((\exists \mathit{isPartOf}.\mathit{Actuality}) \sqcap \mathit{Anode}))$$

$$(\exists \mathit{contains}.(\mathit{Actuality} \sqcap \mathit{Anode} \sqcap \mathit{Cathode} \sqcap \mathit{Electrode}))$$

but none of these are stated directly in axioms.

It is possible to use the standard classification reasoning service to retrieve examples of subsumers and subsumees, and this was the basis of our previous work [36]. Unfortunately, classification is only able to retrieve concepts that have been explicitly named. There is a need for a method to retrieve general subsumers that are worth presenting in natural language. Although there are other important aspects to answering a question "What is an A?", such a method would achieve a large part of NLDI for such questions.

5. Enumerating subsumers

In this and the next sections, we present a procedure for discovering subsumers of a concept A that might be worth presenting in natural language. To make the task feasible, we introduce two simplifications. Firstly, although we allow the ontology $\mathcal{T}$ to be expressed in full OWL DL, nevertheless the subsumers are constrained to be in the more limited language $\mathcal{ALEN}$. The last column of Fig. 1 indicates the constructs that are allowed in $\mathcal{ALEN}$. Crucially, disjunctions are not allowed and negations can only apply to atomic concepts. Secondly, we restrict the concepts α that are generated to be such that $|\alpha| \leqslant lim$, where $|\ldots|$ is a measure of syntactic complexity and lim is a fixed integer. Complexity is calculated as follows. Each negation or conjunction in a concept counts 1 towards the size. Each quantifier counts $n+1$ towards the calculation, where n is the number of enclosing quantifiers. We further assume that no number restrictions can use numbers greater than some maximum π (which is 1,000,000 in our implementation).

These modifications mean that the search space of possible subsumers is now finite (assuming a finite vocabulary), though very large—even an ontology with two proper atomic concepts and two properties has nearly 2000 concepts with complexity limit at most 5, assuming that $\pi = 3$. Also with $\pi = 3$, an ontology with 150 atomic concepts and 30 properties gives rise to over 3×10^{10} $\mathcal{ALEN}$ concepts with complexity at most 4; an ontology with 200 atomic concepts and 100 properties gives rise to over 5×10^{13} concepts of complexity at most 6.

The simplifications described above are motivated and discussed further in the context of natural language direction in Section 5.6.

5.1. Belief models

Answering questions about an ontology is a form of *communication*, and formal theories of communication standardly make reference to models of belief [1]. Here we need to distinguish between (at least) two different sets of beliefs—the system's beliefs and the (system's beliefs about the) user's beliefs (as in the system's *user model*). The first of these is represented by the original ontology $\mathcal{T}$, whilst the second requires a separate *user knowledge base* $\mathcal{U}$. In general, we seek to communicate information which is entailed by $\mathcal{T}$ (i.e. it is true) but which is not entailed by $\mathcal{U}$. The user KB is likely to be different from the system KB because otherwise the user would not have sought information about the target.

Given these two models, two different notions of subsumption arise:[5]

(1) *System-subsumption*: $C_1 \sqsubseteq_{\mathcal{T}} C_2$ iff $\mathcal{T} \models C_1 \sqsubseteq C_2$
(2) *User-subsumption*: $C_1 \sqsubseteq_{\mathcal{U}} C_2$ iff $\mathcal{U} \models C_1 \sqsubseteq C_2$

System-subsumption is subsumption as viewed by the (all-knowing) system, i.e. relative to the background ontology. On the other hand, user-subsumption is subsumption in our model of the hearer's reasoning.

We make the assumption that the user model is an approximation to the system model and contains all obvious truths, in the sense that

$$\text{For all } C_1, C_2,\ \text{if } C_1 \sqsubseteq_{\mathcal{U}} C_2 \quad \text{then } C_1 \sqsubseteq_{\mathcal{T}} C_2$$

$$\text{For all } C_1, C_2,\ \text{if } C_1 \sqsubseteq C_2 \quad \text{then } C_1 \sqsubseteq_{\mathcal{U}} C_2$$

[5] In the following, we will also sometimes mention corresponding variants of other logical tests (e.g. system- vs user-*equivalence*).

Thus the reader can determine some subsumption relationships, but this is a subset of those that are determined by the superior knowledge of the system.[6] The user-known subsumptions are shared by the system, which may also be able to discern further ones as well. Our framework allows for any $\mathcal{U}$ satisfying the above constraint, so, in particular, one could choose to have $\mathcal{U} = \mathcal{T}$ (if one wished to ignore this particular distinction) or to have $\mathcal{U}$ be something that is built up over the dialogue as a result of the information that the user is told.

The user model can potentially be used more than just to filter out subsumptions that need not be told because the hearer is aware of them. It can also be used to detect which propositions are redundant because they are implied by other propositions that will be presented. For example, consider an ontology $\mathcal{T}$ which includes the following axiom:

$$(\exists hasCar.\top) \sqsubseteq (\exists driverID.\top)$$

and the choice of one of the following subsumers of a target to be presented:

(1) $A \sqsubseteq (\exists friend.(\exists hasCar.\top))$
(2) $A \sqsubseteq (\exists friend.(\exists driverID.\top))$

From the system's point of view, it suffices to present the first of these, as it entails the second. If the user is in possession of the background domain knowledge, then the same applies. But if the user is unaware of the background knowledge, each of these subsumptions provides distinct information, and so it is worth considering presenting *both* of them (possibly with a connective such as *hence* to indicate the relationship).

The reasoning of the last paragraph suggests that only concepts that are *most specific* in terms of the user model should be presented. But there is some question about the attitude that the system should take to the difference between the belief models. In the above example, if both subsumers were presented the user might assume that owning cars and having driver IDs are not related, since otherwise the system would have been most brief and informative by presenting just the first subsumer. This is a case of the user making an implicature based on what otherwise would be a violation of Grice's maxim of quantity. If the user can be expected to make such implicatures then it would be better to be less "considerate" and simply judge redundancy on the basis of the system's beliefs. The decision of how to judge redundancy unfortunately depends on aspects of the context that we are not modelling here. Therefore, for generality, we will phrase the judgement of redundancy in terms of a third user model, the *viewpoint* model $\mathcal{V}$, with its own version of subsumption:

- *Viewpoint-subsumption*: $C_1 \sqsubseteq_{\mathcal{V}} C_2$ iff $\mathcal{V} \models C_1 \sqsubseteq C_2$

Although in principle we will allow $\mathcal{V}$ to be any knowledge base that supports a superset of the subsumptions known by $\mathcal{U}$ and a subset of those known by $\mathcal{T}$, in practice we distinguish two main ways in which a system might behave:

Selfish Mode. Here $\mathcal{V} = \mathcal{T}$ and the system judges redundancy in terms of its own knowledge
Selfless Mode. Here $\mathcal{V} = \mathcal{U}$ and the system is "considerate", judging redundancy in terms of its model of the user's knowledge

Of these, selfish mode will have the advantage of producing fewer answers to present, whereas selfless mode will avoid problems of being "too clever" and hence missing out useful information.

To summarise, we see a role for potentially three different belief models: the system model (which determines what is true), the user model (which determines what the user knows) and the viewpoint model (which determines when there is redundancy between propositions).

5.2. Which set of subsumers?

We seek to find concepts in $\mathcal{ALEN}$ which convey true information about the target A, i.e. concepts α such that $A \sqsubseteq_{\mathcal{T}} \alpha$. We restrict ourselves to α that are smaller than the size limit and which do not user-subsume the target.

[6] This means that, in terms of logical consequences (but not necessarily axioms), we are assuming that the user model is akin to an overlay model over the system knowledge base.

We also are not interested in the concept A itself. Assuming that we can leave it up to realisation to decide when to combine $A \sqsubseteq \alpha_1$ and $A \sqsubseteq \alpha_2$ into a single sentence for $A \sqsubseteq (\alpha_1 \sqcap \alpha_2)$, without loss of generality we can assume that the α are not conjunctions or trivially equivalent to conjunctions (we will say such α are not "conjunctive"[7]). Because of the size limit, there is a finite set I_A of such concepts, the "information about A".[8]

$$I_A = \{Y \mid A \sqsubseteq_{\mathcal{T}} Y \wedge |Y| \leqslant lim \wedge Y \text{ is not conjunctive or } A\}$$

I_A is the set of (small) true things that can be said about A which do not include A itself. The conjunction of the concepts in I_A, $\sqcap I_A$[9], is a kind of least common subsumer of $\{A\}$ in $\mathcal{ALEN}$. In general, $A \sqsubseteq_{\mathcal{T}} \sqcap I_A$ and they may or may not be $\mathcal{T}$-equivalent. No non-conjunctive $\mathcal{ALEN}$ concept within the size limits can be strictly $\mathcal{T}$-between A and $\sqcap I_A$ (because it would then be in I_A).

We could express the whole of I_A in natural language, one concept per sentence (or more than one per sentence, if realisation so chooses). However there is a lot of redundancy in I_A. In particular, we wish to avoid redundancy in terms of the viewpoint beliefs. We retain the idea of presenting each concept as a separate sentence, but first of all reduce I_A to a "simpler" set I'_A, for example by eliminating elements from I_A which viewpoint-subsume other elements. We therefore finally define our problem as follows:

The Subsumer Enumeration Problem. Given an atomic, system-satisfiable, target concept A in OWL-DL, find a set I'_A of $\mathcal{ALEN}$ concepts where:

- $\sqcap I'_A \equiv_{\mathcal{V}} \sqcap I_A$
- I'_A is as "simple" as possible according to $\mathcal{V}$

and return $\{\alpha \in I'_A | A \not\sqsubseteq_{\mathcal{U}} \alpha\}$.

We will discuss extra criteria for natural language presentability in Section 5.6, and criteria for "simplicity" of the final set in Section 5.7.

In practice, it would be foolish to enumerate all the elements of I_A and then reduce the set to I'_A, since I_A could be large. We therefore have a way of enumerating elements of a set whose conjunction is equivalent to $\sqcap I_A$ in order of $\mathcal{V}$-specificity. This means that certain redundant subsumers are not ever included in the set. However, we cannot cheaply catch all instances of redundancy during the enumeration, and so the resulting set still has to be checked for possible reductions. Therefore our overall approach has two stages:

(1) *Candidate Generation*. Computing a set of possible subsumers via a focussed search through all possible concepts in $\mathcal{ALEN}$. At each stage we can test whether an enumerated concept Y system subsumes A using a subsumption reasoning service. We call a concept enumerated by this search a *candidate*.
(2) *Filtering*. Further simplifying this set, for instance to ensure that no element is viewpoint-subsumed by the conjunction of the rest.

Candidate generation is described in Sections 5.3 to 5.6; the filtering of the set of candidates is described in Section 5.7.

5.3. The refinement relation

Our method to enumerate candidate subsumers is inspired by work in Inductive Logic Programming to generate program clauses by a general-to-specific search through the space of all possible clauses [12]. In this work, (downward) *refinement operators* are used to generate specialisations of clauses, in such a way that every possible clause can be reached from the most general clause by a sequence of applications of refinement operators.

[7] "Conjunctive" concepts include those with conjunctions embedded only within universal quantifiers, e.g. $(\forall R.A \sqcap B)$.

[8] If there was no size limit, then I_A could be infinite, e.g. for the case where $\mathcal{T}$ contains the axiom $A \equiv \exists R.A$.

[9] We will sometimes use $\sqcap$ as a unary operator, with the intended meaning that $\sqcap\{X_1, \ldots, X_n\} = X_1 \sqcap X_2 \sqcap \ldots X_n$.

For simplicity, we describe our refinement operator for $\mathcal{ALEN}$ concepts in terms of a refinement relation $\searrow$, where $C_1 \searrow C_2$ indicates that C_2 results from a minimal change to C_1 that makes it more syntactically complex. Because $\mathcal{ALEN}$ has no disjunction and limited negation, $\searrow$ is also a way of computing more specific concepts from concepts. The search for candidates starts from the most general concept $\top$, working to concepts C_1 such that $\top \searrow C_1$, then to concepts C_2 such that $C_1 \searrow C_2$, and so on. Thus we are exploring the set of concepts α such that $\top \searrow^* \alpha$, where $\searrow^*$ is the transitive closure of $\searrow$.

