Generating LTAG grammars from a lexicon-ontology interface

Christina Unger **Felix Hieber** **Philipp Cimiano**
Semantic Computing Group
Cognitive Interaction Technology – Center of Excellence (CITEC)
University of Bielefeld, Germany
{cunger,fhieber,cimiano}@cit-ec.uni-bielefeld.de

Abstract

This paper shows how domain-specific grammars can be automatically generated from a declarative model of the lexicon-ontology interface and how those grammars can be used for question answering. We show a specific implementation of the approach using Lexicalized Tree Adjoining Grammars. The main characteristic of the generated elementary trees is that they constitute domains of locality that span lexicalizations of ontological concepts rather than being based on requirements of single lexical heads.

1 Introduction

Many approaches to the interpretation of natural language represent meanings as generic logical forms. However, some domain-specific applications require semantic representations that are aligned to a specific ontology and thus cannot be provided by a generic, ontology-independent semantic construction. To illustrate this, consider the example of a geographical ontology that contains a data property[1] *population* relating states to their number of inhabitants. When constructing a natural language interface to this ontology, it has to be taken into account that *population* can be expressed in different ways – directly as in 1, or with related vocabulary as in 2 and 3.

1. The population of Hawaii is 1 300 000.
2. Hawaii has 1 300 000 inhabitants.
3. 1 300 000 people live in Hawaii.

For these sentences, Boxer (Bos, 2008) generates generic Discourse Representation Structures (DRSs) that can roughly be represented as in 4 (for the sentence in 1) and 5 (for the sentences in 2 and 3).

4.

$x_0\ x_1$
hawaii(x_0)
population(x_1)
of(x_1, x_0)
$\mid x_1 \mid \geq 74\,000\,000$

5.

$x_0\ x_1\ x_2$
hawaii(x_0)
inhabitant(x_1)
have(x_2)
agent(x_2, x_0)
patient(x_2, x_1)
$\mid x_1 \mid \geq 74\,000\,000$

$x_0\ x_1\ x_2$
hawaii(x_0)
people(x_1)
live(x_2)
agent(x_2, x_1)
in(x_2, x_0)
$\mid x_1 \mid \geq 74\,000\,000$

Note that while the predicate *population* can be taken to directly correspond to the concept *population* in the ontology, the predicates *live* and *inhabitant*, for example, would yield an empty extension if evaluated with respect to the ontology. We would need additional meaning postulates that relate these predicates to *population*, or, alternatively, some postprocessing that transforms generic forms like in 5 into a more specific form equivalent to 4.

An arguably more elegant and useful solution is to let the semantic vocabulary follow the vocabulary of the ontology in the first place, i.e. to right

[1] Ontologies distinguish two kinds of relations: *object properties*, that link individuals to individuals, and *data properties*, that link individuals to data values (e.g. strings or integers).

away construct the specifc logical form 4 also for the sentences in 2 and 3. This, however, raises another problem. In order to arrive at the meaning in 4, the semantic contributions of the lexical items have to be adjusted. In the case of 3, for example, the predicate *population* can be assumed to be the semantic contribution of the noun phrase inhabitants, while the verb have is semantically empty. In 2, on the other hand, the predicate *population* might be contributed by the verb live, while the noun people is semantically empty. Although this does not seem problematic in this case, there might not exist a general way to assign meanings to lexical items consistently. Imagine, for example, the ontology also contains a concept *people*, which plays a role in a different context. Then the noun people needs a non-empty semantic interpretation – possibly one which clashes with its use in 3.

The challenging goal thus is to construct an interpretation for sentences that uses the vocabulary of the ontology and therefore might deviate from the surface structure of the sentence. We propose to meet this challenge by determining basic semantic units with respect to an ontology. Assuming a tight connection between syntactic and semantic units, then also basic syntactic units turn out to depend on a particular ontology, and hence the whole syntax-semantics interface becomes ontology-specific.

We show an implementation of this approach using Lexicalized Tree Adjoining Grammars. They are particularly well-suited for the task of compositional ontology-specific interpretation due to their extended domain of locality and their tight connection between syntax and semantics. The general procedure we follow consists of two steps. First, the ontology is connected to a lexicon model that specifies how concepts in the ontology can be realized (e.g. the concept *borders* can be realized as the transitive verb *to border* or as a noun with a prepositional phrase, *border of*). And second, this lexicon model is used to automatically construct grammar entries that are aligned to the underlying ontology. Both steps will be described in the next section.

