Finding Subsumers for Natural Language Presentation

Chris Mellish and Jeff Z. Pan
Department of Computing Science
University of Aberdeen

Abstract

This work is motivated by the task of describing in natural language a concept defined in an OWL DL ontology. However, rather than focussing on linguistic issues, we address the question of how to support natural language presentation with inference. We introduce a new non-standard DL reasoning problem, that of finding subsumers of a concept that are suitable for natural language presentation. We present a solution that works by enumerating successively more complex concepts in the limited language $\mathcal{ALEN}$. Although the search space is formidable, specific optimisations that take into account characteristics of natural language enable it to be tamed. Our initial experiments show that the approach may be quite feasible in practice.

1 Introduction

Knowledge engineers, domain experts and also casual users need better ways to understand ontologies. As the logical structure of ontologies becomes richer, it becomes harder to devise appropriate graphical means of presentation 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 some complex logical structures and requires no special training.

The work described in this paper takes as its starting point the task of answering in natural language a question *What is A?*, where A (the *target*) is an atomic concept mentioned in some given OWL DL ontology. This may take place as a part of a longer dialogue between person and machine where, for instance, subsequently the person asks *What is B?*, for some atomic B mentioned in the answer to the first question. Rather than considering detailed linguistic aspects of this task, however, we focus on how to support it with appropriate reasoning.

A first attempt at the task of answering *What is A?* might somehow render in natural language the set of ontology axioms that mention A. However, an ontology axiom is not necessarily of appropriate complexity to be expressed as a natural language sentence. Also, it could be misleading to present true, but incomplete information. Finally, important information about the target may arise from logical consequences of the axioms, not only

from explicitly stated axioms. Elsewhere we have used these arguments to argue the need for new kinds of inference, *natural language directed inference* (NLDI), which are capable of deriving those logical consequences suitable for natural language presentation [8].

DL-based Ontologies have available powerful reasoning services, such as classification, subsumption and satisfiability checking. Standard reasoning services, 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, standard DL reasoning services cannot be used immediately to implement NLDI. The challenge is to exploit these efficiently implemented services in more complex ways to make NLDI possible.

What kind of information is needed to answer a question *What is A?* McKeown [7] discusses a number of kinds of facts present in human descriptions of a target A, which include **identification**: *An aircraft carrier is a surface ship*, **attributive**: *A torpedo has an underwater target location*, and **equivalent**: *Wines described as 'great' are fine wines from an especially good village*. In DL terms, these correspond to concept subsumptions $A \sqsubseteq C$ w.r.t. a TBox $\mathcal{T}$, with A (A= $AircraftCarrier$, $Torpedo$, $GreatWine$) being the subsumee. Unfortunately, not all (or even any) of these concept descriptions C need necessarily appear explicitly directly in axioms of the form "$A \sqsubseteq C$" in the ontology.

In this paper we present a procedure for discovering subsumers C of a concept A that might be worth presenting in natural language. Although we allow the ontology $\mathcal{T}$ to be expressed in full OWL DL, nevertheless the subsumers C are in the more limited language $\mathcal{ALEN}$. In a sense, therefore, our real goal is to produce the most specific $\mathcal{ALEN}$ subsumer, which contains all the information known about the target. In general, however, this will be a conjunction, within which the conjuncts could be generated in many possible orders. Instead of generating the single conjunction, therefore, we generate a set of most specific non-conjunctive subsumers, which could then be combined together by conjunction if this was wished. We require these individual conjuncts to be the sort of things that could be presented in individual natural language sentences.

The closest related work is that on non-standard inferences in DLs. On the one hand, our task could be characterised as looking for a least subsumer (other than A itself) of A using a less expressive DL [3]. On the other hand, the task can be regarded as a "matching" problem, "$A \sqsubseteq^? P$", where P is a concept pattern [2]. Unfortunately, existing approaches to both computing least subsumers and matching only apply to less expressive DLs and assume TBoxes to be unfoldable. In this work, apart from assuming the existence of standard reasoning services, we do not assume unfoldability of the axioms.

