Representing and resolving ambiguities in ontology-based question answering

Christina Unger **Philipp Cimiano**
Cognitive Interaction Technology – Center of Excellence (CITEC),
Universität Bielefeld, Germany
{cunger|cimiano}@cit-ec.uni-bielefeld.de

Abstract

Ambiguities are ubiquitous in natural language and pose a major challenge for the automatic interpretation of natural language expressions. In this paper we focus on different types of lexical ambiguities that play a role in the context of ontology-based question answering, and explore strategies for capturing and resolving them. We show that by employing underspecification techniques and by using ontological reasoning in order to filter out inconsistent interpretations as early as possible, the overall number of interpretations can be effectively reduced by 44 %.

1 Introduction

Ambiguities are ubiquitous in natural language. They pose a key challenge for the automatic interpretation of natural language expressions and have been recognized as a central issue in question answering (e.g. in (Burger et al., 2001)). In general, ambiguities comprise all cases in which natural language expressions (simple or complex) can have more than one meaning. These cases roughly fall into two classes: They either concern *structural* properties of an expression, e.g. different parses due to alternative preposition or modifier attachments and different quantifier scopings, or they concern alternative meanings of *lexical* items. It is these latter ambiguities, ambiguities with respect to lexical meaning, that we are interested in. More specifically, we will look at ambiguities in the context of ontology-based interpretation of natural language.

The meaning of a natural language expression in the context of ontology-based interpretation is the ontology concept that this expression verbalizes. For example, the expression *city* can refer to a class `geo:city` (where `geo` is the namespace of the corresponding ontology), and the expression *inhabitants* can refer to a property `geo:population`. The correspondence between natural language expressions and ontology concepts need not be one-to-one. On the one hand side, different natural language expressions can refer to a single ontology concept, e.g. *flows through*, *crosses through* and *traverses* could be three ways of expressing an ontological property `geo:flowsThrough`. On the other hand, one natural language expression can refer to different ontology concepts. For example, the verb *has* is vague with respect to the relation it expresses – it could map to `geo:flowsThrough` (in the case of rivers) as well as `geo:inState` (in the case of cities). Such mismatches between the linguistic meaning of an expression, i.e. the user's conceptual model, and the conceptual model in the ontology give rise to a number of ambiguities. We will give a detailed overview of those ambiguities in Section 3, after introducing preliminaries in Section 2.

For a question answering system, there are mainly two ways to resolve ambiguities: by interactive clarification and by means of background knowledge and the context with respect to which a question is asked and answered. The former is, for example, pursued by the question answering system FREyA (Damljanovic et al., 2010). The latter is incorporated in some recent work in machine learning. For example, (Kate & Mooney, 2007) investigate the task of

Proceedings of the TextInfer 2011 Workshop on Textual Entailment, EMNLP 2011, pages 40–49,
Edinburgh, Scotland, UK, July 30, 2011.

learning a semantic parser from a corpus whith sentences annotated with multiple, alternative interpretations, and (Zettlemoyer & Collins, 2009) explore an unsupervised algorithm for learning mappings from natural language sentences to logical forms, with context accounted for by hidden variables in a perceptron.

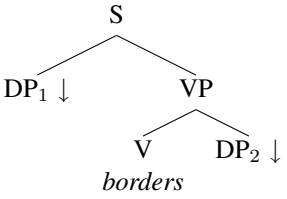
In ontology-based question answering, context as well as domain knowledge is provided by the ontology. In this paper we explore how a given ontology can be exploited for ambiguity resolution. We will consider two strategies in Section 4. The first one consists in simply enumerating all possible interpretations. Since this is not efficient (and maybe not even feasible), we will use underspecification techniques for representing ambiguities in a much more compact way and then present a strategy for resolving ambiguities by means of ontological reasoning, so that the number of interpretations that have to be considered in the end is relatively small and does not comprise inconsistent and therefore undesired interpretations. We will summarize with quantitative results in Section 5.

2 Preliminaries

All examples throughout the paper will be based on Raymond Mooney's *GeoBase*[1] dataset and the DBpedia question set published in the context of the *1st Workshop on Question Answering Over Linked Data* (QALD-1)[2]. The former is a relatively small and well-organized domain, while the latter is considerably larger and much more heterogenous. It is interesting to note that ontological ambiguituies turn out to be very wide-spread even in a small and homogenuous domain like GeoBase (see Section 3 for specific results).