2 Generating ontology-specific grammars

We assume a grammar to be composed of two parts: an ontology-specific part and an ontology-independent part. The ontology-specific part contains lexical entries that refer to individuals, concepts and properties in the underlying ontology. It is generated automatically from an ontology-lexicon interface model, as described below. The ontology-independent part comprises domain-independent expressions like determiners and auxiliary verbs. It needs to be specified by hand, but can be reused across domains.

We assume grammar entries to be pairs of a syntactic and a semantic representation. As syntactic representation we take trees from Lexicalized Tree Adjoining Grammar (Schabes 1990), LTAG for short. LTAG is very well-suited for ontology-based grammar generation, mainly because it allows for flexible basic units; we will demonstrate the importance of this below. As semantic representations we take DUDEs (Cimiano, 2009) – representations similar to Underspecified Discourse Representation Structures (UDRSs) (Reyle, 1993) augmented with information that facilitate a flexible semantic composition. This semantic flexibility nicely matches the syntactic flexibility provided by LTAG.

The first step in generating the domain-specific part of a grammar is to enrich the underlying ontology with linguistic information. The framework we use for this is LexInfo (Buitelaar et al., 2009). LexInfo offers a general frame for creating a declarative specification of the lexicon-ontology interface by connecting concepts of the ontology to information about their linguistic realization, in particular word forms, morphological information, subcategoriziation frames and mappings between syntactic and semantic arguments. For example, the LexInfo entry for the verb *to border* looks as depicted in Figure 1. It comprises the verb's lemma together with other written forms (in principle all inflected forms, generated by a separate morphology module), and its syntactic behaviour. The syntactic behaviour specifies the syntactic arguments on the one hand, here simply called subject and object, and the semantic arguments on the other hand. The semantic arguments correspond to domain and range of a predicate that refers to the concept *borders* in the ontology (in our case a relation between two states). The predicative representation also specifies mappings between the syntactic and semantic arguments, more specifically between the subject of the verb and the domain of

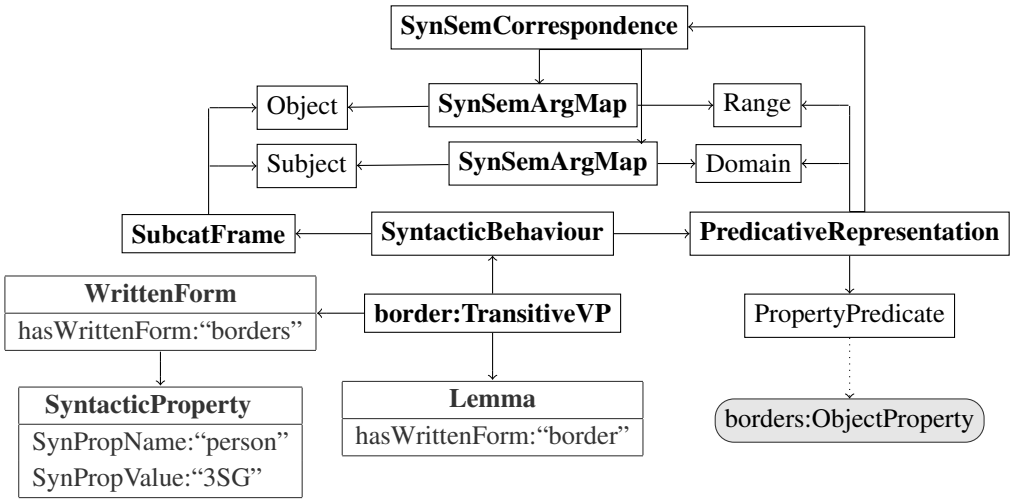


Figure 1: LexInfo entry for *to border*

the denoted relation, as well as the object of the verb and the range of the relation.

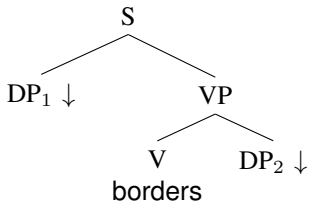
Analogously, LexInfo provides general structures for specifying the syntactic and semantic properties of other categories such as intransitive verbs, noun phrases, nouns with prepositional arguments, and so on. An example of a noun with a prepositional argument is *border of*; the according LexInfo entry is given in Figure 2. Its structure is very similar to the transitive verb entry, the main difference being the additional specification of the preposition required by the noun.