2 Discovering Subsumers

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 two different sets of beliefs – the system's beliefs and 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* $\mathcal{U}$. The user KB is likely to be different from the system KB because otherwise the user would not have sought information

about the target. Hence two notions of subsumption arise:[1]

1. C_1 **system-subsumes** C_2 iff $\mathcal{T} \models C_2 \sqsubseteq C_1$;
2. C_1 **user-subsumes** C_2 iff $\mathcal{U} \models C_2 \sqsubseteq C_1$.

We make the assumption that the user KB has the same vocabulary as the system KB and is an approximation to the system KB:

For all C_1, C_2, if $\mathcal{U} \models C_1 \sqsubseteq C_2$ then $\mathcal{T} \models C_1 \sqsubseteq C_2$.

I.e. everything the user knows is also known by the system. Our framework allows for any $\mathcal{U}$ satisfying the above constraints; in particular, one could have $\mathcal{U} = \mathcal{T}$ (ignoring this particular distinction) or have $\mathcal{U}$ be something that is built up over the dialogue as a result of the information that the user is told. Our current implementation considers just the first question of a dialogue, where the user has no initial domain knowledge, i.e. $\mathcal{U} = \phi$.

We can now present the task more formally:[2]

> **The natural language subsumer enumeration problem:** Given a system-satisfiable target named concept A, find the most specific (w.r.t. user-subsumption) non-conjunctive concepts C which system-subsume A, do not user-subsume A and are appropriate for natural language presentation.

Here we use the user KB to decide which of these are worth presenting, because the extra knowledge of the system may obscure certain user-relevant distinctions. For example, consider an ontology $\mathcal{T}$ which includes the axiom $C_1 \equiv C_2$ not in $\mathcal{U}$. Given this knowledge, if C_1 subsumes a target A, so does C_2. As C_1 and C_2 are system-equivalent, from the system's point of view the choice of which to present is arbitrary. From the point of view of the user who is not in possession of all the knowledge in $\mathcal{T}$, however, the descriptions C_1 and C_2 provide distinct information, and so it is worth considering presenting *both* of them.

Our approach is to enumerate concepts C subsuming A via a search through all possible concepts in $\mathcal{ALEN}$. At each stage we can test whether an enumerated concept C system-subsumes A using the subsumption reasoning service. Some such concepts are returned as *candidates* which may be user-least non-conjunctive subsumers of A. The set of candidates is then further filtered, in order to obtain a set, no element of which user-subsumes another element or user-subsumes the target (exactly one of a set of user-equivalent concepts is returned). In practice, the enumeration of candidates is organised in such a way as to avoid many candidates that would otherwise be filtered out by the second step.

3 The Refinement Relation

The search space is expressed 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. The

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

[2] We consider extra requirements for natural language presentation in Section 5.

search starts from the most general concept $\top$, working to candidates C_1 such that $\top \searrow C_1$, then to candidates 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, concepts are assumed to be in a normal form, so that conjunctions are only allowed inside $\exists$ constructs, and conjunctive information at the top level or just inside $\forall$ constructs must be expanded out to yield multiple candidates. Nested conjunctions are flattened. Within conjunctions, conjuncts (if present) occur in the following order: negations (in lexicographic order) before role restrictions (with properties in a fixed order) before atomic concepts (in lexicographic order). Within the role restrictions, all the restrictions for a given role occur together, in the order: number restrictions before $\forall$ before $\exists$. For any role P, there are at most two number restrictions: either a single $=$ restriction or at most one of each of $\leq$ and $\geq$, in this order. For any role P, at most one $\forall P$ restriction can occur within any allowed conjunction.