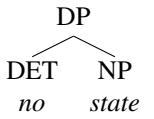
For specifying entries of a grammar that a question answering system might work with, we will use the general and principled linguistic representations that our question answering system *Pythia*[3] (Unger et al., 2010) relies on, as they are suitable for dealing with a wide range of natural language phenomena. Syntactic representations will be trees from *Lexicalized Tree Adjoining Grammar* (LTAG (Schabes, 1990)). The syntactic representation of a lexical item is a tree constituting an extended projection of that item, spanning all of its syntactic and semantic arguments. Argument slots are nodes marked with a down arrow (↓), for which trees with the same root category can be substituted. For example, the tree for a transitive verb like *borders* looks as follows:

1.



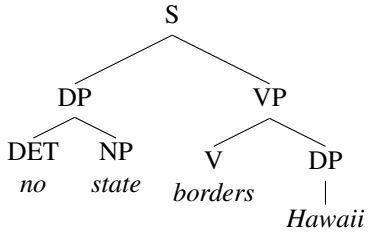
The domain of the verb thus spans a whole sentence, containing its two nominal arguments – one in subject position and one in object position. The corresponding nodes, DP_1 and DP_2, are slots for which any DP-tree can be substituted. For example, substituting the two trees in 2 for subject and object DP, respectively, yields the tree in 3.

2. (a)



(b)

3.



As semantic representations we take DUDEs (Cimiano, 2009), representations similar to structures from *Underspecified Discourse Representation Theory* (UDRT (Reyle, 1993)), extended with some additional information that allows for flexible meaning composition in parallel to the construction of LTAG trees. The DUDE for the verb *to border*, for example, would be the following (in a slightly simplified version):

`geo:borders` (x, y)
$(\mathrm{DP}_1, x), (\mathrm{DP}_2, y)$

[1] `cs.utexas.edu/users/ml/nldata/geoquery.html`

[2] `http://www.sc.cit-ec.uni-bielefeld.de/qald-1`

[3] `http://www.sc.cit-ec.uni-bielefeld.de/pythia`

It provides the predicate `geo:borders` corresponding to the intended concept in the ontology. This correspondence is ensured by using the vocabulary of the ontology, i.e. by using the URI[4] of the concept instead of a more generic predicate. The prefix `geo` specifies the namespace, in this case the one of the GeoBase ontology. Furthermore, the semantic representation contains information about which substitution nodes in the syntactic structure provide the semantic arguments x and y. That is, the semantic referent provided by the meaning of the tree substituted for DP_1 corresponds to the first argument x of the semantic predicate, while the semantic referent provided by the meaning of the tree substituted for DP_2 corresponds to the second argument y. The uppermost row of the box contains the referent that is introduced by the expression. For example, the DUDE for *Hawaii* (paired with the tree in 2b) would be the following:

h
`geo:name` $(h,$ `'hawaii'` $)$

It introduces a referent h which is related to the literal `'hawaii'` by means of the relation `geo:name`. As it does not have any arguments, the third row is empty. The bottom-most row, empty in both DUDEs, is for selectional restrictions of predicates; we will see those in Section 4.

Parallel to substituting the DP-tree in 2b for the DP_1-slot in 1, the DUDE for *Hawaii* is combined with the DUDE for *borders*, amounting to the saturation of the argument (DP_2, y) by unifying the variables h and y, yielding the following DUDE:

h
`geo:borders` (x, h) `geo:name` $(h,$ `'hawaii'` $)$
(DP_1, x)

Substituting the subject argument *no state* involves quantifier representations which we will gloss over as they do not play a role in this paper. At this point it suffices to say that we implement the treatment of quantifier scope in UDRT without modifications.

Once a meaning representation for a question is built, it is translated into a SPARQL query, which can then be evaluated with respect to a given dataset.

Not a lot hinges on the exact choice of the formalisms; we could as well have chosen any other syntactic and semantic formalism that allows the incorporation of underspecification mechanisms. The same holds for the use of SPARQL as formal query language. The reason for choosing SPARQL is that it is the standard query language for the Semantic Web[5]; we therefore feel safe in relying on the reader's familiarity with SPARQL and use SPARQL queries without further explanation.