To limit the amount of redundancy in the search space, non-conjunctive concepts are assumed to be in a normal form. Thus:

- Conjunctions are only allowed (at any level) inside $\exists$ constructs. Conjunctive information at the top level or just inside $\forall$ constructs must be expanded out to yield multiple candidates.
- Within conjunctions, conjuncts (if present) occur in the following order: negations (in lexicographic order) before property restrictions (with properties in lexicographic order) before atomic concepts (in lexicographic order). Within the property restrictions, all the restrictions for a given property occur together, in the order: number restrictions before $\forall$ before $\exists$. For any property P, there are at most two number restrictions: either a single $=$ restriction or at most one of each of $\leqslant$ and $\geqslant$, in this order.
- For any property P, at most one $\forall P$ restriction can occur in any conjunction.
- Nested conjunctions are flattened and $\bot$ does not appear anywhere.[10]

$X \searrow Y$ if and only if it fits one of the following cases. This definition can also be read as the basis of an algorithm for enumerating, for a given concept α, the concepts β such that $\alpha \searrow \beta$.[11] In the actual implementation, we make a number of optimisations compared to using the basic refinement relation, as detailed below. Note that if X is in normal form and $X \searrow Y$ then Y is also in normal form.

$\top \searrow A_i$ if A_i is a named concept other than $\bot$ or $\top$
$\top \searrow \neg A_i$ if A_i is a named concept other than $\bot$ or $\top$
$\top \searrow (\exists P.\top)$ if P is a property name
$\top \searrow (\forall P.\top)$ if P is a property name
$\top \searrow (\geqslant nP)$ if P is a simple property and $0 \leqslant n \leqslant \pi$
$\top \searrow (\leqslant nP)$ if $(0 \leqslant n \leqslant \pi)$, and P is a simple property
$\top \searrow (=nP)$ if $(0 \leqslant n \leqslant \pi)$, and P is a simple property
$(\exists P.\alpha) \searrow (\exists P.\beta)$ if $\alpha \searrow \beta$
$(\forall P.\alpha) \searrow (\forall P.\beta)$ if $\alpha \searrow \beta$
$\alpha \searrow (\top \sqcap \alpha)$ where this is within the scope of an $\exists$
$\alpha_1 \sqcap \alpha_2 \sqcap \ldots \alpha_n \searrow \beta \sqcap \alpha_2 \sqcap \ldots \alpha_n$ if $\alpha_1 \searrow \beta$ and β is of a type allowed before the α_i by the conjunction ordering rules.

Refinement of a conjunction either adds a new conjunct at the front or refines the first existing conjunct. It is not possible to refine other elements of a conjunction. This is another means to reduce redundancy in the search space.

The following are two examples of paths through the refinement search space, assuming a vocabulary with just two properties (p_1, p_2) and two proper atomic concepts (c_1 and c_2):

$$\top \searrow (\exists p_1.\top) \searrow (\exists p_1.c_1) \searrow (\exists p_1.\top \sqcap c_1) \searrow \big(\exists p_1.(\leqslant 27 p_1) \sqcap c_1\big) \searrow \cdots$$
$$\top \searrow (\forall p_2.\top) \searrow \big(\forall p_2.(\exists p_1.\top)\big) \searrow \big(\forall p_2.\top \sqcap (\exists p_1.\top)\big) \searrow \cdots$$

The following properties of $\searrow$ can be shown by induction on the number of symbols (other than $\top$) occurring in C.

Lemma 1. *If C is a satisfiable non-conjunctive concept in normal form in $\mathcal{ALEN}$, expressed in terms of the vocabulary of the ontology, then $\top \searrow^* C$.*

[10] Note that we do not need to allow $\bot$ in concepts, because of equivalences such as $(\forall P.\bot) \equiv (=0P)$.

[11] The notion of "simple property" in the definition excludes transitive properties, their inverses or their superproperties. OWL DL does not permit number restrictions to be stated for such properties.

```
on-path($x$):    return ($|x| \leqslant lim$) $\wedge$ ($A \sqsubseteq_{\mathcal{T}} x$) $\wedge$ $x$ is not $A$

candidates($x$):
    $s = \{y | \exists x'(x \searrow\searrow x') \wedge \text{on-path}(x') \wedge y \in \text{candidates}(x')\}$
    if $s = \emptyset$ then return $\{x\}$
    else return $s$
    endif

return candidates($\top$)
```

Fig. 2. Search algorithm for target A.

Lemma 2. *If $C \searrow C'$ then C' is no less syntactically complex then C (for a range of possible complexity metrics, including $|\ldots|$.)*

Lemma 3. *If $C \searrow C'$ then C subsumes C' (hence C system- and user-subsumes C').*

Lemma 4. *If $\top \searrow^* C$ then C is not conjunctive.*

The first of these means that we can reach all possible concepts through $\searrow$ Because of Lemmas 2 and 3, a search following the transitive closure of $\searrow$ is both a search in terms of increasing syntactic complexity and also a (perhaps rather slow) search in terms of increasing logical specificity.

5.4. Candidate generation

The basic search strategy for candidate generation is defined in terms of a further relation $\searrow\searrow$ derived from $\searrow$, which is guaranteed to produce results strictly viewpoint-subsumed by the original concept. Again, this definition can be thought of as an algorithm to enumerate the relevant refinements:

$$X \searrow\searrow Y \quad \text{if } X \searrow Y \text{ and } X \not\sqsubseteq_{\mathcal{V}} Y$$
$$X \searrow\searrow Y \quad \text{if } X \searrow Z,\ X \sqsubseteq_{\mathcal{V}} Z \text{ and } Z \searrow\searrow Y$$

The search algorithm is shown in Fig. 2. The search through $\searrow\searrow^*$ is organised in a depth-first manner.

Theorem 1 *(Termination). candidates($\top$) always terminates.*

Proof. From the definition, it can be seen that $\searrow$ always either introduces an additional connective or replaces $\top$ by some non-$\top$ concept. It follows that there can be no loops in $\searrow^*$. The same extends to $\searrow\searrow^*$. There are only finitely many concepts x such that $|x| \leqslant lim$ and, by Lemma 2, in $\searrow\searrow^*$ such concepts can only be reached by going via other such concepts. Therefore, from any concept x for which on-path(x) holds, there are only finitely many $\searrow\searrow^*$ paths going to concepts for which no $\searrow\searrow$ value is on-path. Define the *height* of x, $h(x)$, to be the length of the longest such path. It can be shown by induction on $h(x)$ that for all concepts x such that on-path(x), *candidates*(x) terminates. In particular, *candidates*($\top$) terminates. □

Theorem 2 *(Correctness). For every x in candidates($\top$), $|x| \leqslant lim$, $A \sqsubseteq_{\mathcal{T}} x$ and x is not conjunctive or A. Thus $\sqcap\{Y | A \sqsubseteq_{\mathcal{T}} Y \wedge |Y| \leqslant lim \wedge Y$ is not conjunctive or $A\} \sqsubseteq_{\mathcal{V}} \sqcap candidates(\top)$.*

Proof. It can be seen from the definition that *candidates*(x) is only invoked for x such that on-path(x). By induction on $h(x)$ it can be shown that if on-path(x) then every element of *candidates*(x) also satisfies on-path. Since on-path($\top$) is true, all the elements of *candidates*($\top$) are on-path too. By Lemma 4, they are also not conjunctive. □

Theorem 3 *(Completeness). For any concept C in $\mathcal{ALEN}$, if $|C| \leqslant lim$ and $A \sqsubseteq_{\mathcal{T}} C$ and C is not conjunctive or A then there exists C' in candidates($\top$) such that $C' \sqsubseteq_{\mathcal{V}} C$. Thus $\sqcap candidates(\top) \sqsubseteq_{\mathcal{V}} \sqcap\{Y \mid A \sqsubseteq_{\mathcal{T}} Y \wedge |Y| \leqslant lim \wedge Y$ is not conjunctive or $A\}$.*

Proof. Without loss of generality, we can assume that C is in normal form. Then by Lemma 1, there is a refinement sequence $C_0 = \top \searrow C_1 \searrow \cdots \searrow C_k = C$. By Lemma 3, $C \sqsubseteq_{\mathcal{T}} C_i$ for all i, whence $A \sqsubseteq_{\mathcal{T}} C_i$. Also by Lemma 2, $|C_i| \leqslant |C|$ and so on-path(C_i) is always true. Within the sequence $< C_i >$ we now construct a subsequence of concepts $< C_{d(i)} >$ related by $\searrow\searrow$ by choosing their indices as follows:

$$d(0) = 0$$
$$d(i) = \text{the smallest } j > d(i-1) \text{ such that } C_{d(j)} \not\equiv_{\mathcal{V}} C_{d(j)-1}$$

Then $C_{d(0)} = \top$ and $C_{d(i)} \searrow\searrow C_{d(i+1)}$. By induction on i it can be shown that $\mathit{candidates}(\top) \supseteq \mathit{candidates}(C_{d(i)})$. Also, by induction on $h(C_{d(i)})$ it can be shown that for every i there exists at least one $x_i \sqsubseteq_{\mathcal{V}} C_{d(i)}$ such that $x_i \in \mathit{candidates}(C_{d(i)})$.

Now consider $P = C_{d(i)}$ for the largest i such that $d(i) \leqslant k$. Clearly $P \equiv_{\mathcal{V}} C$. Let C' be some x_i for this value of i. Then $C' \sqsubseteq_{\mathcal{V}} C_{d(i)} \equiv_{\mathcal{V}} C$. Since $\mathit{candidates}(\top) \supseteq \mathit{candidates}(P)$, C' is in $\mathit{candidates}(\top)$. $\square$

Theorems 2 and 3 guarantee that the above search is complete and correct (it produces a set whose conjunction is viewpoint-equivalent to I_A):

$$\sqcap \mathit{candidates}(\top) \equiv_{\mathcal{V}} \sqcap \{ Y \mid A \sqsubseteq_{\mathcal{T}} Y \wedge |Y| \leqslant \mathit{lim} \wedge Y \text{ is not conjunctive or } A \}$$

Theorem 1 guarantees termination, but does not exclude the possibility that the same solution (or logically equivalent solutions) can be generated (finitely) many times. Ensuring minimality of the set is the task of the second filtering stage of the system.

Unfortunately, these results do not mean that the search is practical. Section 5 discussed the total number of possible $\mathcal{ALEN}$ concepts with complexity below different limits. It is clear that an exhaustive search through all possible concepts cannot work. The directed search strategy just described is an improvement, given that it can exclude whole areas of the search space that do not subsume the target. However, given a target A that occurs near $\bot$ in the ontology, almost all of the possible concepts may need to be considered as possibly subsuming it. Thus it is unclear *a priori* that this procedure can be made feasible.

5.5. Optimisations

The following optimisations have no effect on the correctness or completeness of the algorithm, but merely affect efficiency.

5.5.1. Vocabulary

The size of the search space depends on the size of the vocabulary used in the ontology (especially the number of properties). In fact, some of this vocabulary could not possibly appear[12] in concepts subsuming the target concept. The vocabulary of concepts and properties used is limited to those which appear in axioms relevant to a decision on $A \sqsubseteq_{\mathcal{T}} ?\alpha$, using the relevance filter of [44]. This optimisation is harmless, given the properties of that procedure.

5.5.2. Conjunctions

In general, when the first element of a conjunction is refined, the result can be rejected if it is viewpoint-subsumed by an existing element of the conjunction. In this situation, any refinement of the new concept could have been achieved by refining the conjunction without that existing conjunct. This involves an extra subsumption test but can allow a considerable pruning of the search space.

Secondly, when a new element is added to a conjunction, this means that the previous first element will not be refined further. The new addition can be rejected if the previous first element is trivial, i.e. $\top$ or $(\forall P.\top)$. Similarly if the previous first element viewpoint-subsumes a later element or the previous first element is $(\exists P.\alpha)$, a later element is $(\forall P.\beta)$ and β does not viewpoint-subsume α. In all of these cases, the previous first element needs to be further refined in order that the overall candidate is not redundant.

[12] Except inside tautologies.

Finally, when a concept containing a conjunction is about to be enumerated as a candidate, it only needs to be considered unrefined if that conjunction would have passed the above test for adding a new element. This can be generalised into a test that checks the first conjunct recursively, inside any quantifiers, whenever a candidate is about to be proposed.

All of these optimisations simply avoid the generation of candidates viewpoint-equivalent to others that will be generated anyway. Hence they result in a $\sqcap$*candidates*$(\top)$ viewpoint-equivalent to what was computed before.