Associating concepts of the ontology with such LexInfo structures is a one-to-many mapping, since most concepts can have different lexicalizations. For the relation *borders*, for example, we would specify the TransitiveVP entry in Figure 1 as well as the NounPP entry in Figure 2. Note that both denote a predicate that is related to the same relation *borders* in the ontology.

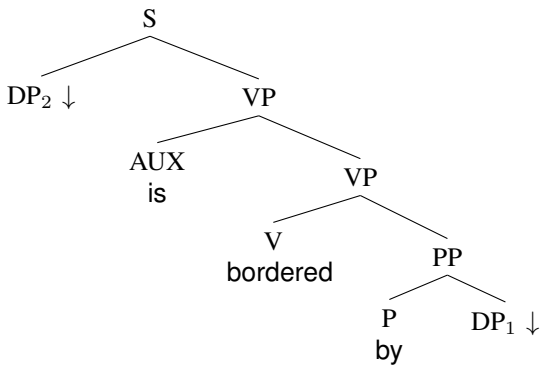
The process of specifying all possible lexicalization alternatives is, up to now, done by hand. However it can in principle be automatized if the ontology labels are chosen in a way such that they can be processed by a part-of-speech tagger and related to possible realizations, e.g. with the help of a corpus.

Once a LexInfo model of the ontology is specified, the next step is to generate grammar entries from it. To this end, the lexical entries specified in the LexInfo model are input to a general mechanism that automatically generates corresponding pairs of syntactic and semantic representations. For the transitive verb entry of *to border* in Figure 1, for example, a number of elementary LTAG trees are generated – two of them are given in 6 and 7. Both trees are paired with the DUDE in 8.

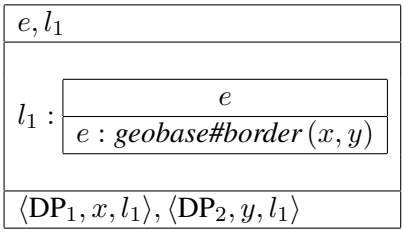
6.



7.



8.



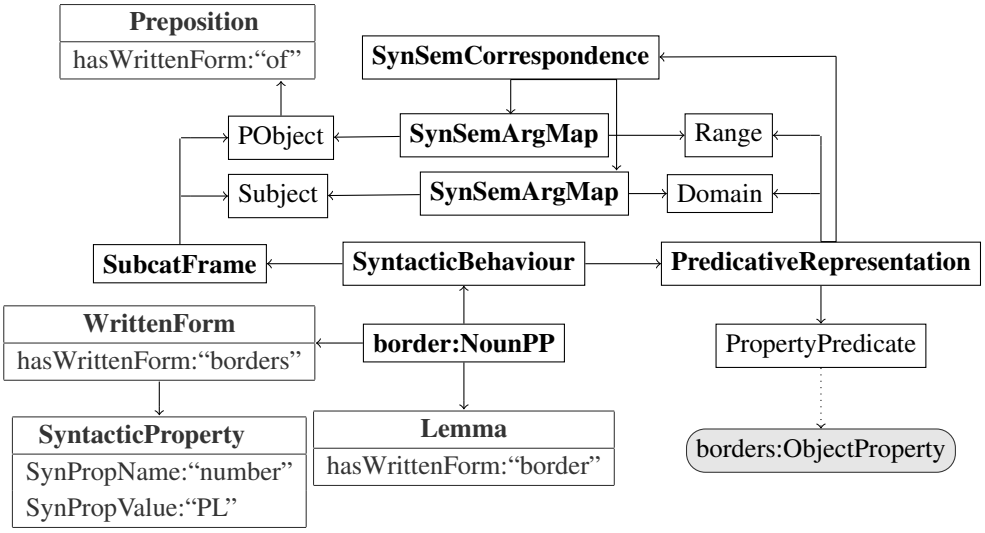
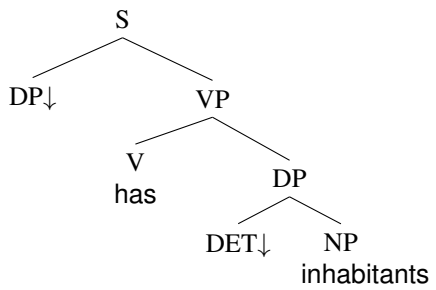