3 Types of ambiguities

As described in the introduction above, a central task in ontology-based interpretation is the mapping of a natural language expression to an ontology concept. And this mapping gives rise to several different cases of ambiguities.

First, ambiguities can arise due to homonymy of a natural language expression, i.e. an expression that has several lexical meanings, where each of these meanings can be mapped to one ontology concept unambiguously. The ambiguity is inherent to the expression and is independent of any domain or ontology. This is what in linguistic contexts is called a lexical ambiguity. A classical example is the noun *bank*, which can mean a financial institution, a kind of seating, the edge of a river, and a range of other disjoint, non-overlapping alternatives. An example in the geographical domain is *New York*. It can mean either New York city, in this case it would be mapped to the ontological entity `geo:new_york_city`, or New York state, in this case it would be mapped to the entity `geo:new_york`. Ambiguous names are actually the only case of such ambiguities that occur in the GeoBase dataset.

Another kind of ambiguities is due to mismatches between a user's concept of the meaning of an expression and the modelling of this meaning in the ontology. For example, if the ontology modelling is more fine-grained than the meaning

[4] URI stands for *Uniform Resource Identifier*. URIs uniquely identify resources on the Web. For an overview, see, e.g., `http://www.w3.org/Addressing/`.

[5] For the W3C reference, see `http://www.w3.org/TR/rdf-sparql-query/`.

of a natural language expression, then an expression with one meaning can be mapped to several ontology concepts. These concepts could differ extensionally as well as intensionally. An example is the above mentioned expression *starring*, that an ontology engineer could want to comprise only leading roles or also include supporting roles. If he decides to model this distinction and introduces two properties, then the ontological model is more fine-grained than the meaning of the natural language expression, which could be seen as corresponding to the union of both ontology properties. Another example is the expression *inhabitants* in question 4, which can be mapped either to `<http://dbpedia.org/property/population>` or to `<http://dbpedia.org/ontology/populationUrban>`. For most cities, both alternatives give a result, but they differ slightly, as one captures only the core urban area while the other also includes the outskirts. For some city, even only one of them might be specified in the dataset.

4. *Which cities have more than two million inhabitants?*

Such ambiguities occur in larger datasets like DBpedia with a wide range of common nouns and transitive verbs. In the QALD-1 training questions for DBpedia, for example, at least 16 % of the questions contain expressions that do not have a unique ontological correspondent.

Another source for ambiguities is the large number of vague and context-dependent expressions in natural language. While it is not possible to pinpoint such expressions to a fully specified lexical meaning, a question answering system needs to map them to one (or more) specific concept(s) in the ontology. Often there are several mapping possibilities, sometimes depending on the linguistic context of the expression.

An example for context-dependent expressions in the geographical domain is the adjective *big*: it refers to size (of a city or a state) either with respect to population or with respect to area. For the question 5a, for example, two queries could be intended – one refering to population and one refering to area. They are given in 5b and 5c.

5. (a) *What is the biggest city?*

   (b)
```
SELECT ?s WHERE {
  ?s a geo:city .
  ?s geo:population ?p . }
  ORDER BY DESC ?p LIMIT 1
```

   (c)
```
SELECT ?s WHERE {
  ?s a geo:city .
  ?s geo:area ?a . }
  ORDER BY DESC ?a LIMIT 1
```

Without further clarification – either by means of a clarification dialog with the user (e.g. employed by FREyA (Damljanovic et al., 2010)) or an explicit disambiguation as in *What is the biggest city by area?* – both interpretations are possible and adequate. That is, the adjective *big* introduces two mapping alternatives that both lead to a consistent interpretation.

A slightly different example are vague expressions. Consider the questions 6a and 7a. The verb *has* refers either to the object property `flowsThrough`, when relating states and rivers, or to the object property `inState`, when relating states and cities. The corresponding queries are given in 6b and 7b.

6. (a) *Which state has the most rivers?*

   (b)
```
SELECT COUNT(?s) AS ?n WHERE {
  ?s a geo:state .
  ?r a geo:river .
  ?r geo:flowsThrough ?s. }
  ORDER BY DESC ?n LIMIT 1
```

7. (a) *Which state has the most cities?*

   (b)
```
SELECT COUNT(?s) AS ?n WHERE {
  ?s a geo:state .
  ?c a geo:city .
  ?c geo:inState ?s. }
  ORDER BY DESC ?n LIMIT 1
```

In contrast to the example of *big* above, these two interpretations, `flowsThrough` and `inState`, are exclusive alternatives: only one of them is admissible, depending on the linguistic context. This is due to the sortal restrictions of those properties: `flowsThrough` only allows rivers as domain, whereas `inState` only allows cities as domain.