The following exhaustive definition can also be read as the basis of an algorithm for enumerating, for a given concept α, the concepts β such that $\alpha \searrow \beta$. This gives us the original theoretical search space. In the actual implementation, we make a number of optimisations compared to using the basic refinement relation, as detailed below.

$\top \searrow A_i$ if A_i is a named concept
$\top \searrow \neg A_i$ if A_i is a named concept
$\top \searrow (\exists P.\top)$ if P is a role name
$\top \searrow (\forall P.\alpha)$ if P is a role name and $\top \searrow \alpha$
$\top \searrow (\geq nP)$ if P is a simple role and $0 \leq n \leq \pi$, where π a large number (1000000)
$\top \searrow (\leq nP)$ if $(0 \leq n \leq \pi)$, and P is a simple role
$\top \searrow (= nP)$ if $(0 \leq n \leq \pi)$, and P is a simple role
$(\exists P.\alpha) \searrow (\exists P.\beta)$ if $\alpha \searrow \beta$
$(\forall P.\alpha) \searrow (\forall P.\beta)$ if $\alpha \searrow \beta$
$\alpha \searrow (\beta \sqcap \alpha)$ where α is not a conjunction, $\top \searrow \beta$, this is within the scope of an $\exists$ and β is of a type allowed before α by the conjunction ordering rules.
$\alpha_1 \sqcap \alpha_2 \sqcap \ldots \alpha_n \searrow \beta \sqcap \alpha_2 \sqcap \ldots \alpha_n$ if $\alpha_1 \searrow \beta$
$\alpha_1 \sqcap \alpha_2 \sqcap \ldots \alpha_n \searrow \beta \sqcap \alpha_1 \sqcap \alpha_2 \sqcap \ldots \alpha_n$ if $\top \searrow \beta$ and β is of a type allowed before the α_i by the conjunction ordering rules.

The above relation $\searrow$ has the following properties. These can be shown by induction on the number of symbols (other than $\top$) occurring in C_1.

Lemma 1. If C_1 is a satisfiable concept in $\mathcal{ALEN}$ expressed in terms of the vocabulary of the ontology then $\top \searrow^* \beta$ for some concept β logically equivalent to C_1.

Lemma 2. If $C_1 \searrow C_2$ then C_2 is strictly more syntactically complex then C_1 (for a range of possible complexity metrics)

Lemma 3. If $C_1 \searrow C_2$ then C_1 subsumes C_2 (hence C_1 system- and user-subsumes C_2)

The first of these means that we can reach all possible concepts through $\searrow$ (notice that we do not need to allow $\bot$ in formulae, because of equivalences such as $(\forall P.\bot) \equiv (= 0P)$). 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.

4 The Search Strategy

The search algorithm works with a derived relation $\searrow\!\!\searrow$, defined in terms of $\searrow$, which produces results strictly user-subsumed by the original concept. Again, this definition can be thought of as an algorithm to enumerate the relevant refinements:

$C_1 \searrow\!\!\searrow C_2$ iff $C_1 \searrow C_2$ and C_2 does not user-subsume C_1
or $\exists C_3.C_1 \searrow C_3$, C_3 user-subsumes C_1 and $C_3 \searrow\!\!\searrow C_2$

Our search is organised in a depth-first manner. If a point in the tree is reached where the concept C_i does not system-subsume the target, there is no point in considering further refinements of this concept. By Lemma 3, such further refinements will be subsumed, and hence also system-subsumed, by C_i. One of these cannot system-subsume the target, because if so then, by transitivity of system-subsumption, C_i would have to as well. If such a C_i is reached, search down that path of the tree is terminated. Lemma 2 means that we can achieve termination by terminating the search path when it reaches a concept with a complexity equal to or exceeding a preset limit. A generated concept is returned as a candidate exactly when none of the first $\searrow\!\!\searrow$ descendents both are of acceptable size (see Section 5) and also system-subsume the target. This tends to lead to only user-least solutions being returned.