Figure 2: LexInfo entry for *border of*

This semantic representation contains a DRS labeled l_1, which provides the predicate *geobase#border* 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 of the concept instead of a more generic predicate. The prefix *geobase#* is short for `http://www.geobase.org/` and specifies the namespace of the *Geobase* ontology, which we use (more on this in the next section). 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 domain of the semantic predicate, while the semantic referent provided by the meaning of the tree substituted for DP_2 corresponds to the predicate's range. Note that the order of the substitution nodes is reversed in the passive tree structure in 7.

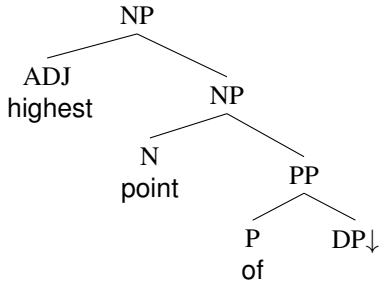
Additional grammar entries generated from the LexInfo model in Figure 1 are adjunction trees for relative clauses and the gerund *bordering*, together with a semantic representation similar to 8. In all these cases the syntactic structure encoded in the elementary trees captures the lexical material that is needed for verbalizing the concept *borders*. This is the reason why the generated syntactic structures slightly differ from what one would expect from an LTAG elementary tree. Elementary trees are commonly assumed to constitute extended projections of lexical items, comprising their syntactic and semantic arguments, cf. (Frank, 1992). In our generated grammar, however, elementary trees do not build around lexical items but rather around lexicalizations of concepts from the ontology. That is, not only the semantic primitives but also the syntactic domain of locality are imposed by the underlying ontology. In many cases, this domain does not extend beyond the familiar lexical projections (recall the tree in 6), however it might contain more lexical material. For example, for the *population* example from the beginning we would need a tree like in 9.

9.



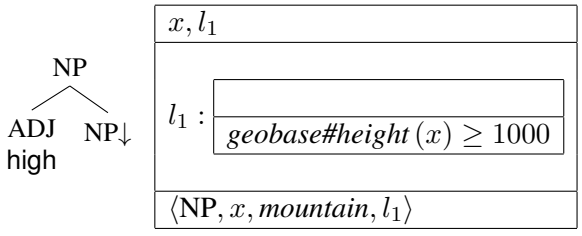
Also, for a concept such as *highest point*, that relates states and locations, we would want to generate a tree like in 10.

10.



As one of the reviewers points out, an alternative approach to dealing with such cases would be to generate derivation trees that are composed of a more conventional set of elementary trees. Although this is doubtlessly a reasonable and elegant idea, we confined ourselves to generating grammar entries solely from the ontology, i.e. without reference to an additional, given set of basic structures.

Yet another example worth considering is the case of adjectives. Figure 3 shows the LexInfo entry for the adjective *high* in its attributive use. Semantically, it refers to a data property *height* in the ontology, that maps mountains to integers. The domain of this property corresponds to the syntactic argument which the adjective requires (namely the noun that it modifies), and it is also linked to the ontology via a selectional restriction that specifies to which class the domain must belong, in this case *mountain*. When building a grammar entry, this selectional restriction is encoded in the DUDE's argument requirements. Additionally, the PredicativePresentation contains information about the polarity of *high* (which provides the $\geq$ in the adjective's semantics) and the value from which on something counts as high. So a generated grammar entry for *high* would contain the following LTAG tree and DUDE:



From the same LexInfo model, grammar entries for the comparative and the superlative case are generated. The written form of the adjective in these cases is determined by the morphology module according to the MorphologicalPattern.[2]

The methods for generating grammar entries from the LexInfo model run automatically. They build on templates that are defined for every LexInfo type (or syntactic category, if you want), i.e. for intransitive and transitive verbs, for nouns with prepositional complements, for adjectives, and so on. They use the subcategorization frame of a lexical entry in LexInfo in order to build an appropriate LTAG tree, and they use the mapping between syntactic and semantic arguments to connect substitution nodes in the tree with argument requirements in the semantic representation.