5.5.3. Refinement of number restrictions

The refinement of number restrictions can be optimised. Whenever a number restriction for a property P is to be added at some position in a candidate, the most viewpoint specific number restriction which can be inserted there and which results in the candidate system-subsuming the target can be chosen. This can be found by a directed search as follows. First of all, number restrictions of the form $(\geqslant nP)$ are placed in the candidate, with n increasing from 1 upwards, until the candidate no longer system-subsumes the target. This determines the greatest lower bound on the cardinality. Then a test with the number restriction $(\leqslant \pi P)$ is performed, where π is the largest allowed cardinality (1 million in our implementation; possibly a tighter upper bound could be found by static analysis of the axioms of the ontology). If this candidate does not subsume the target then we assume no upper bound on the cardinality. Otherwise we need to find the smallest bound less than π. We do this by counting upwards from the greatest lower bound of the cardinality, terminating when we find the first number restriction $(\leqslant nP)$ which causes subsumption of the target. If the two bounds are the same n, the result is $(= nP)$; otherwise the result is the conjunction of number restrictions for the upper (if it exists) and lower (if non-zero) bounds. If this is in fact a conjunction and size restrictions prevent its inclusion in the candidate, the individual conjuncts are non-deterministically added.

In order to allow a conjunctive number restriction always to be able to appear as the most specific number restriction in some context (size restrictions apart), the restriction on conjunctions arising in a concept is relaxed so as to allow such a two-element conjunction always to appear.

Since $\mathcal{ALEN}$ is monotonic with respect to the substitution of concepts not occurring inside the scope of negations, choosing a most specific number restriction corresponds to finding a most specific subsumer of the target. Thus these optimisations simply avoid generating subsumers of subsumers and result in an equivalent $\sqcap$*candidates*$(\top)$.

5.5.4. Introducing named concepts

Since we are interested in finding just the most specific concepts that subsume the target, whenever $\top$ is refined to an atomic concept without loss of generality it can be refined to a most viewpoint specific atomic concept such that the whole candidate, with this concept substituted, system-subsumes the target. This has the potential to limit the possibilities significantly. Similarly, when a concept $\neg C$ is introduced, it can be done with a most viewpoint general C such that the candidate still system-subsumes the target. NB these optimisations will be strengthened in Section 5.6.4.

5.5.5. Other optimisations

A number of other minor optimisations make the implementation more efficient. These include "eager" testing of "on-path" constraints during the construction of potential concepts. Also, because of the expense of calls to the reasoner, results of reasoning requests are memoised so that repeat requests can be avoided.

5.6. Natural language direction

Natural language direction allows us to constrain the search space for subsumers in important ways. These measures do not affect the correctness of solutions returned by the algorithm, but they do affect completeness. Unfortunately, logical completeness results in many solutions that are inappropriate for natural language presentation.

In this section, Sections 5.6.1 to 5.6.3 describe the relevance of the algorithm already described to natural language issues. Sections 5.6.4 to 5.6.7 describe new optimisations which incorporate additional natural language direction into the algorithm, sometimes at the expense of logical completeness.

5.6.1. Expressiveness of concept language

Our framing of the problem is already influenced by the fact that the subsumers are to be presented in natural language. Natural language easily expresses conjunctive information (e.g. the semantic structure of a noun phrase

Concept α	Size	Informal realisation of $A \sqsubseteq \alpha$
Y	0	An A is a Y
$\exists eats.Y$	1	An A eats a Y
$\forall eats.Y$	1	An A can only eat Ys
$\exists eats.Y \sqcap Z$	2	An A eats something which is a Y and a Z
$\exists eats.(\exists eats.Y)$	3	An A eats something which eats a Y
$\forall eats.(\forall eats.Y)$	3	An A can only eat something that can only eat a Y
$\forall eats.(\exists eats.Y)$	3	An A can only eat something that eats a Y
$\exists eats.Y \sqcap Z \sqcap W$	3	An A eats something which is a Y, a Z and a W
$\forall eats.(\exists eats.Y \sqcap Z)$	4	An A can only eat something that eats something that is a Y and a Z
$\exists eats.(\exists eats.Y) \sqcap (\exists eats.Z)$	5	An A eats something which eats a Y and something which eats a Z
$\exists eats.(\exists eats.(\exists eats.Y))$	6	An A eats something which eats something which eats a Y

Fig. 3. Realisations of concepts with different complexities.

like *the red book on the right shelf* is naturally conjunctive). Although natural language can express disjunction, expressions including disjunction are often ambiguous when other operators appear. People also have well-established difficulties in absorbing disjunctive information [7]. Similarly, negation of complex propositions (where negation has scope over other logical operators) tends to result in sentences that are unnatural. $\mathcal{ALEN}$ allows no disjunctions or complex negations and thus is a natural DL to act as the target for unambiguous and comprehensible natural language presentation.

5.6.2. Concept complexity

There is a limit to the complexity of a concept that can be presented in a sentence. We therefore generate least subsumers within the class of concepts that do not exceed a complexity limit. Because of the possibly complex behaviour of realisation, logical complexity and linguistic complexity may differ, but in this case we use the simple structural calculation described at the start of Section 5. The complexity calculation for $|\ldots|$ is motivated by the fact that natural language tends to express conjunctions very straightforwardly, but that nested quantifications are likely to be problematic. A quantified formula is likely to be prototypically realised as a relative clause in natural language, and nested quantifiers will therefore give rise to embedded relative clauses, which may (depending on the details of realisation) be centre embedded. It is known that centre embedded structures of a degree more than two are extremely difficult for people to process, and indeed that any kind of recursive nesting in natural language can lead to human processing difficulties. This has led some researchers to propose that there is no support for recursion in the human sentence processor [33,37]. Our complexity calculation is designed to penalise nested quantifiers because of the danger that their realisations might not be understandable.

A complexity limit of around 4 or 5 seems roughly plausible for what can give rise to a comprehensible natural language sentence. To see this, consider the informal examples of concepts and possible realisations shown in Fig. 3. This limit is also roughly in accord with the observation of Kalyanpur et al. [24] that "If … OWL class expressions are deeply nested (depth > 3), the NL sentences become difficult to read".

5.6.3. Specificity and complexity

The framing of the problem in terms of finding most viewpoint-specific concepts is an important measure for tackling the problems with Quantity introduced in Section 3. This is because Quantity suggests that material communicated should be as informative as possible. In addition, the algorithm has good properties in terms of the tradeoff between specificity and complexity.

The $\searrow$ relation usually produces some children of a concept that are logically equivalent to it, and others that are more specific. Because we use $\searrow\!\searrow$, rather than $\searrow$ in the algorithm, refinements of a concept that are viewpoint-equivalent to it are not returned. Given that these refinements are more complex than the original, the effect is that (apart from where equivalent concepts are reached by different paths through the search space) only one of the smallest of a set of viewpoint-equivalent concepts is ever returned. For instance, Fig. 4 shows 10 hypothetical concepts in a branch of the $\searrow$ search space. Each concept is equally or more syntactically complex than the concept on its left. The square boxes show sets of concepts that are equivalent according to $\mathcal{T}$ and the ellipses show sets of concepts that are equivalent according to $\mathcal{V}$. Since $\sqsubseteq_{\mathcal{V}}$ is a weakened form of $\sqsubseteq_{\mathcal{T}}$, $\mathcal{V}$ introduces a finer set of equivalence classes than $\mathcal{T}$. The use of $\searrow\!\searrow$ means that only the leftmost concepts in each ellipse can be returned as candidates (these are

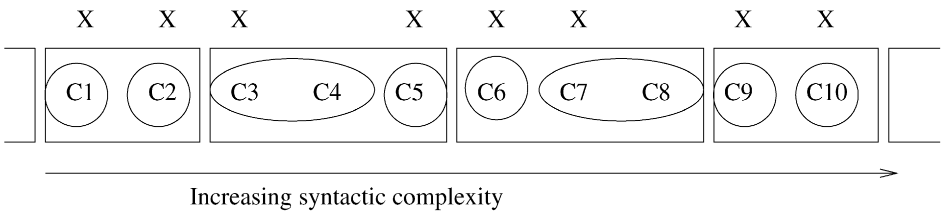

Fig. 4. Concepts in a branch of the search tree.

indicated by "X"). Thus in this example, concepts 4 and 8 cannot be returned. These are concepts that are at least as complex as the $\mathcal{V}$-equivalent concepts 3 and 7. The consequence of using $\searrow\searrow$ is that, roughly speaking, the simplest way of saying some particular content is chosen (c.f. Grice's maxims of quantity and manner).

As an example, consider the following hypothetical sequence of refinements:

(1) $\exists \mathit{father}.\mathit{Driver}$
(2) $\exists \mathit{father}.\top \sqcap \mathit{Driver}$
(3) $\searrow \exists \mathit{father}.(\exists \mathit{hasCar}.\top) \sqcap \mathit{Driver}$
(4) $\searrow \exists \mathit{father}.(\exists \mathit{hasCar}.\mathit{Vehicle}) \sqcap \mathit{Driver}$
(5) $\searrow \exists \mathit{father}.(\exists \mathit{hasCar}.\mathit{Illegal} \sqcap \mathit{Vehicle}) \sqcap \mathit{Driver}$

If the user (or the system if the system viewpoint is taken) knows that drivers have cars and that cars are vehicles, $\searrow\searrow$ will go immediately from (1) to (5). (1) to (4) are logically equivalent according to the viewpoint, and so the smallest, i.e. (1), is the only one that can be returned. In examples like this, the algorithm is tackling a limited case of the problem of logical form equivalence—choosing from among the logically equivalent possibilities using natural language criteria (simplicity).

5.6.4. *Introducing new terminology*

We now move to additional natural language motivated optimisations. Consider the situation when the two following candidates are being considered:

$$A \sqsubseteq (\exists \mathit{hasPet}.\mathit{Poodle})$$
$$A \sqsubseteq (\exists \mathit{hasPet}.\mathit{Dog})$$

where $\mathit{Poodle} \sqsubseteq_{\mathcal{T}} \mathit{Dog}$ but $\mathit{Poodle} \not\sqsubseteq_{\mathcal{V}} \mathit{Dog}$. Given that the hearer has the opportunity in the dialogue to ask followup questions about mentioned atomic concepts and can in this way obtain information in terms of more general atomic concepts, it is actually complete in a dialogue sense simply to return just the concept including *Poodle*. If the user does not understand *Poodle* then they can ask a followup question *What is a Poodle?*, which will then introduce more general concepts such as *Dog*.

Because of the special situation with named concepts that they can be asked about in followup questions, we use *system*-subsumption, instead of viewpoint-subsumption, in choosing the least atomic concepts to include. This makes a significant reduction in the search space. Since system least concepts are also viewpoint least, there is an effect on completeness but not correctness.

5.6.5. *Negations*

In natural languages, negation is used primarily to *deny* an explicitly or implicitly available proposition [46]. This means that exchanges like the following are inappropriate:

Q. What is a mammal?
*A. *It is not a kind of mushroom.*

In the context of a longer natural language discourse, it is reasonable to generate negations to contradict expectations raised earlier in the discourse. In the generation of single propositions, however, it only seems to make sense if somehow the expectation is raised within the same proposition. Our interpretation of this requirement is therefore the condition that a negation $\neg\alpha$ can only be generated if it is within a conjunction where there is another conjunct (which must be a positive atomic concept) β such that $\alpha \sqcap \beta$ is satisfiable. This can be thought of as a kind of "β but not α" negation. For instance, it is perfectly reasonable to say "... a pizza but not a vegetarian dish", because it is possible to be both a pizza and a vegetarian dish.

For this handling of negations, we use *system*, rather than viewpoint, satisfiability. This is because a sentence like *It is a kind of vertebrate and not a mushroom* seems to support the implicature that it is possible to be both a vertebrate and a mushroom. This is unacceptable unless this is actually true.

Since conjunctions are only allowed inside $\exists$ constructions, the restriction on negations would prevent them from occurring outside these contexts. Thus we relax the conjunctions stipulation and allow a conjunction of just a positive and a negative atomic concept in all places. In this situation, the positive atom acts as a kind of "guard" for the negation.

The result of this constraint is to considerably reduce the possible occurrences of negations in candidates, at the cost of logical completeness.