This kind of ambiguities are very frequent, as a lot of user questions contain semantically light expressions, e.g. the copula verb *be*, the verb *have*,

and prepositions like *of*, *in* and *with* (cf. (Cimiano & Minock, 2009)) – expressions which are vague and do not specify the exact relation they are denoting. In the 880 user questions that Mooney provides, there are 1278 occurences of the light expressions *is/are*, *has/have*, *with*, *in*, and *of*, in addition to 151 ocurrences of the context-dependent expressions *big*, *small*, and *major*.

4 Capturing and resolving ambiguities

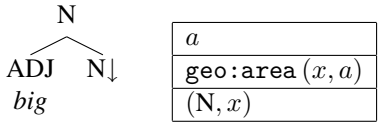
When constructing a semantic representation and a formal query, all possible alternative meanings have to be considered. We will look at two strategies to do so: simply enumerating all interpretations (constructing a different semantic representation and query for every possible interpretation), and underspecification (constructing only one underspecified representation that subsumes all different interpretations).

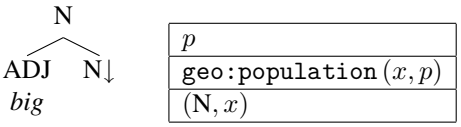
4.1 Enumeration

Consider the example of a lexically ambiguous question in 8a. It contains two ambiguous expressions: *New York* can refer either to the city or the state, and *big* can refer to size either with respect to area or with respect to population. This leads to four possible interpretations of the questions, given in 8b–8e.

8. (a) *How big is New York?*
   (b) `SELECT ?a WHERE { geo:new_york_city geo:area ?a . }`
   (c) `SELECT ?p WHERE { geo:new_york_city geo:population ?p.}`
   (d) `SELECT ?a WHERE { geo:new_york geo:area ?a . }`
   (e) `SELECT ?p WHERE { geo:new_york geo:population ?p . }`

Since the question in 8a can indeed have all four interpretations, all of them should be captured. The enumeration strategy amounts to constructing all four queries. In order to do so, we specify two lexical entries for *New York* and two lexical entries for the adjective *big* – one for each reading. For *big*, these two entries are given in 9 and 10. The syntactic tree is the same for both, while the semantic representations differ: one refers to the property `geo:area` and one refers to the property `geo:population`.

9. 

10. 

When parsing the question *How big is New York*, both entries for *big* are found during lexical lookup, and analogously two entries for *New York* are found. The interpretation process will use all of them and therefore construct four queries, 8b–8e.

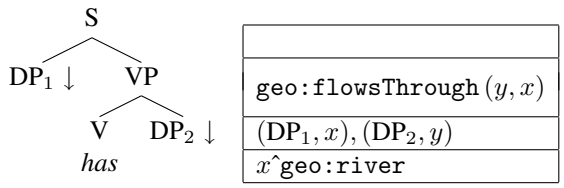
Vague and context-dependent expressions can be treated similarly. The verb *to have*, for example, can map either to the property `flowsThrough`, in the case of rivers, or to the property `inState`, in the case of cities. Now we could simply specify two lexical entries *to have* – one using the meaning `flowsThrough` and one using the meaning `inState`. However, contrary to lexical ambiguities, these are not real alternatives in the sense that both lead to consistent readings. The former is only possible if the relevant argument is a river, the latter is only relevant if the relevant argument is a city. So in order not to derive inconsistent interpretations, we need to capture the sortal restrictions attached to such exclusive alternatives. This will be discussed in the next section.

4.2 Adding sortal restrictions

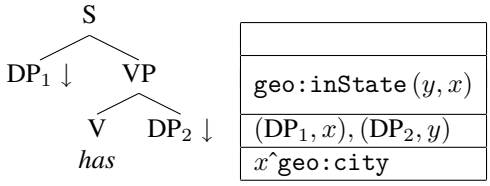
A straightforward way to capture ambiguities consists in enumerating all possible interpretations and thus in constructing all corresponding formal queries. We did this by specifying a separate lexical entry for every interpretation. The only difficulty that arises is that we have to capture the sortal restrictions that come with some natural language expressions. In order to do so, we add sortal restrictions to our semantic representation format.