Although the templates that guide the generation of grammar entries have to be specified by hand for each language one wants to cover, they are completely general in the sense that they are not tied to a particular LexInfo model. Therefore they need to be built only once and can be reused when other LexInfo models are built for other ontologies. Up to now we provide extensive templates for English and basic ones for German. However there is no principled restriction to certain languages since LexInfo is designed in a largely language-independent way.

In the following section we demonstrate our approach with a particular application, a question answering system, and point out the amount of work that is needed for lexicalizing an ontology.

3 Applying the approach to question answering on Geobase

We deployed the approach sketched in the previous section in the context of a question answering system on Raymond Mooney's *Geobase* dataset, which comprises geographical information about the US. Our OWL version of it contains 10 classes (such as *city, state, river, mountain*), 692 individuals instantiating them, 9 object properties (such as *borders, capital, flows through*) and another 9 data properties (such as *population* and *height*).

The LexInfo model constructed for this ontology contains 762 lexical entries. 692 of them correspond to common nouns representing individuals, which are constructed automatically. The remaining 70 entries were built by hand, using the API that LexInfo

[2]For more details on adjectives in LexInfo see (Buitelaar et al., 2009).

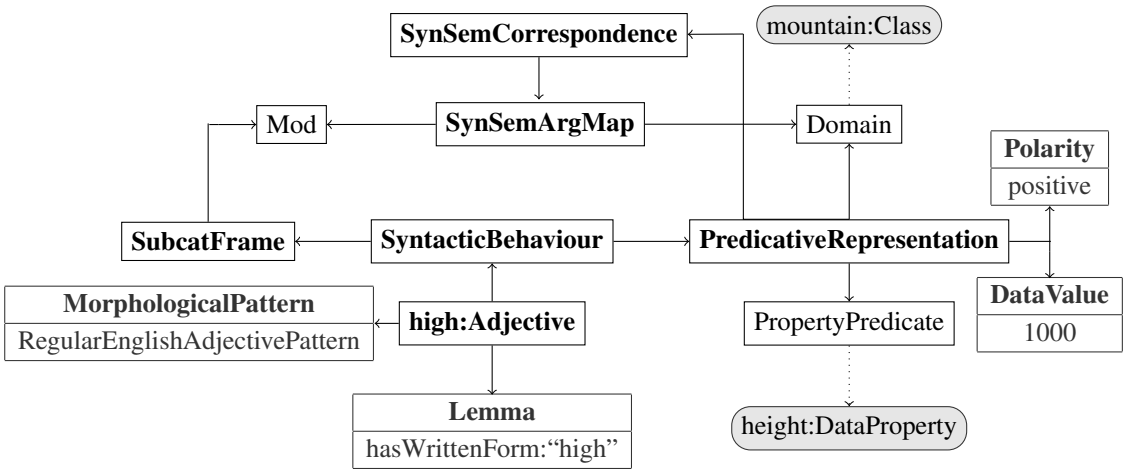
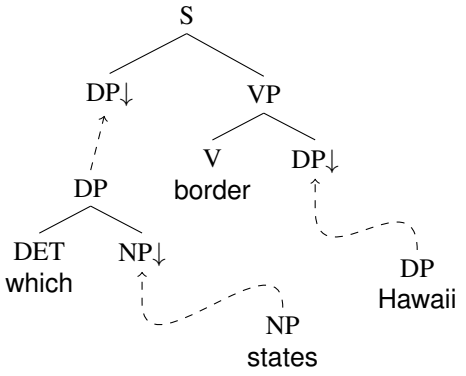


Figure 3: LexInfo entry for *high*

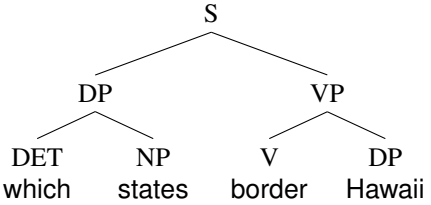
provides. If we generously assume an effort of 5 minutes for building one such entry, the amount of work needed for lexicalizing the ontology amounts to approximately 6 hours. Then from those LexInfo entries, 2785 grammar entries (i.e. pairs of LTAG trees and DUDEs) were automatically generated according to the templates we specified. In addition, we manually specified 149 grammar entries for domain-independent elements such as determiners, wh-words, auxiliary verbs, and so on.