The above properties guarantee that the above search is complete, in that all user-least non-conjunctive concepts (or concepts logically equivalent to them) that system-subsume the target and have a size below the limit are enumerated (as well as possibly some other concepts). Logically equivalent solutions could, however, be generated many times. The search is partially correct, in that all candidates system-subsume the target. User-minimality (and so the rest of correctness) is then ensured by the subsequent filtering process.

Apart from the natural language based optimisations discussed in the next section, space does not permit us to describe in detail a number of other optimisations used to enhance the basic search approach. Firstly, the relevance filter of [9] is used to limit the vocabulary of atomic concept and role names used in the enumerated concepts. Secondly, we disallow the addition of elements to conjunctions which are either user-subsumed by or user-subsume existing conjuncts. Thirdly, a focussed search is used to ensure that any introduced number restriction is in fact the most specific such restriction such that the candidate with this restriction in it subsumes the target.

5 Natural Language Direction

Natural language easily expresses conjunctive information and often produces scope ambiguities in complex examples involving disjunction and negation. $\mathcal{ALEN}$ allows no disjunctions or complex negations and thus is a natural DL to act as the target for natural language based approximation.

The following summarises other ways in which we incorporate natural language direction into the algorithm, sometimes at the expense of *logical* completeness.

Concept Complexity. Because there is a limit to the complexity of a concept that can be presented in a sentence, we impose a size limit on concepts. Each negation or conjunction in a concept counts 1 towards the "size" of a formula, and each quantifier counts $n + 1$ towards the calculation, where n is the number of enclosing quantifiers. A complexity limit of around 4 or 5 seems roughly plausible for what can give rise to a comprehensible natural language sentence.

Specificity and Complexity. Because we use $\searrow\!\!\searrow$, rather than $\searrow$ in the algorithm, refinements of a concept that are user-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 user-equivalent concepts is ever returned. The consequence is that, roughly speaking, the simplest way of saying some particular content is chosen (c.f. Grice's principle of brevity [4]).

Introducing new Terminology. Since we are interested in finding just the most specific concepts that subsume the target, whenever $\top$ is refined to a named concept it can be refined to a most user-specific named concept such that the whole candidate, with this concept substituted, system-subsumes the target. Similarly, when a concept $\neg A_i$ is introduced, it can be done with a most user-general A_i such that the candidate still system-subsumes the target. Unfortunately, if $\mathcal{U} = \phi$, then there are no non-trivial user-subsumption relationships between named concepts, and so this measure has no effect. As a result, for instance, if a target is system-subsumed by $(\exists hasPet.Poodle)$ then concepts like $(\exists hasPet.Animate)$ will also appear as candidates (none of these is user-subsumed by any other). All of these convey new information to the imagined user, but they are not all equally good for natural language presentation. Given that the user has the opportunity in the dialogue to ask followup questions about mentioned atomic concepts, it is actually complete in a dialogue sense simply to return just the concept including the *system* most specific concepts in these cases. In contrast, if all the above concepts are presented then the impression may be given that there are no noteworthy system-subsumption relationships between them.

Negations. In natural languages, negation is used primarily to *deny* an explicitly or implicitly available proposition. This means that (in the absence of some specific context) it would be strange to answer the question *What is a mammal?* with something like *Every mammal is not a mushroom.* Our interpretation of this requirement is 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 is 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.

Trivial Universals. If instances of a given concept cannot possibly have values for a role P then all $\forall$ restrictions on this role trivially subsume the concept. Such restrictions are however not appropriate to be expressed. This means that it would be strange to answer the question *What is a SpicyTopping?* with something like *A SpicyTopping can only have*

a pizza as a topping. In this example, pizza toppings cannot themselves have toppings. This means that, for instance, $\forall topping.Pizza$ 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. The same problem can arise at any nesting within a candidate. Our solution to this problem is that if at any point we are planning to insert the concept $(\forall P.\top)$ at some point 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.