Sortal restrictions will be of the general form *variable*`^class`. For example, the sortal restriction that instances of the variable x must belong to the class `river` in our domain would be represented as x`^geo:river`. Such sortal restrictions are added as

a list to our DUDEs. For example, for the verb *has* we specify two lexical entries. One maps *has* to the property `flowThrough`, specifying the sortal restriction that the first argument of this property must belong to the class `river`. This entry looks as follows:



The other lexical entry for *has* consists of the same syntactic tree and a semantic representation that maps *has* to the property `inState` and contains the restriction that the first argument of this property must belong to the class `city`. It looks as follows:



When a question containg the verb *has*, like 11a, is parsed, both interpretations for *has* are found during lexical lookup and two semantic representations are constructed, both containing a sortal restriction. When translating the semantic representations into a formal query, the sortal restriction is simply added as a condition. For 11a, the two corresponding queries are given in 11b (mapping *has* to `flowsThrough`) and 11c (mapping *has* `inState`). The contribution of the sortal restriction is boxed.

11. (a) *Which state has the most rivers?*

    (b)
```
SELECT COUNT(?r) as ?c WHERE {
?s a geo:state .
?r a geo:river .
?r geo:flowsThrough ?s .
?r a geo:river . }
ORDER BY ?c DESC LIMIT 1
```

    (c)
```
SELECT COUNT(?r) as ?c WHERE {
?s a geo:state .
?r a geo:river .
?r geo:inState ?s .
?r a geo:city . }
ORDER BY ?c DESC LIMIT 1
```

In the first case, 11b, the sortal restriction adds a redundant condition and will have no effect. We can say that the sortal restriction is satisfied. In the second case, in 11c, however, the sortal restriction adds a condition that is inconsistent with the other conditions, assuming that the classes `river` and `city` are properly specified as disjoint. The query will therefore not yield any results, as no instantiiation of `r` can be found that belongs to both classes. That is, in the context of rivers only the interpretation using `flowsThrough` leads to results.

Actually, the sortal restriction in 11c is already implicitly specified in the ontological relation `inState`: there is no river that is related to a state with this property. However, this is not necessarily the case and there are indeed queries where the sortal restriction has to be included explicitly. One example is the interpretation of the adjective *major* in noun phrases like *major city* and *major state*. Although with respect to the geographical domain *major* always expresses the property of having a population greater than a certain threshold, this threshold differs for cities and states: *major* with respect to cities is interpreted as having a population greater than, say, 150 000, while *major* with respect to states is interpreted as having a population greater than, say, 10 000 000. Treating *major* as ambiguous between those two readings without specifying a sortal restriction would lead to two readings for the noun phrase *major city*, sketched in 12. Both would yield non-empty results and there is no way to tell which one is the correct one.

12. (a)
```
SELECT ?c WHERE {
?c a geo:city .
?c geo:population ?p .
FILTER ( ?p > 150000 ) }
```

    (b)
```
SELECT ?c WHERE {
?c a geo:city .
?c geo:population ?p .
FILTER ( ?p > 10000000 ) }
```

Specifying sortal restrictions, on the other hand, would add the boxed material in 13, thereby causing the wrong reading in 13b to return no results.

13. (a)
```
SELECT ?c WHERE {
?c a geo:city .
?c geo:population ?p .
```

```
FILTER ( ?p > 150000 ) .
?c a geo:city . }
```

(b)
```
SELECT ?c WHERE {
?c a geo:city .
?c geo:population ?p .
FILTER ( ?p > 10000000 ) .
?c a geo:state . }
```

The enumeration strategy thus relies on a conflict that results in queries which return no result. Unwanted interpretations are thereby filtered out automatically. But two problems arise here. The first one is that we have no way to distinguish between queries that return no result due to an inconsistency introduced by a sortal restriction, and queries that return no result, because there is none, as in the case of *Which states border Hawaii?*. The second problem concerns the number of readings that are constructed. In view of the large number of ambiguities, even in the restricted geographical domain we used, user questions easily lead to 20 or 30 different possible interpretations. In cases in which several natural language terms can be mapped to many different ontological concepts, this number rises. Enumerating all alternative interpretations is therefore not efficient. A more practical alternative is to construct one underspecified representation instead and then infer a specific interpretation in a given context. We will explore this strategy in the next section.