5.6.6. *Empty universals*

If instances of a given concept cannot possibly have values for a property P then all $\forall$ restrictions on this property are trivially true of such instances. Such restrictions are not appropriate to be expressed. Consider, for instance, the following inappropriate exchange about a pizza topping:

Q. What is a SpicyTopping?
*A. *A SpicyTopping can only have a Pizza as a topping*

In this example, the implicature that spicy toppings can have toppings is facilitated. However, pizza toppings cannot themselves have toppings. This means that, for instance, $\forall \mathit{topping}.\mathit{Pizza}$ system-subsumes *SpicyTopping*. In this situation, *any* concept of the form $(\forall P.\alpha)$ system-subsumes the target. Although logically each of these is a subsumer, in natural language terms each of them is trivial and not worth expressing (maxim of Manner). The same problem can arise with $\forall$ occurring at any nesting within a candidate.

Our solution to this problem is that if at any point we are planning to insert the concept $(\forall P.\top)$ or its refinements at some place in a candidate then first of all we look to see whether the concept with $(\forall P.\bot)$ in this position system-subsumes the target. If so, then we judge that any universal would be trivial and refrain from introducing one. Again, this results in a substantial reduction in the search space, at the cost of logical completeness.

5.6.7. *Trivial subconcepts*

Section 5.5.2 discussed avoiding leaving $\top$ or $(\forall R.\top)$ in concepts. This can be generalised to a test that avoids including any concepts system-equivalent to $\top$. Such concepts include $\forall R.C$ if C is defined as the range of R and also $(= 1R)$ if R is defined to be functional. Thus candidates like

$$A \sqsubseteq \big(\forall \mathit{father}.(\forall \mathit{child}.\mathit{Person})\big)$$

should not be returned if *Person* is inside the range of *child*. If this was expressed, for instance as *An A can only have a father whose children are people*, then a natural implicature would be that children need not be people. In general, it seems plausible that the reader will assume that no subconcept of an expressed subsumer (apart from literally $\top$ inside an $\exists$) system-subsumes $\top$. This follows from the maxims of Quantity and Manner—why should a speaker indicate $\top$ in such a laborious way?

Although the above test prevents a large number of candidates with nested $\forall$s from being returned, nevertheless all of these have to be proposed and tested in the first place. We introduce a further optimisation to prevent some of these from ever being considered. In a procedure analogous to the relevance filter (Section 5.5.1) we can process the axioms of the ontology in advance to see for which properties the axioms involving the target and related concepts say "something special" (i.e. use explicit restrictions for the properties[13]). Only for such "special properties" do we need to consider introducing top-level restrictions into candidates—for all others, the only possible true restrictions will be simply restating global information (e.g. ranges).

[13] Some care is needed here to consider also propositions inside the context of properties whose inverses might be stateable.

What is said	What might be inferred	Section
"$A \sqsubseteq \ldots X \ldots$"	$X \not\equiv_{\mathcal{T}} \top$	5.6.7
"$A \sqsubseteq \ldots B_1 \ldots$", not "$A \sqsubseteq \ldots B_2 \ldots$"	$B_2 \not\sqsubseteq_{\mathcal{T}} B_1$	5.6.4
"$A \sqsubseteq \ldots \neg B_2 \ldots$", not "$A \sqsubseteq \ldots \neg B_1 \ldots$"	$B_2 \not\sqsubseteq_{\mathcal{T}} B_1$	5.6.4
"$A \sqsubseteq \ldots B_1 \sqcap \neg B_2 \ldots$"	$(B_1 \sqcap B_2) \not\sqsubseteq_{\mathcal{T}} \bot$	5.6.5
"$A \sqsubseteq \ldots C_1 \sqcap C_2 \ldots$"	$C_1 \not\sqsubseteq_{\mathcal{V}} C_2, C_2 \not\sqsubseteq_{\mathcal{V}} C_1$	5.5.2
"$A \sqsubseteq \ldots (\forall R.C) \ldots$"	$A \not\sqsubseteq_{\mathcal{T}} \ldots (\forall R.\bot) \ldots$	5.6.6
"$A \sqsubseteq C_1$", "$A \sqsubseteq C_2$"	$C_1 \not\sqsubseteq_{\mathcal{V}} C_2, C_2 \not\sqsubseteq_{\mathcal{V}} C_1$	5.1

Fig. 5. Anticipated implicatures.

The procedure of Section 5.5.3 can be further focussed to avoid coming up with number restrictions which are simply expressing global restrictions for the properties concerned. This is done by starting the iterations from the global number restrictions for the property concerned (these can be found once and for all by invoking the procedure of Section 5.5.3 to find the tightest number restriction subsuming $\top$) and only returning results that are tighter.

None of these optimisations actually affect logical completeness in the limited sense that the $\sqcap$*candidates*$(\top)$ computed is viewpoint-equivalent to what it would have been before.

5.6.8. *Summary of implicatures*

Fig. 5 summarises the implicatures which we have assumed in this paper will be made by a reader expecting the Gricean maxims to be followed (B_1 and B_2 are atomic concepts and X is any concept other than $\top$ immediately inside an $\exists$). In each case, we state the logical content of something that could be said, the inference that the user is liable to make and the number of the section where we have discussed it.

In stating what will be inferred, we have explicitly specified the knowledge base $\mathcal{T}$ or $\mathcal{V}$. In the former case (or if $\mathcal{V} = \mathcal{T}$), we assume that the reader infers something about what is true. In the other case, we assume that the reader will infer that the system believes that this is true in the user's world model.

We have specified $\mathcal{T}$ here in places where it seems that the implicature is very strong and also perhaps unconscious. On the other hand, we have specified $\mathcal{V}$ where the user might consciously consider what might be intended by a "considerate" system and how this might not correspond directly to what is true. Our judgement here could certainly be challenged. The most contentious cases here are probably the second and third. Although we considered the use of $\mathcal{V}$ for these in Section 5.5.4, we modified this to $\mathcal{T}$ in Section 5.6.4 when we considered the dialogue context. In our evaluation of the system, we will consider the implications of this choice.

5.7. *Filtering the set of candidates*

Ideally, we wish to produce a set I'_A such that $\sqcap I'_A \equiv_{\mathcal{V}} \sqcap I_A$ and where I'_A is as "simple" as possible.[14] From candidate generation, we have the set *candidates*$(\top)$ whose conjunction is viewpoint-equivalent to $\sqcap I_A$ and which we therefore seek to transform into such an I'_A. Unfortunately, in general, conjunctions of elements in *candidates*$(\top)$ may subsume one another in unexpected and idiosyncratic ways, and finding an optimal I'_A is surely intractable. We have therefore considered three possible methods of approximating this (where α_i, α_j are elements of *candidates*$(\top)$):

(1) *Pairwise*. Elements α_i, α_j are removed in a greedy way, according to the following principles:
  - If $\alpha_i \equiv_{\mathcal{V}} \alpha_j$ then remove whichever of α_i or α_j is the more complex (or one of them randomly, if they are of equal complexity).
  - If $\alpha_j \sqsubset_{\mathcal{V}} \alpha_i$ then remove α_i

(2) *Single*. Elements α_i are removed in a greedy way, as follows:
  - If $\sqcap_{i \neq j} \alpha_j \sqsubset_{\mathcal{V}} \alpha_i$ then remove α_i

(3) *Classification*. Each α_i is declared to be equivalent to a new named concept in $\mathcal{V}$. Also $\sqcap \alpha_i$ is given a name. The immediate parents of $\sqcap \alpha_i$ are retrieved from the reasoner. From each set of $\mathcal{V}$-synonyms in this set, one element of minimal complexity is non-deterministically returned.

[14] We count the complexity of a set of concepts as the sum of the complexities of the concepts in the set.

All three of these approaches are limited in that they can only return a subset of the original set. There are, however, other ways in which the set can be simplified. For instance, for some element α_i, if there is a concept $\alpha'_i \sqsupset_{\mathcal{V}} \alpha_i$ (not in *candiates*$(\top)$) such that $(\alpha'_i \sqcap \alpha_j) \equiv_{\mathcal{V}} (\alpha_i \sqcap \alpha_j)$ (for some α_j) then α_i can be replaced by α'_i. For a few examples, we carried out an exhaustive search for all cases of this and found only instances of the following two cases, which can be detected structurally. In our implementation, these are taken account of before the set of candidates is simplified by one of the above three methods.

(1) "Reduction using local ranges". An α_j of the form $(\forall p_1.\ldots \sqcap (\forall p_2.\ldots \sqcap (\ldots(\forall p_n.C \sqcap \ldots)\ldots)))$ is expressing a constraint on the possible values of the property chain $p_1 p_2 \ldots p_n$ applied to the target. Therefore, if some other α_i has the expression $C \sqcap \ldots$ in the context of quantifiers $(\exists p_1.(\exists p_2.\ldots(\exists p_n.\ldots)))$ α_i can be replaced by a simpler concept that does not mention C in this context. As an example, for *MEA* in the fuel cell ontology, the subsumer $(\exists \mathit{contains}.(\forall \mathit{contains}.\mathit{Support}) \sqcap \mathit{Electrolyte})$ is replaced by $(\exists \mathit{contains}.(\forall \mathit{contains}.\mathit{Support}))$ because of the presence of the subsumer $(\forall \mathit{contains}.\mathit{Electrolyte})$. This in turn is replaced by $(\exists \mathit{contains}.\top)$ because of the presence of the subsumer $(\forall \mathit{contains}.(\forall \mathit{contains}.\mathit{Support}))$. In fact, finally, $(\exists \mathit{contains}.\top)$ is removed because it subsumes the existing subsumer $(\exists \mathit{contains}.(\exists \mathit{contains}.\top))$.
(2) "Redundant negation guards". This is a consequence of our strategy of requiring that negations are conjoined with corresponding positive information. Because of this, we can have two candidates $\neg A \sqcap B$ and $\neg A \sqcap C$. In this case, the latter can be replaced by simply C. The same can apply at any level of nesting. As an example, the candidate subsumers of *AmericanHot* in one version of the Manchester pizza ontology include $\neg \mathit{VegetarianPizza} \sqcap \mathit{NamedPizza}$ and $\neg \mathit{VegetarianPizza} \sqcap \mathit{SpicyPizza}$. The latter is reduced to simply *SpicyPizza* (which is in fact then eliminated, if a system viewpoint is taken, as it subsumes the existing candidate $(\exists \mathit{hasTopping}.\mathit{HotGreenPepperTopping})$).

By consideration of the individual cases, it can be seen that none of these final changes to the set of candidates can introduce a concept that violates the various natural language appropriateness principles of Section 5.6.

Once the set of candidates has been enumerated, any elements which user-subsume the target can be removed.[15]

6. Implementation and performance

6.1. Implementation and example

The system is implemented in SWI Prolog (version 5.4.7), using a version of Huang and Visser's extended DIG interface for Prolog [23] to communicate with the RacerPro reasoner (version 1.9.0) [19]. RacerPro is used to provide all reasoning services associated with the system and user (i.e. system- and user-subsumption, system children and system parents).

With a complexity limit of 4 and in selfish mode, the system generates 24 candidates for *MEA* in the fuel cell ontology with 1139 calls to RacerPro being required in the process. Using the pairwise filtering strategy, this is reduced to a set of 6 subsumers:

$$(\exists \mathit{contains}.(\exists \mathit{contains}.\top))$$

$$(\exists \mathit{contains}.(\forall \mathit{contains}.(\mathit{Catalyst} \sqcap \mathit{Support})))$$

$$(= 1 \mathit{contains})$$

$$(\forall \mathit{contains}.\mathit{Electrolyte})$$

$$(\forall \mathit{contains}.\mathit{Cathode})$$

$$(\forall \mathit{contains}.\mathit{Anode})$$

[15] There does not seem to be any obvious way to use this to guide the candidate enumeration or filtering, since if the general-to-specific search method encounters a concept that user-subsumes the target, some further refinements of it still might not do.

In the filtering process, amongst other things the subsumer *Actuality* is removed, as this strictly system-subsumes $(= 1 \mathit{contains})$. Also subsumers such as $(\exists \mathit{contains}.(\mathit{Anode} \sqcap \mathit{Cathode}))$ are filtered out using local ranges. With appropriate aggregation[16] and ordering, the 6 survivors might be realised as:

> A MEA contains exactly 1 thing. A MEA can only contain something which is an Anode, a Cathode and an Electrolyte. A MEA contains something which contains something. A MEA contains something which can only contain something which is a Support and a Catalyst.