6 Discussion

Our prototype considers just the first question of a question answering dialogue, where $\mathcal{U} = \emptyset$. It is implemented in SWI Prolog (version 5.4.7), using a DIG interface for Prolog [5]. RacerPro (version 1.9.0) is used via DIG to provide all reasoning services w.r.t. the system KB (e.g. system-subsumption) and results are cached. As $\mathcal{U} = \phi$, the checking of user-subsumption is implemented structurally in Prolog with simple subset of the algorithm of [6], to avoid overheads introduced by using the DIG interface.

The following table shows statistics about some examples. The two ontologies used are a food ontology from `http://www.w3.org/TR/owl-guide/food.rdf` and a pizza ontology from `http://www.co-ode.org/ontologies/pizza/pizza_20041007.owl`. For each example, we give the complexity limit, the search space size[3], the target, the number of calls to RacerPro needed, the number of subsumers found by the initial search and the number that this is filtered to.

Ontology	Limit	Search Space	Target	Calls	Subsumers	Filtered
Food	4	2.8E8	$FruitCourse$	2354	87	63
Food	5	1.8E10	$FruitCourse$	4706	124	93
Food	6	1.2E12	$FruitCourse$	13756	385	324
Pizza	4	2.1E8	$AmericanHot$	1386	80	76
Pizza	5	1.8E10	$AmericanHot$	2116	80	76
Pizza	6	1.5E12	$AmericanHot$	3970	99	95

The concepts found subsuming $AmericanHot$, assuming complexity limit 4, include:

$\neg\ VegetarianPizza \sqcap NamedPizza$
$(\forall hasTopping.\neg\ ArtichokeTopping \sqcap PizzaTopping)$
$(\forall hasTopping.(\leq 1 hasSpiciness))$
$(\exists hasTopping.(\forall hasSpiciness.Spiciness) \sqcap MozzarellaTopping)$
$(\geq 5 hasTopping)$

Our results so far show that it can be possible, given an appropriate search strategy and natural language direction, to solve the natural language subsumer enumeration problem

[3]The size of the basic refinement search space, without any optimisations, ignoring lexicographic ordering of conjuncts and assuming a maximum cardinality of 3. The implementation allows cardinalities up to 1 million, but using this figure in the calculation of the raw search space size would be misleading.

in spite of the huge search space involved. Actual runtimes are still somewhat problematic (the above calculation for *AmericanHot* with limit 5 takes about 31 seconds elapsed time on a 1695MHz PC), but further optimisations could be introduced if the reasoner and the rest of the system were in the same process.

7 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. Thanks to Holger Wache and Frank van Harmelen for useful discussions. 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).

References

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

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

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

[4] H. P. Grice. Logic and conversation. In P. Cole and J. Morgan, editors, *Syntax and Semantics: Vol 3, Speech Acts*. Academic Press, 1975.

[5] Zhisheng Huang and Cees Visser. An Extended DIG Description Logic Interface for Prolog. Technical Report SEKT/2004/D3.4.1.2/v1.0, Dept of Artificial Intelligence, Vrije Universiteit Amsterdam, 2004.

[6] R. Küsters and R. Molitor. Structural subsumption and least common subsumers in a description logic with existential and number restrictions. *Studia Logica*, 81(2):227–259, 2005.

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

[8] C. Mellish and X. Sun. Natural language directed inference in the presentation of ontologies. In *Procs of the Tenth European Workshop on Natural Language Geeration*, Aberdeen, Scotland, 2005.

[9] Dmitry Tsarkov, Alexandre Riazanov, Sean Bechhofer, and Ian Horrocks. Using vampire to reason with owl. In Sheila A. McIlraith, Dimitris Plexousakis, and Frank van Harmelen, editors, *Procs of the 2004 International Semantic Web Conference (ISWC 2004)*, pages 471–485. Springer LNCS 3298, 2004.