4.3 Underspecification

In the following, we will explore a strategy for representing and resolving ambiguities that uses underspecification and ontological reasoning in order to keep the number of constructed interpretations to a minimum. For a general overview of underspecification formalisms and their applicability to linguistic phenomena see (Bunt, 2007).

In order not to construct a different query for every interpretation, we do not any longer specify separate lexical entries for each mapping but rather combine them by using an underspecified semantic representation. In the case of *has*, for example, we do not specify two lexical entries – one with a semantic representation using `flowsThrough` and one entry with a representation using `inState` – but instead specify only one lexical entry with a representation using a metavariable, and additionally specify which properties this metavariable stands for under which conditions.

So first we extend DUDEs such that they now can contain metavariables, and instead of a list of sortal restrictions contain a list of *metavariable specifications*, i.e. possible instantiations of a metavariable given that certain sortal restrictions are satisfied, where sortal restrictions can concern any of the property's arguments. Metavariable specifications take the following general form:

$$\begin{aligned}\mathcal{P} \rightarrow\ & \mathtt{p}_1\ (x = \mathtt{class}_1, \ldots, y = \mathtt{class}_2)\\ |\ & \mathtt{p}_2\ (x = \mathtt{class}_3, \ldots, y = \mathtt{class}_4)\\ |\ & \ldots\\ |\ & \mathtt{p}_n\ (x = \mathtt{class}_i, \ldots, y = \mathtt{class}_j)\end{aligned}$$

This expresses that some metavariable $\mathcal{P}$ stands for a property $\mathtt{p}_1$ if the types of the arguments $x, \ldots, y$ are equal to or a subset of $\mathtt{class}_1$,...,$\mathtt{class}_2$, and stands for some other property if the types of the arguments correspond to some other classes. For example, as interpretation of *has*, we would chose a metavariable $\mathcal{P}$ with a specification stating that $\mathcal{P}$ stands for the property `flowsThrough` if the first argument belongs to class `river`, and stands for the property `inState` if the first argument belongs to the class `city`. Thus, the lexical entry for *has* would contain the following underspecified semantic representation.

14. *Lexical meaning of 'has':*

$\mathcal{P}\,(y, x)$
$(\mathrm{DP}_1, x), (\mathrm{DP}_2, y)$
$\mathcal{P} \rightarrow \mathtt{geo:flowsThrough}\ (y = \mathtt{geo:river})$ $\quad\ |\ \mathtt{geo:inState}\ (y = \mathtt{geo:city})$

Now this underspecified semantic representation has to be specified in order to lead to a SPARQL query that can be evaluated w.r.t. the knowledge base. That means, in the course of interpretation we need to determine which class an instantiation of y belongs to and accordingly substitute $\mathcal{P}$ by the property `flowsThrough` or `inState`. In the following section, we sketch a way of exploiting the ontology to this end.

4.4 Reducing alternatives with ontological reasoning

In order to filter out interpretations that are inconsistent as early as possible and thereby reduce the number of interpretations during the course of a derivation, we check whether the type information of a variable that is unified is consistent with the sortal restrictions connected to the metavariables. This check is performed at every relevant step in a derivation, so that inconsistent readings are not allowed to percolate and multiply. Let us demonstrate this strategy by means of the example *Which state has the biggest city?*.

In order to build the noun phrase *the biggest city*, the meaning representation of the superlative *biggest*, given in 15, is combined with that of the noun *city*, which simply contributes the predication $\texttt{geo:city}\,(y)$, by means of unification.