In selfless mode, the system produces 78 candidates for the same question, which is filtered to just 18 subsumers. These include the last 5 above, plus also *Actuality* and concepts such as:

$$(\exists \mathit{contains}.(\forall \mathit{contains}.(\mathit{Actuality} \sqcap \mathit{Support})))$$
$$(\exists \mathit{contains}.((\exists \mathit{contains}.\mathit{Support}) \sqcap \mathit{Actuality}))$$
$$(\forall \mathit{contains}.(\exists \mathit{contains}.(\mathit{Actuality} \sqcap \mathit{Support})))$$
$$(\forall \mathit{contains}.(= 1 \mathit{contains}))$$
$$(\forall \mathit{contains}.(\forall \mathit{contains}.\mathit{Support}))$$

6.2. Overall feasibility

Although, using a 1.7 GHz PC, when running the system we observe calls to RacerPro occurring up to about 100 times a second (which means that RacerPro itself is working faster than this), it is clear that the use of the reasoner represents the system bottleneck, and hence we have measured performance in terms of the number of calls to RacerPro that are required. In evaluating overall feasibility, we assume simplistically that a computation involving up to about 5000 calls to RacerPro will be achievable in quasi-real time, whereas one involving over 50 000 calls will not be practical at all.

Clearly the complexity of our procedure for generating subsumers depends on the shape of the particular ontology, the number of subsumers there are and other factors that will vary widely between examples. In order to get an impression for general performance, we selected a number of existing ontologies in the following way. We started from a set of ontologies gathered at the University of Manchester for benchmarking of DL reasoners [15], and from these extracted those with “owl” as part of their file names. The result was a set of 112 ontologies, which can be classified crudely in terms of the size of their XML representation: 70 less than 50 K bytes, 16 between 50 K and 100 K bytes, 19 between 100 K and 500 K bytes, and 7 greater than 500 K bytes. As an initial test of feasibility, we looked at the 16 ontologies with size between 50 K and 100 K bytes as being hopefully representative of medium-sized ontologies. Of these, 4 were trivial for our purposes (they used no property restrictions), 2 failed to get past the SWI Prolog RDF parser without errors and 1 failed to classify in RacerPro. The 9 remaining ontologies had on average 382 ± 164[17] axioms, mentioning 162 ± 113 different named concepts and 32 ± 28 properties. For each of these ontologies, we randomly chose two concepts within the immediate parents of $\bot$. The ontologies and concepts used are listed in Appendix B. For each concept as target, we measured the performance of the candidate generation process, in terms of the number of candidates returned and the number of calls to the reasoner. For this test, the viewpoint was the system, the user model was empty (with this viewpoint, in fact the user model makes no difference to the candidate generating process) and the complexity limit was 4. The result was that on average 25 ± 42 candidates were found, using 5024 ± 4172 calls to RacerPro.

In order to get larger-sized ontologies, we then selected the 19 whose XML files were between 100 K and 500 K bytes long. Unfortunately 8 of these were trivial for our purposes (the ontologies used no property restrictions), 1 was not legal OWL DL, 1 failed to read correctly using the RDF parser and 1 produced RacerPro errors under reasoning. That left us with 8 ontologies for testing with, each containing on average 1019 ± 369 axioms, mentioning 148 ± 73

[16] Aggregating a conjunction of concepts of the form $\forall \rho.C_i$ can be done structurally without any knowledge of the background theory. As we have pointed out, because we do not enumerate concepts with top level conjunctions, this kind of aggregation must be performed subsequent to the enumeration of subsumers, as a kind of linguistic optimisation.

[17] We present statistical summaries in the form *mean* ± *standard deviation* and round all figures to the nearest whole number.

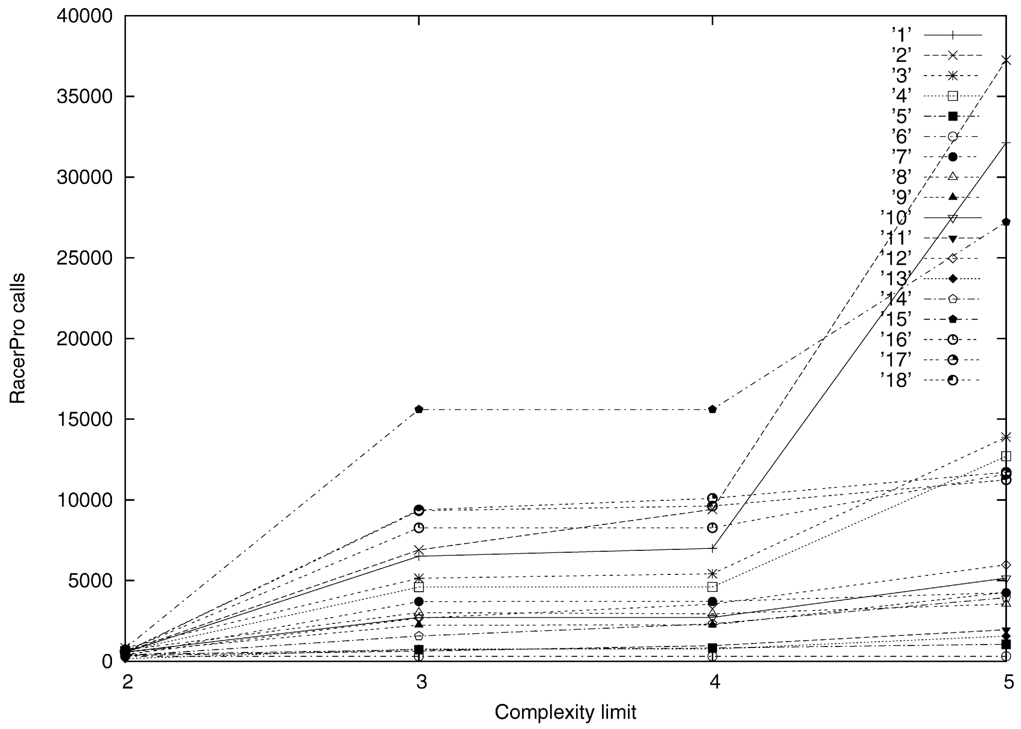

Fig. 6. RacerPro calls (system viewpoint) according to complexity limit.

different named concepts and 100 ± 92 different properties. For each of these, we chose two named concepts as before (see Appendix B for their identities). For one of the ontologies, we abandoned the search for candidates for both concepts (numbers 13 and 14) after using 50,000 calls to RacerPro. For the remaining concepts, on average 39 ± 36 candidate subsumers were found, with 6166 ± 7085 calls to RacerPro.

These results suggest that, even if we exclude ontologies which are trivial or ill-formed, in practice quasi-real time behaviour is achievable for about half of medium-sized ontologies (and presumably also for smaller ones). We found no example with a medium-sized ontology where the computation was impractical. For larger ontologies, the computation can become impractical, but for some examples the computation is quasi-realtime.

6.3. Scalability

Performance of the system clearly depends significantly on the complexity limit for concepts. Fig. 6 shows how the number of calls to RacerPro changed for each of the 18 medium-sized examples as the complexity limit changed from 2 to 5. Changing the complexity limit from 3 to 4 did not significantly affect the computation, and changing up to 5 did not affect many significantly. These facts are partly because of the characteristics of our complexity measure—concepts with complexity 3, 4 and 5 can contain two quantifiers and no more, whereas concepts with complexity 6 can contain three. We abandoned doing the same tests with complexity limit 6 because most examples were timing out after 50,000 calls to RacerPro.

Fig. 7 shows the number of candidates generated as the complexity limit went up from 2 to 5 (also on the medium-sized examples). The number of candidates has a bearing on the efficiency of the filtering stage. Generally the numbers are quite modest, and the number does not necessarily increase significantly with the complexity limit.

6.4. Value of specific optimisations

What about the individual optimisations arising from natural language direction? Of course, natural language direction has already contributed to feasibility in guiding the formulation of the problem. It soon emerged that there was little sense in testing the system without the optimisation for vacuous universal quantifiers (Section 5.6.6)—if the

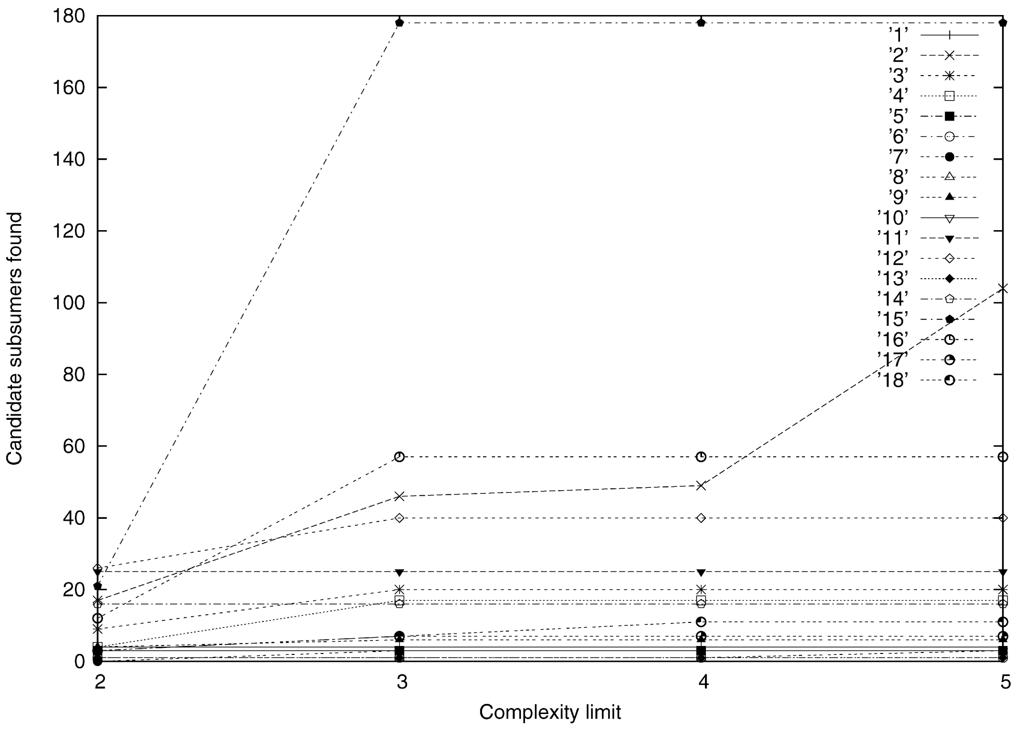

Fig. 7. Candidates generated (system viewpoint) according to complexity limit.

target is system-subsumed by ... $(\forall P.\bot)$ ... then all possible concepts that are not too complex can be substituted for $\bot$, which can result in a huge number of subsumption tests.

Several optimisations come into play in the transition from selfless to selfish mode (improved refinement of number restrictions and pruning of conjunctions). We therefore compared performance in these modes for the medium-sized examples. Fig. 8 shows the number of RacerPro calls with the viewpoint set to "system" or "user". We also compared these to running the system with "user" viewpoint and with the "new terminology" optimisation (Section 5.6.4) disabled (this is indicated by "allnames" in the figure). For this test, we imposed an artificial limit of 50,000 calls to RacerPro. Fig. 9 shows the number of candidates generated with these settings. For examples that timed out, we can give no numbers for candidates. The figures show that there is relatively little difference between the two different viewpoint settings. This is probably because many of the implicatures that we consider are computed with respect to the system knowledge base (Section 5.6.8). On the other hand, performance worsens significantly if the "new teminology" optimisation is missed out. This underlines the importance of exploiting the dialogue situation to minimise the amount of information expressed in one turn.