The complete set of grammar entries finally constitutes a domain-specific grammar that can be used for parsing and interpretation. We demonstrate this by feeding it into our question answering system *Pythia*. (For the Geobase dataset, the LexInfo model, the grammar files and a demo check `http://sc.cit-ec.uni-bielefeld/pythia`). Its architecture is depicted in Figure 4. First, the input is handed to a parser that works along the lines of the Earley-type parser devised by Schabes & Joshi (Schabes & Joshi, 1988). It constructs an LTAG derivation tree, considering only the syntactic components of the lexical entries involved. Next, syntactic and semantic composition rules apply in tandem in order to construct a derived tree together with an according DUDE. The syntactic composition rules are TAG's familiar substitution and adjoin operation, and the semantic composition rules are parallel operations on DUDEs: an argument saturating operation (much like function application) that interprets substitution, and a union operation that interprets adjoin. Once all argument slots are filled, the constructed DUDE corresponds to an equivalent UDRS, which is then subject to scope resolution, resulting in a set of disambiguated Discourse Representation Structures. Those can subsequently be translated into a query language.

As an example, consider the input question Which states border Hawaii?. The parser produces the following derivation tree:



Applying all substitutions yields the derived tree:



Parallel to this, a semantic representation is built which resolves to the following DRS:

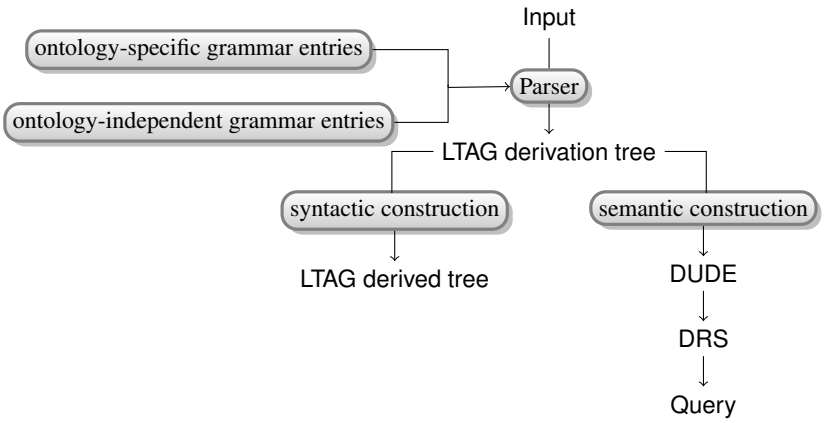


Figure 4: Architecture of *Pythia*

$?x\ y$
geobase#state(x)
$y =$ *geobase#hawaii*
geobase#border(x, y)

Subsequently, this DRS is translated into the following FLogic query (Kifer & Lausen, 1989):

```
FORALL X Y <- X:geobase#state
 AND equal(Y,geobase#hawaii)
 AND X[geobase#border -> Y].
 orderedby X
```

The query is then evaluated with respect to the ontology using the OntoBroker Engine (Decker et al., 1999). Since there are no states bordering Hawaii, the returned result is empty.

4 Related work

Our work bears strong resemblance to semantic grammars (Burton, 1976; Hendrix, 1977), for their motivation was very similar to ours: Since some syntactic constituents contribute little or nothing to meaning, and since meaning components may be distributed across parse trees, grammar rules should not be syntax-driven but tailored to a particular semantic domain. Burton and Hendrix therefore specified phrase-structure grammars that rely on semantic classes instead of syntactic categories. The main disadvantage, however, are its limited possibilities for reuse. A semantic grammar is tailored to one specific domain and cannot easily be ported to another one. Although our approach also yields a grammar which is aligned to a given ontology, it builds on a principled syntactic and semantic theory and therefore bears enough linguistic generality to be easily portable to new domains.

Our approach is also related to the *Ontological Semantics* framework of Nirenburg & Raskin (Nirenburg & Raskin, 2004), which attempts to construct ontological representations as well. However, in their case the ontology is assumed to be fixed and cannot be exchanged. The strength of our approach is that it can be adapted to different domains by the fact that the grammar is directly generated from an abstract specification of the lexicon-ontology interface.