15.

z
$\mathcal{Q}\,(y, z)$
(N, y)
$\mathcal{Q} \rightarrow \texttt{geo:area}(y = \texttt{geo:city} \sqcup \texttt{geo:state})$ $\mid \texttt{geo:population}(y = \texttt{geo:city} \sqcup \texttt{geo:state})$

The exact details of combining meaning representations do not matter here. What we want to focus on is the metavariable $\mathcal{Q}$ that *biggest* introduces. When combining 15 with the meaning of *city*, we can check whether the type information connected to the unified referent y is compatible with the domain restrictions of $\mathcal{Q}'s$ interpretations. One way to do this is by integrating an OWL reasoner and checking the satisfiability of

$$\texttt{geo:city} \sqcap (\texttt{geo:city} \sqcup \texttt{geo:state})$$

(for both interpretations of $\mathcal{Q}$, as the restrictions on y are the same). Since this is indeed satisfiable, both interpretations are possible, thus cannot be discarded, and the resulting meaning representation of *the biggest city* is the following:

$y\ z$
$\texttt{geo:city}(y)$ $\mathcal{Q}\,(y, z)$ $max(z)$
$\mathcal{Q} \rightarrow \texttt{geo:area}(y = \texttt{geo:city} \sqcup \texttt{geo:state})$ $\mid \texttt{geo:population}(y = \texttt{geo:city} \sqcup \texttt{geo:state})$

This is desired, as the ambiguity of *biggest* is a lexical ambiguity that could only be resolved by the user specifying which reading s/he intended.

In a next step, the above representation is combined with the semantic representation of the verb *has*, given in 14. Now the type information of the unified variable y has to be checked for compatibility with instantiations of an additional metavariable, $\mathcal{P}$. The OWL reasoner would therefore have to check the satisfiability of the following two expressions:

16. (a) $\texttt{geo:city} \sqcap \texttt{geo:river}$
    (b) $\texttt{geo:city} \sqcap \texttt{geo:city}$

While 16b succeeds trivially, 16a fails, assuming that the two classes `geo:river` and `geo:city` are specified as disjoint in the ontology. Therefore the instantiation of $\mathcal{P}$ as `geo:flowsThrough` is not consistent and can be discarded, leading to the following combined meaning representation, where $\mathcal{P}$ is replaced by its only remaining instantiation `geo:inState`:

$y\ z$
$\texttt{geo:city}(y)$ $\texttt{geo:inState}\,(y, x)$ $\mathcal{Q}\,(y, z)$
(DP_1, x)
$\mathcal{Q} \rightarrow \texttt{geo:area}(y = \texttt{geo:city} \sqcup \texttt{geo:state})$ $\mid \texttt{geo:population}(y = \texttt{geo:city} \sqcup \texttt{geo:state})$

Finally, this meaning representation is combined with the meaning representation of *which state*, which simply contributes the predication $\texttt{geo:state}\,(x)$. As the unified variable x does not occur in any metavariable specification, nothing further needs to be checked. The final meaning representation thus leaves one metavariable with two possible instantiations and will lead to the following two corresponding SPARQL queries:

17. (a)
```
SELECT ?x WHERE {
?x a geo:city .
?y a geo:state.
?x geo:population ?z .
?x geo:inState ?y . }
ORDER BY DESC(?z) LIMIT 1
```
(b)
```
SELECT ?x WHERE {
?x a geo:city .
?y a geo:state.
?x geo:area ?z .
?x geo:inState ?y . }
ORDER BY DESC(?z) LIMIT 1
```

Note that if the ambiguity of the metavariable $\mathcal{P}$ were not resolved, we would have ended up with four SPARQL queries, where two of them use the relation `geo:flowsThrough` and therefore yield empty results. So in this case, we reduced the number of constructed queries by half by discarding inconsistent readings. We therefore solved the problems mentioned at the end of 4.2: The number of constructed queries is reduced, and since we discard inconsistent readings, null answers can only be due to the lack of data in the knowledge base but not cannot anymore be due to inconsistencies in the generated queries.

5 Implementation and results

In order to see that the possibility of reducing the number of interpretations during a derivation does not only exist in a small number of cases, we applied Pythia to Mooney's 880 user questions, implementing the underspecification strategy in 4.3 and the reduction strategy in 4.4. Since Pythia does not yet integrate a reasoner, it approximates satisfiability checks by means of SPARQL queries. Whenever meaning representations are combined, it aggregates type information for the unified variable, together with selectional information connected to the occuring metavariables, and uses both to construct a SPARQL query. This query is then evaluated against the underlying knowledge base. If the query returns results, the interpetations are taken to be compatible, if it does not return results, the interpretations are taken to be incompatible and the according instantiation possibility of the metavariable is discarded. Note that those SPARQL queries are only an approximation for the OWL expressions used in 4.4. Furthermore, the results they return are only an approximation of satisfiability, as the reason for not returning results does not necessarily need to be unsatisfiability of the construction but could also be due the absence of data in the knowledge base. In order to overcome these shortcomings, we plan to integrate a full-fledged OWL reasoner in the future.