7. Relation to other work

7.1. Natural language processing and description logics

Although, in most NLG work, content-determination is considered to be domain-dependent and only implicitly guided by general constraints on natural language communication, there is a precursor to the idea of NLDI in the work of Hovy [22]. Hovy considers a notion of "generator-directed inference" that is able to, for instance, conclude that *Kennedy beat Carter by* 335 *votes* follows from *Carter got* 1850 *votes. Kennedy got* 2185. He points out that defining a "fixed correspondance" between provided information and grammatical rules and lexical elements results in text that is bad or boring. An act of "interpretation" is needed to produce new valid information that supports better text. In Hovy's system such inference is interleaved with realisation. Unfortunately, Hovy's approach relies on domain-specific inference rules and also does not answer the question *why* certain propositions give rise to better text

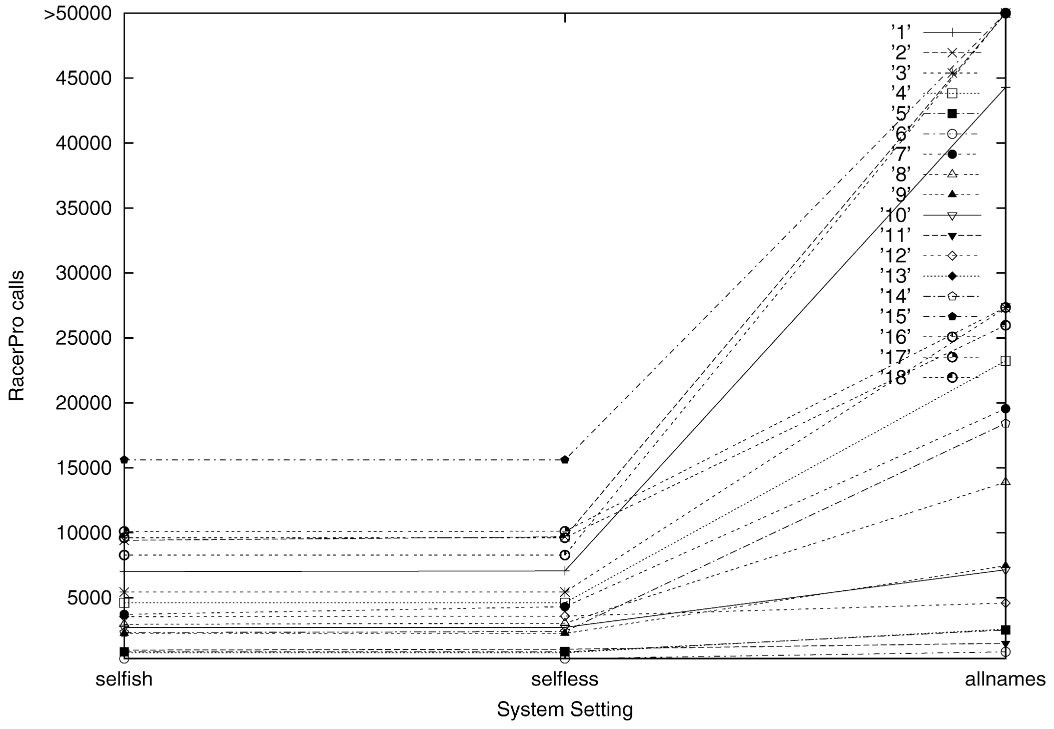

Fig. 8. RacerPro calls according to system setting (limit 4).

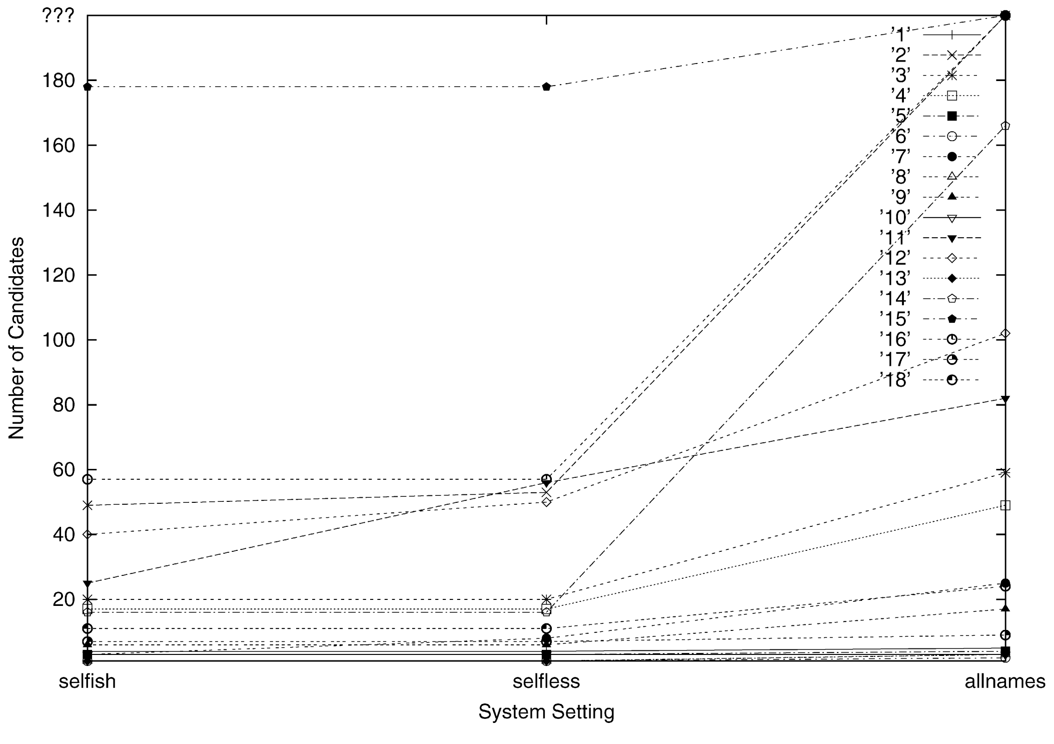

Fig. 9. Candidates generated according to system setting (limit 4).

than others. Our work can be seen as an attempt to produce some domain-independent principles for NLDI, though as Hovy's work shows this would be more powerful if domain-dependent rules were allowed as well.

As for the connection between ontologies and NLG, ontology editing tools standardly make use of natural language glosses to present logical axioms or concepts to the user [9,24,25]. By the nature of these applications, content determination and relevance is determined entirely by the user, and so there is no call for anything like natural language directed inference. Other researchers [11,47] have looked at presenting ontologies/ontology instance data in natural language, but again the NLG systems involved assume a prior stage of content determination which is not described.

There has been a considerable amount of work on the use of theorem provers of various kinds to support other natural language processing tasks (mostly aspects of natural language *understanding*), as for instance is motivated strongly by Blackburn et al. [5]. Almost all of this work assumes more expressive logical languages than OWL DL, and the highly optimised tools for DL reasoning have unfortunately remained relatively unexploited for NLP.

7.2. Related uses of the Gricean maxims

There are interesting connections between our work and Young's work on generating descriptions of plans [48]. In his case, the goal is to follow the Gricean maxims by generating the minimal description of a plan from which the user can recreate essentially the whole plan. One of his algorithms (the "brute force" algorithm) is similar to ours, in that plans are enumerated in order of specificity until an acceptable one is reached. Young's work relies on a model of the user that is specifically tied to planning. It is hard to speculate about the relative complexity of Young's plan space and the space of $\mathcal{ALEN}$ concepts with a given vocabulary, though it seems that the structure of the planning task (or maybe just our relative understanding of it) means that there is less redundancy and less need to worry about conversational implicatures than in our case.

The Gricean maxims have been suggested previously as being relevant to NLG. [43] found that, for generating natural language summaries of time series data, standard data analysis algorithms such as segmentation had to be modified and that the extra requirements that forced these modifications could be characterised partly in terms of the Gricean maxims. The Gricean ideas are also appealed to for the task of generating referring expressions, where criteria such as adequacy and minimality are often invoked [10]. However, with referring expressions the task is generally taken to be one of *reliable identification* of the referent(s). The task of ontology explanation we consider here is more that of *informative description* of the concept, which means that standard algorithms for generating referring expressions cannot be used. More relevant to our application is work on lexical choice. Reiter [38] presents a model of lexical choice which assumes a description logic-like underlying knowledge representation (this aspect is later explicitly developed by Gardent and Jacquey [14]) and then uses Gricean preference criteria for individual words (dealing with brevity and basic level status).

7.3. Non-standard inference for DLs

The idea that non-standard reasoning might be essential for explaining ontologies goes back at least to [26], where DL reasoning is recast in ways that lead to better explanations. Unfortunately we are aware of no previous work that adapts DL reasoning to support presentation of the logical *consequences* of an ontology.

The subsumer enumeration problem has relationships to a number of "non-standard" types of inference for DLs. On the one hand, the task can be regarded as a "matching" problem, "$A \sqsubseteq^? P$", where P is a concept pattern [2]. On the other hand, our task could be characterised as that of "approximation"—looking for a least subsumer (other than A itself) of A using a less expressive DL [6]. Unfortunately, existing approaches to computing approximation and matching only apply to less expressive DLs than OWL DL and assume axioms to be unfoldable (i.e. with no cyclic definitions). In this work, apart from assuming the existence of standard reasoning services, we do not assume unfoldability of the axioms.

Baader et al. [4] present a general framework for computing in a sublanguage $\mathcal{L}_1$ a least common subsumer of a set of concepts in an expressive DL $\mathcal{L}_2$. Computing $\sqcap I_A$ can be regarded as an instance of this, where the set of concepts is $\{A\}$, $\mathcal{L}_2$ is OWL-DL and $\mathcal{L}_1$ is $\mathcal{ALEN}$. Our problem is unusual in imposing a size limit on subsumers, finding the lcs of a singleton set, and excluding the use of A itself. Baader shows that a lcs may not exist if $\mathcal{T}$ has cycles, but an lcs will exist if a size limit is imposed. Baader's work unfortunately only considers $\mathcal{L}_2$ being $\mathcal{ALC}$ and $\mathcal{L}_1$ being $\mathcal{ALE}$.

There are also interesting similarities to meta-level control of inference, where factors outside of the logic (e.g. other kinds of descriptions of the shapes of logical formulae) are used to guide inference [8].

7.4. Relation to rewriting

Baader et al. [3] present the general problem of rewriting a concept C in DL $\mathcal{L}_s$ into a new concept E in DL $\mathcal{L}_d$, such that $C \rho E$, for some relation ρ, and where E is minimal with respect to an ordering $\preceq$. The rewriting can take into account a third DL $\mathcal{L}_t$.

The whole process of going from A to $\sqcap I'_A$ can be regarded as an instance of this, where $C = A$, $\mathcal{L}_s = \mathcal{L}_d = \mathcal{ALEN}$, $\mathcal{L}_t =$ OWL-DL, ρ is $\sqsubseteq_{\mathcal{T}}$ and $\preceq$ is an ordering taking into account concept size and $\sqsubseteq_{\mathcal{V}}$. However, our problem is significantly different from the *minimal rewriting problems* investigated in detail by Baader, which means that it is unlikely his algorithm could be adapted. Because ρ is $\sqsubseteq_{\mathcal{T}}$ (rather than $\equiv$), we have no useful starting concept to extend. Also Baader's algorithm can only introduce new names and remove redundant subconcepts—it cannot introduce other complex concepts.

Although our whole process results in a much more complex example of rewriting than has been considered before, just the step of reducing $\sqcap I_A$ to $\sqcap I'_A$ can also be seen as a kind of rewriting. Here $\mathcal{L}_s = \mathcal{L}_d = \mathcal{ALEN}$, $\mathcal{L}_t =$ OWL-DL, the axioms used are $\mathcal{V}$, ρ is $\equiv_{\mathcal{V}}$ and $\preceq$ is an ordering preferring concepts which are short. In fact, this is an instance of Baader's minimal rewriting problem.[18] Assuming that Baader's algorithms could be generalised to deal with $\mathcal{ALEN}$, rather than just $\mathcal{ALE}$ or $\mathcal{ALN}$, we would gain both an (expensive) algorithm for filtering guaranteeing optimality and a new heuristic algorithm with reasonable computational properties.

8. Conclusions and further work

8.1. Achievements

Generating subsumers for natural language presentation is a novel kind of non-standard inference for Description Logics, which has been formalised and for which an algorithm with acceptable computational properties has been developed. The algorithm presented here could be incorporated into tools for manipulating ontologies, for instance to enable someone to explore the implications of possible changes to an ontology.

Our work shows that generating subsumers for natural language presentation is computationally feasible, but that it is essential to take into account principles of natural language presentability. We have begun to tease out some of the relevant principles, at least for ontologies, expressing some ideas in terms of the Gricean maxims. This is an important demonstration that Natural Language Directed Inference can be made to work, and in a relatively general way.

Although we have motivated NLDI through our own particular content determination problem, it may be useful in general to view content determination as a form of inference from a logical theory. At some level, the concepts developed here may therefore transfer to other NLG applications. However, although our approach is *domain* independent, it is certainly dependent to some extent on the nature of the *application*, in particular only addressing "bottom–up" content determination. Whether there are also particular features of ontologies that significantly colour this application also needs to be investigated.

8.2. Further work

The work presented in this paper does not present a complete solution to generating natural language presentations from ontologies. Although we have tackled important aspects of content determination and we could in principle use the approach in a system such as ILEX, we have not addressed how to prioritise the material to be presented (if many subsumers are generated), how to order or structure this material or how to realise the material as text. These problems all need to be addressed in further work. A major limitation is the way that we have simplified the reasoning about concept/linguistic complexity—although we have captured important broad aspects of linguistic adequacy, we have

[18] Our situation is somewhat unusual, in that we still have an interesting problem when $\mathcal{V} = \emptyset$. In this situation, Baader's approach reduces to just using his *reduction* algorithm.

been forced to approximate linguistic complexity by a simple syntactic measure of concept complexity. It is obvious that linguistic complexity depends on the available lexicon (e.g. whether an atomic concept can be realised by a noun, an adjective or only in a more complex way) and the nature of the realisation relation in important ways that we have ignored.