Our approach is also close to recent work by Bobrow and others (Bobrow et al., 2009). The main difference, however, is that they rely on rewriting steps while we directly construct an ontology-specific underspecified semantic representation. Thus, in our case the syntax-semantics interface itself is ontology-specific. This is an advantage, in our view, as the ontology can be used at construction time for disambiguation, for example. But it is still up to future research to show whether rewriting or direct mapping approaches are more suitable to ontology-specific interpretation.

5 Conclusion

A range of applications, such as question answering with respect to a particular domain, require a domain-specific interpretation of natural language expressions. We argued that building generic log-

ical forms is often not suitable for this task. We therefore proposed an approach that relies on generating grammars from an underlying ontology. We took grammar entries to consist of semantic representations with a vocabulary that is aligned to that of the ontology, and syntactic representations that comprise all lexical material needed for verbalizing the ontology concepts.

We demonstrated that Tree Adjoining Grammars fit such an approach very nicely, mainly because of their extended domain of locality, the resulting flexibility regarding basic syntactic units, and due to their tight syntax-semantics interface.

Although the grammars we generate are aligned to one specific ontology, the mechanism that generates them is completely general and works independent of the underlying ontology. This is because it does not generate grammar entries from the ontology itself but rather relies on a mediating lexicon model that contains all relevant linguistic information. Whenever we want to port our question answering system, for example, to a different domain, only the lexicon model would need to be replaced.

We illustrated that the lexicalization of an ontology requires only a reasonable amount of manual engineering. However, it does not easily scale to very large ontologies. It will therefore be subject of future research to automatize this process.

Another area for further work is the amount of LTAG trees that are generated. Up to now, they comprise all possible alternations; for example, for a transitive verb, the basic transitive structure is generated, together with corresponding wh-movement trees, passive trees, relative clause trees, and so on. This leads to redundancies in the grammar generation mechanism and misses important linguistic generalizations. A more principled approach would involve general rules that derive those additional trees from the basic transitive structure.

Acknowledgments

We would like to thank the reviewers for their helpful comments and John McCrae for all LexInfo-related work and discussions.

References

Daniel G. Bobrow and Cleo Condoravdi and Lauri Karttunen and Annie Zaenen. 2009. *Learning by reading: normalizing complex linguistic structures onto a knowledge representation*. AAAI Symposium 2009: Learning by Reading and Learning to Read.

Johan Bos. 2008. *Wide-Coverage Semantic Analysis with Boxer*. Proceedings of Semantics in Text Processing (STEP 2008), pages 277–286. Research in Computational Semantics, College Publications.

Paul Buitelaar and Philipp Cimiano and Peter Haase and Michael Sintek. 2009. *Towards Linguistically Grounded Ontologies*. Proceedings of the 6th Annual European Semantic Web Conference (ESWC).

Richard R. Burton. 1976. *Semantic Grammar: An Engineering Technique for Constructing Natural Language Understanding Systems*. Tech. Report 3453, Bolt, Beranak and Newman Inc., Cambridge, MA.

Philipp Cimiano. 2009. *Flexible Composition with DUDES*. Proceedings of the 8th International Conference on Computational Semantics (IWCS'09).

Stefan Decker and Michael Erdmann and Dieter Fensel and Rudi Studer. 1999. *Ontobroker: Ontology Based Access to Distributed and Semi-Structured Information*. Database Semantics: Semantic Issues in Multimedia Systems, Kluwer. Pages 351–369.

Robert Frank. 1992. *Syntactic Locality and Tree Adjoining Grammar: Grammatical, Acquisition and Processing Perspectives*. Ph.D. thesis, University of Pennsylvania.

Gary G. Hendrix. 1977. *Human engineering for applied natural language processing*. Proceedings of the Fifth International Joint Conference on Artificial Intelligence, pages 183–191.

Michael Kifer and Georg Lausen. 1989. *F-Logic: A Higher-Order language for Reasoning about Objects, Inheritance, and Scheme*. SIGMOD Record 18(2): 134–146. ACM, New York.

Sergei Nirenburg and Victor Raskin. 2004. *Ontological Semantics*. MIT Press.

Uwe Reyle 1993. *Dealing with ambiguities by underspecification: construction, representation and deduction*. Journal of Semantics 10: 123–179.

Yves Schabes and Aravind K. Joshi 1988. *An Earley-type parsing algorithm for Tree Adjoining Grammars*. Proceedings of the 26th annual meeting on Association for Computational Linguistics (ACL), pages 258–269.