Out of the 880 user questions, 624 can be parsed by Pythia (for an evaluation on this dataset and reasons for failing with the remaining 256 questions, see (Unger & Cimiano, 2011)). Implementing the enumeration strategy, i.e. not using disambiguation mechanisms, there was a total of 3180 constructed queries. With a mechanism for removing scope ambiguities by means of simulating a linear scope preference, a total of 2936 queries was built. Additionally using the underspecification and resolution strategies described in the previous section, by exploiting the ontology with respect to which natural language expressions are interpreted in order to discard inconsistent interpretations as early as possible in the course of a derivation, the number of total queries was further reduced to 2100. This amounts to a reduction of the overall number of queries by 44 %. The average and maximum number of queries per question are summarized in the following table.

	Avg. # queries	Max. # queries
Enumeration	5.1	96
Linear scope	4.7 (-8%)	46 (-52%)
Reasoning	3.4 (-44%)	24 (-75%)

6 Conclusion

We investigated ambiguities arising from mismatches between a natural language expressions' lexical meaning and its conceptual modelling in an ontology. Employing ontological reasoning for disambiguation allowed us to significantly reduce the number of constructed interpretations: the average number of constructed queries per question can be reduced by 44 %, the maximum number of queries per question can be reduced even by 75 %.

References

Bunt, H.: Semantic Underspecification: Which Technique For What Purpose? In: Computing Meaning, vol. 83, pp. 55–85. Springer Netherlands (2007)

Cimiano, P.: Flexible semantic composition with DUDES. In: Proceedings of the 8th International Conference on Computational Semantics (IWCS). Tilburg (2009)

Unger, C., Hieber, F., Cimiano, P.: Generating LTAG grammars from a lexicon-ontology interface. In: S. Bangalore, R. Frank, and M. Romero (eds.): 10th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+10), Yale University (2010)

Unger, C., Cimiano, P.: Pythia: Compositional meaning construction for ontology-based question answering on the Semantic Web. In: Proceedings of the 16th International Conference on Applications of Natural Language to Information Systems (NLDB) (2011)

Schabes, Y.: Mathematical and Computational Aspects of Lexicalized Grammars. Ph. D. thesis, University of Pennsylvania (1990)

Reyle, U.: Dealing with ambiguities by underspecification: Construction, representation and deduction. Journal of Semantics 10, 123–179 (1993)

Kamp, H., Reyle, U.: From Discourse to Logic. Kluwer, Dordrecht (1993)

Cimiano, P., Minock, M.: Natural Language Interfaces: What's the Problem? – A Data-driven Quantitative Analysis. In: Proceedings of the International Conference on Applications of Natural Language to Information Systems (NLDB), pp. 192–206 (2009)

Damljanovic, D., Agatonovic, M., Cunningham, H.: Natural Language Interfaces to Ontologies: Combining Syntactic Analysis and Ontology-based Lookup through the User Interaction. In: Proceedings of the 7th Extended Semantic Web Conference, Springer Verlag (2010)

Zettlemoyer, L., Collins, M.: Learning Context-dependent Mappings from Sentences to Logical Form. In: Proceedings of the Joint Conference of the Association for Computational Linguistics and International Joint Conference on Natural Language Processing (ACL-IJCNLP), pp. 976–984 (2009)

Burger, J., Cardie, C., Chaudhri, V., Gaizauskas, R., Israel, D., Jacquemin, C., Lin, C.-Y., Maiorano, S., Miller, G., Moldovan, D., Ogden, B., Prager, J., Riloff, E., Singhal, A., Shrihari, R., Strzalkowski, T., Voorhees, E., Weischedel, R.: Issues, tasks, and program structures to roadmap research in question & answering (Q & A). `http://www-nlpir.nist.gov/projects/duc/papers/qa.Roadmap-paper_v2.doc` (2001)

Kate, R., Mooney, R.: Learning Language Semantics from Ambiguous Supervision. In: Proceedings of the 22nd Conference on Artificial Intelligence (AAAI-07), pp. 895–900 (2007)