Our approach to generating subsumers has been evaluated in terms of computational feasibility, but (largely because we do not have a complete NLG system) no evaluation with actual users has taken place. Thus although there are plausible arguments for the optimisations we make, we cannot yet guarantee that we are indeed selecting the most useful and informative content. It is important that experiments with human subjects are used to evaluate the assumptions of the algorithm, in terms of how well it supports people effectively completing ontology-related tasks in some application.

In terms of the details of the methods developed, one problem that needs to be addressed arises from our use of a normal form for $\mathcal{ALEN}$ concepts. This means that the concepts enumerated are not necessarily in the best form for realisation (or perhaps that realisation should be able to choose between different ways of "undoing" the normal form conversion).

The issue of incompleteness also needs to be addressed, e.g. for applications which are safety critical. Although many subsumers not generated (e.g. those containing trivial subconcepts and empty universals) could not conceivably be useful in any application, nevertheless the complexity limit and the limited expressive power of $\mathcal{ALEN}$ could mean that important information is not expressed (though weaker versions of it might be). How could one detect that something important has been missed out, and how could one then indicate this, given that natural language presentation may be inappropriate?

A related problem is that of "horizon effects". If a target is subsumed by $(\exists P.A \sqcap B \sqcap C)$ and the complexity limit is set to 2, for instance, then the search procedure generates all of the following as candidates:

$$(\exists P.A \sqcap B)$$
$$(\exists P.A \sqcap C)$$
$$(\exists P.B \sqcap C)$$

because the optimal concept is just beyond the limits of the search space (of course, one of these candidates will be filtered out subsequently).

8.3. Extension to other rhetorical predicates

Although we have spent some time describing the generation of propositions that could be used to fill *identification, attributive* or *equivalent* slots in a McKeown-style schema, we believe that the same approach may be productive for other rhetorical predicates. Again, many of these could be formalised in DL terms—a *comparison* between A and B could perhaps involve something that system-subsumes A but not B, and *constituency* of A could involve subsumees of A. Fig. 10 shows a generalised version of the search algorithm where, apart from on-path, we have introduced two new explicit tests, the three tests being defined generally as follows:

```
candidates($x$):
    if try-descendents($x$) then
        $s = \{y \mid \exists x'(x \searrow\searrow x') \wedge \text{on-path}(x') \wedge y \in \text{candidates}(x')\}$
        if $s = \emptyset \wedge \text{return-solution}(x)$ then return $\{x\}$
        else return $s$
        endif
    elseif return-solution($x$) then return $\{x\}$
    else return $\emptyset$
    endif
return $\textit{candidates}(\top)$
```

Fig. 10. Generalised search algorithm.

on-path(x): Returns true for everything on the path from $\top$ to a candidate, including the candidate.
try-descendents(x): For any x satisfying on-path(x), returns true if it is worthwhile trying descendents of x as possible candidates (if any of them are returned then x will not be returned).
return-solution(x): Given that on-path(x) succeeds and either try-descendents(x) fails or there are no solutions in descendents, returns true if x is a solution.

Possible definitions for these tests given different reasoning tasks are shown below:

To find:	on-path(x)	try-descendents(x)	return-solution(x)
Most specific non-conjunctive subsumers of A	$A \sqsubseteq_{\mathcal{T}} x \wedge \lvert x\rvert \leqslant lim \wedge x$ is not A	*true*	*true*
Most general subsumers of A that don't subsume B	$A \sqsubseteq_{\mathcal{T}} x \wedge \lvert x\rvert \leqslant lim$	$B \sqsubseteq_{\mathcal{T}} x$	$B \not\sqsubseteq_{\mathcal{T}} x$
Most general non-conjunctive subsumees of A	$A \sqcap x \not\sqsubseteq_{\mathcal{T}} \bot \wedge \lvert x\rvert \leqslant lim \wedge x$ is not A	$x \not\sqsubseteq_{\mathcal{T}} A$	$x \sqsubseteq_{\mathcal{T}} A$

The feasibility of extending the approach to deal with these new cases remains to be evaluated.

Acknowledgements

Thanks to Racer Systems GmbH for free use of RacerPro, Jan Wielemaker for SWI Prolog, and Zhisheng Huang and Cees Visser for their DIG interface for Prolog. Many thanks to Holger Wache and Frank van Harmelen for useful discussions, and to three anonymous reviewers for valuable comments. Chris Mellish's contribution to this work is funded by EPSRC grant GR/S62932/01. Jeff Z. Pan's contribution is partially funded by the FP6 Network of Excellence EU project Knowledge Web (IST-2004-507842).

Appendix A. Example ontology

The following is a full listing of the axioms of the ontology used for our running example. This is a version (unchanged by us) of a fuel cell ontology obtained from http://metadata.net/sunago/fusion/fusion.owl and dated 8th September 2003. It should be emphasised that this is used purely as an example of a real ontology under development, and that no endorsement or criticism of the ontology is implied.

$FuelCell \sqsubseteq Actuality$
$FuelCell \sqsubseteq (\forall contains.MEA)$
$MEA \sqsubseteq Actuality$
$MEA \sqsubseteq (\exists contains.Electrolyte)$
$MEA \sqsubseteq (\exists contains.Anode)$
$MEA \sqsubseteq (\exists contains.Cathode)$
$MEA \sqsubseteq (\leqslant 1 contains)$
$Electrolyte \sqsubseteq Actuality$
$Electrode \sqsubseteq Actuality$
$Electrode \sqsubseteq (\exists contains.Catalyst)$
$Electrode \sqsubseteq ((\exists contains.Support) \sqcap (\leqslant 1 contains))$
$Support \sqsubseteq Actuality$
$Catalyst \sqsubseteq Actuality$
$Catalyst \sqsubseteq (\exists contains.ActiveMetal)$
$Catalyst \sqsubseteq (\exists contains.CatalystSupport)$
$CatalystSupport \sqsubseteq Actuality$
$ActiveMetal \sqsubseteq Actuality$
$Anode \sqsubseteq Electrode$
$Cathode \sqsubseteq Electrode$
$Dimension \sqsubseteq Abstraction$
$Conductivity \sqsubseteq Dimension$
$Conductivity \sqsubseteq (\forall value.float)$
$Cost \sqsubseteq Dimension$
$Cost \sqsubseteq (\forall value.float)$

$Composition \sqsubseteq Dimension$
$Composition \sqsubseteq (\forall value.string)$
$Density \sqsubseteq Dimension$
$Density \sqsubseteq (\forall value.float)$
$Density \sqsubseteq (\geqslant 1 context)$
$Efficiency \sqsubseteq Dimension$
$Efficiency \sqsubseteq (\forall value.float)$
$Lifetime \sqsubseteq Dimension$
$Lifetime \sqsubseteq (\forall value.float)$
$NearestNeighbour \sqsubseteq Dimension$
$NearestNeighbour \sqsubseteq (\forall value.float)$
$OperatingVoltage \sqsubseteq Dimension$
$OperatingVoltage \sqsubseteq (\forall value.float)$
$Porosity \sqsubseteq Dimension$
$Porosity \sqsubseteq (\forall value.float)$
$Porosity \sqsubseteq (\forall unit.string)$
$Shape \sqsubseteq Dimension$
$Shape \sqsubseteq (\forall value.float)$
$Size \sqsubseteq Dimension$
$Size \sqsubseteq (\forall value.float)$
$Size \sqsubseteq (\forall unit.string)$
$SpecificPower \sqsubseteq Dimension$
$SpecificPower \sqsubseteq (\forall value.float)$
$SurfaceArea \sqsubseteq Dimension$
$SurfaceArea \sqsubseteq (\forall value.float)$
$SurfaceRoughness \sqsubseteq Dimension$
$SurfaceRoughness \sqsubseteq (\forall value.float)$
$Type \sqsubseteq Dimension$
$Type \sqsubseteq (\forall value.string)$
$Voltage \sqsubseteq Dimension$
$Voltage \sqsubseteq (\forall value.float)$
$Width \sqsubseteq Dimension$
$Width \sqsubseteq (\forall value.float)$
$Width \sqsubseteq (\forall unit.string)$
$distribution \sqsubseteq_r statisticalQuantifier$
$mean \sqsubseteq_r statisticalQuantifier$
$percentage \sqsubseteq_r statisticalQuantifier$
$standardDeviation \sqsubseteq_r statisticalQuantifier$
$composition \sqsubseteq_r dimension$
$conductivity \sqsubseteq_r dimension$
$cost \sqsubseteq_r dimension$
$density \sqsubseteq_r dimension$
$efficiency \sqsubseteq_r dimension$
$lifetime \sqsubseteq_r dimension$
$nearestNeighbour \sqsubseteq_r dimension$
$operatingVoltage \sqsubseteq_r dimension$
$porosity \sqsubseteq_r dimension$
$shape \sqsubseteq_r dimension$
$size \sqsubseteq_r dimension$
$specificPower \sqsubseteq_r dimension$
$surfaceArea \sqsubseteq_r dimension$
$surfaceRoughness \sqsubseteq_r dimension$
$type \sqsubseteq_r dimension$
$voltage \sqsubseteq_r dimension$
$width \sqsubseteq_r dimension$
$thickness \equiv_r width$
$inverse(isPartOf, contains)$

$domain(contains, (FuelCell \sqcup MEA \sqcup Electrode \sqcup Catalyst))$
$domain(value, Dimension)$
$domain(unit, Dimension)$
$domain(context, (Density \sqcup Conductivity))$
$domain(statisticalQuantifier, Dimension)$
$domain(distribution, Porosity)$
$domain(percentage, Porosity)$
$domain(dimension, Actuality)$
$domain(composition, (ActiveMetal \sqcup CatalystSupport \sqcup Support \sqcup Electrolyte))$
$domain(conductivity, (Electrode \sqcup Electrolyte))$
$domain(cost, FuelCell)$
$domain(density, (Catalyst \sqcup Electrode))$
$domain(efficiency, FuelCell)$
$domain(lifetime, FuelCell)$
$domain(nearestNeighbour, (ActiveMetal \sqcup CatalystSupport \sqcup Catalyst \sqcup Support \sqcup Electrode))$
$domain(operatingVoltage, FuelCell)$
$domain(porosity, (CatalystSupport \sqcup Electrode \sqcup Electrolyte))$
$domain(shape, (ActiveMetal \sqcup Catalyst))$
$domain(size, (ActiveMetal \sqcup CatalystSupport \sqcup Catalyst))$
$domain(specificPower, FuelCell)$
$domain(surfaceArea, (Catalyst \sqcup Electrode))$
$domain(surfaceRoughness, (ActiveMetal \sqcup Catalyst \sqcup Electrode))$
$domain(type, FuelCell)$
$domain(voltage, FuelCell)$
$domain(width, (Electrode \sqcup Electrolyte))$
$domain(degree, NearestNeighbour)$
$range(contains, Actuality)$
$range(context, Actuality)$
$range(statisticalQuantifier, Dimension)$
$range(distribution, Porosity)$
$range(percentage, Porosity)$
$range(dimension, Dimension)$
$range(composition, Composition)$
$range(conductivity, Conductivity)$
$range(cost, Cost)$
$range(density, Density)$
$range(efficiency, Efficiency)$
$range(lifetime, Lifetime)$
$range(nearestNeighbour, NearestNeighbour)$
$range(operatingVoltage, OperatingVoltage)$
$range(porosity, Porosity)$
$range(shape, Shape)$
$range(size, Size)$
$range(specificPower, SpecificPower)$
$range(surfaceArea, SurfaceArea)$
$range(surfaceRoughness, SurfaceRoughness)$
$range(type, Type)$
$range(voltage, Voltage)$
$range(width, Width)$
$range(degree, integer)$
$functional(value)$
$functional(unit)$
$functional(context)$
$functional(statisticalQuantifier)$
$functional(density)$
$functional(width)$
$functional(degree)$

Appendix B. Example ontologies and classes

The concepts and ontologies from the "medium sized" test set were as follows. #C and #P indicate the number of named classes and properties in the ontology.

No	Ontology	#C	#P	Concept
1	http://siul02.si.ehu.es/Aingeru/OperOnt.owl.txt	117	17	DumpLocation
2	http://siul02.si.ehu.es/Aingeru/OperOnt.owl.txt	117	17	LocationQuery
3	http://sweet.jpl.nasa.gov/ontology/earthrealm.owl.txt	248	23	ActiveLayer
4	http://sweet.jpl.nasa.gov/ontology/earthrealm.owl.txt	248	23	LandSurface
5	http://sweet.jpl.nasa.gov/ontology/property.owl.txt	277	9	CropMoistureIndex
6	http://sweet.jpl.nasa.gov/ontology/property.owl.txt	277	9	Productivity
7	http://sweet.jpl.nasa.gov/ontology/substance.owl.txt	327	16	Methane
8	http://sweet.jpl.nasa.gov/ontology/substance.owl.txt	327	16	SnowFacies
9	http://sweet.jpl.nasa.gov/sweet/phenomena.owl#.txt	253	17	TidalWave
10	http://sweet.jpl.nasa.gov/sweet/phenomena.owl#.txt	253	17	BaroclinicCirculation
11	http://www.biopax.org/release/biopax-level1.owl.txt	30	49	bioSource
12	http://www.biopax.org/release/biopax-level1.owl.txt	30	49	catalysis
13	http://www.co-ode.org/ontologies/amino-acid/2005/10/11/amino-acid.owl.txt	47	6	Aliphatic
14	http://www.co-ode.org/ontologies/amino-acid/2005/10/11/amino-acid.owl.txt	47	6	E
15	http://www.cs.man.ac.uk/$\sim$horrocks/OWL/Ontologies/ka.owl.txt	98	92	PSMlibraries
16	http://www.cs.man.ac.uk/$\sim$horrocks/OWL/Ontologies/ka.owl.txt	98	92	Conference
17	http://www.loa-cnr.it/Files/DLPOnts/Plans/397.owl.txt	64	55	goal-situation
18	http://www.loa-cnr.it/Files/DLPOnts/Plans/397.owl.txt	64	55	decision-activity

The concepts and ontologies from the "large" test set were as follows:

No	Ontology	Concept
1	http://loki.cae.drexel.edu/~wbs/ontology/2004/08/fgdc-csdgm.owl.txt	Other_Grid_Systems_Definition
2	http://loki.cae.drexel.edu/~wbs/ontology/2004/08/fgdc-csdgm.owl.txt	Spatial_Data_Organization_Information
3	http://mged.sourceforge.net/ontologies/MGEDOntology.owl.txt	PhysicalArrayDesign
4	http://mged.sourceforge.net/ontologies/MGEDOntology.owl.txt	SequenceOntologyBioSequenceType
5	http://protege.stanford.edu/plugins/owl/owl-library/MGEDOntology.owl.txt	InitialTimePoint
6	http://protege.stanford.edu/plugins/owl/owl-library/MGEDOntology.owl.txt	IndividualChromosomalAbnormality
7	http://protege.stanford.edu/plugins/owl/owl-library/resume.owl.txt	LeadingActivity
8	http://protege.stanford.edu/plugins/owl/owl-library/resume.owl.txt	ReducedCasts
9	http://www.co-ode.org/ontologies/pizza/pizza/20041007.owl.txt	OliveTopping
10	http://www.co-ode.org/ontologies/pizza/pizza/20041007.owl.txt	ThinAndCrispyBase
11	http://www.loa-cnr.it/Files/DLPOnts/DOLCE-Lite/397.owl.txt	spatio-temporal-region
12	http://www.loa-cnr.it/Files/DLPOnts/DOLCE-Lite/397.owl.txt	accomplishment
13	http://www.loa-cnr.it/Files/DLPOnts/ExtDnS/397.owl.txt	constitutive-description
14	http://www.loa-cnr.it/Files/DLPOnts/ExtDnS/397.owl.txt	reconstructed-flux
15	http://xobjects.seu.edu.cn/resource/drm/orel/orel0/5.owl#.txt	TimeOfDestroying
16	http://xobjects.seu.edu.cn/resource/drm/orel/orel0/5.owl#.txt	PlaceOfMovingThrough

Concepts	#C	#P
1, 2	172	302
3, 4	232	108
5, 6	216	97
7, 8	170	47
9, 10	85	3
11, 12	37	70
13, 14	71	134
15, 16	200	40

References

[1] J. Allen, Natural Language Understanding, Benjamin Cummings, 1995.

[2] F. Baader, R. Küsters, A. Borgida, D. McGuinness, Matching in description logics, Journal of Logic and Computation 9 (3) (1999) 411–447.

[3] F. Baader, R. Küsters, R. Molitor, Rewriting concepts using terminologies—revisited, Tech. Rep. 00-04, Aachen University of Technology, Research Group for Theoretical Computer Science, Aachen, 2000.

[4] F. Baader, B. Sertkaya, A.-Y. Turhan, Computing the least common subsumer w.r.t. a background terminology, Journal of Applied Logic (2007).

[5] P. Blackburn, J. Bos, M. Kohlhase, H. de Nivelle, Inference and computational semantics, in: Bunt, Muskens, Thijsse (Eds.), Computing Meaning, vol. 2, Kluwer, 2001, pp. 11–28.

[6] S. Brandt, R. Küsters, A. Turhan, Approximation and difference in description logics, in: Proc. of the 8th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'2002), 2002, pp. 203–214.

[7] J. Bruner, J. Goodnow, G. Austin, A Study of Thinking, John Wiley, 1956.

[8] A. Bundy, B. Welham, Using meta-level inference for selective application of multiple rewrite rule sets in algebraic manipulation, Artificial Intelligence 16 (2) (1981) 111–224.

[9] T. Catarci, P. Dongilli, T.D. Mascio, E. Franconi, G. Santucci, S. Tessaris, An ontology based visual tool for query formulation support, in: Proc. of the 16th European Conference on Artificial Intelligence, ECAI 2004, Valencia, Spain, 2004.

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

[11] J. Davies, A. Duke, D. Mladenič, K. Bontcheva, M. Grčar, R. Benjamins, J. Contreras, M.B. Civico, T. Glover, Next generation knowledge access, Journal of Knowledge Management 9 (5) (2005) 64–84.

[12] S. Džeroski, N. Lavrač, Refinement graphs for FOIL and LINUS, in: S. Muggleton (Ed.), Inductive Logic Programming, Academic Press, 1992.

[13] T. Gaasterland, P. Godfrey, J. Minker, An overview of cooperative answering, Intelligent Information Systems 1 (2) (1992) 123–157.

[14] C. Gardent, E. Jacquey, Lexicalisation as a description logic inference task, in: Proc. of ICOS'03 (Inference in Computational Semantics), Nancy, France, 2003.

[15] T. Gardiner, I. Horrocks, D. Tsarkov, Automatic benchmarking of description logic reasoners, in: Proceedings of the 2006 International Workshop on Description Logics (DL06), Windermere, UK, 2006.

[16] G. Gazdar, Pragmatics: Implicature, Presupposition, and Logical Form, Academic Press, 1979.

[17] H.P. Grice, Logic and conversation, in: P. Cole, J. Morgan (Eds.), Syntax and Semantics: vol. 3, Speech Acts, Academic Press, 1975.

[18] T.R. Gruber, Towards principles for the design of ontologies used for knowledge sharing, in: N. Guarino, R. Poli (Eds.), Formal Ontology in Conceptual Analysis and Knowledge Representation, Kluwer Academic Publishers, Deventer, The Netherlands, 1993.

[19] V. Haarslev, R. Möller, RACER system description, in: R. Goré, A. Leitsch, T. Nipkow (Eds.), Proc. of the International Joint Conference on Automated Reasoning (IJCAR'2001), Springer Verlag, 2001.

[20] L. Horn, On the semantic properties of logical operators in English, Ph.D. thesis, University of California, LA, 1972.

[21] I. Horrocks, Using an expressive description logic: Fact or fiction? in: Proc. of the 6th International Conference on Principles of Knowledge Representation and Reasoning (KR'98), 1998, pp. 636–647.

[22] E. Hovy, Pragmatics and natural language generation, Artificial Intelligence 43 (1990) 153–197.

[23] Z. Huang, C. Visser, An extended DIG description logic interface for prolog, Tech. rep. SEKT/2004/D3.4.1.2/v1.0, Department of Artificial Intelligence, Vrije Universiteit Amsterdam, 2004.

[24] A. Kalyanpur, C. Halaschek-Wiener, V. Kolovski, J. Hendler, Effective NL paraphrasing of ontologies on the semantic web, Tech. rep., Department of Computer Science, University of Maryland, 2005.

[25] H. Knublauch, R.W. Fergerson, N.F. Noy, M.A. Musen, The Protégé OWL plugin: An open development environment for semantic web applications, in: Proc. of the Third International Semantic Web Conference, Hiroshima, Japan, 2004.

[26] D. McGuinness, A. Borgida, Explaining subsumption in description logics, in: Proc. of the 14th International Joint Conference on Artificial Intelligence. Montreal, 1995, pp. 816–821.

[27] D.L. McGuinness, F. van Harmelen, Owl web ontology language overview, http://www.w3.org/TR/owl-features/, 2004.

[28] K.R. McKeown, Text Generation: Using Discourse Strategies and Focus Constraints to Generate Natural Language Text, Cambridge University Press, Cambridge, 1985.

[29] C. Mellish, M. O'Donnell, J. Oberlander, A. Knott, An architecture for opportunistic text generation, in: Proc. of the Ninth International Workshop on Natural Language Generation, Association for Computational Linguistics, 1998, pp. 28–37.

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

[31] C. Mellish, X. Sun, Natural language directed inference in the presentation of ontologies, in: Proc. of the 10th European Workshop on Natural Language Generation, Aberdeen, 2005, pp. 118–124.

[32] M. Meteer, Expressibility and the Problem of Efficient Text Planning, Pinter publishers, London, 1992.

[33] G. Miller, S.D. Isard, Free recall of self-embedded English sentences, Information and Control 7 (1964) 292–303.

[34] J. Moore, Participating in Explanatory Dialogues, MIT Press, 1994.

[35] M. O'Donnell, A. Knott, C. Mellish, J. Oberlander, *ILEX*: The architecture of a dynamic hypertext generation system, Natural Language Engineering 7 (2001) 225–250.

[36] J.Z. Pan, C. Mellish, Supporting semi-automatic semantic annotation of multimedia resources, in: Proc. of the 3rd IFIP Conference on Artificial Intelligence Applications and Innovations, Athens, 2006, pp. 609–617.

[37] S.G. Pulman, Grammars, parsers and memory limitations, Language and Cognitive Processes 1 (3) (1986) 197–225.

[38] E. Reiter, A new model of lexical choice for nouns, Computational Intelligence 7 (1990) 240–251.

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

[40] N. Shadbolt, W. Hall, T. Berners-Lee, The semantic web revisited, IEEE Intelligent Systems 21 (3) (2006) 96–101.

[41] J. Shaw, Conciseness through aggregation in text generation, in: Proc. of the 33rd Annual Meeting of the Association for Computational Linguistics, MIT, 1995.

[42] S. Shieber, The problem of logical-form equivalence, Computational Linguistics 19 (1) (1993) 179–190.

[43] S. Sripada, E. Reiter, J. Hunter, J. Yu, Generating English summaries of time series data using the Gricean maxims, in: Proc. of the Ninth ACM SIGMOD International Conference on Knowledge Discovery and Data Mining (KDD-2003), 2003, pp. 187–196.

[44] D. Tsarkov, A. Riazanov, S. Bechhofer, I. Horrocks, Using Vampire to reason with OWL, in: S.A. McIlraith, D. Plexousakis, F. van Harmelen (Eds.), Proc. of the 2004 International Semantic Web Conference (ISWC 2004), in: LNCS, vol. 3298, Springer, 2004, pp. 471–485.

[45] M. Uschold, M. Gruninger, Ontologies: Principles, Methods and Applications, The Knowledge Engineering Review, 1996.

[46] P. Wason, The contexts of plausible denial, Journal of Verbal Learning and Verbal Behaviour 4 (1965) 7–11.

[47] G. Wilcock, Talking owls: Towards an ontology verbalizer, in: Human Language Technology for the Semantic Web and Web Services, ISWC-2003, Sanibel Island, Florida, 2003, pp. 109–112.

[48] R.M. Young, Using Grice's maxim of quantity to select the content of plan descriptions, Artificial Intelligence 115 (1999) 215–256.