Typeful Ontologies with Direct Multilingual Verbalization

Abstract. There is an exciting development in the ontology description languages. However the main focus is on the knowledge representation aspect and not so much on two other aspects that are just as important in practice. First, most languages are based on some kind of untyped logic which allows to assert axioms which are not well-formed. In contrast, even the simplest database systems are equiped with database schema which rules out incorrect records. In a long run this helps a lot to maintain the information consistent. Another aspect which is of interest in many ontology based systems is to have verbalization of facts and axioms in some controlled language. Although this is not something new, it is usually seen as completely separated component. From engineering perespective, it is advantageous to use the same language for both ontology description and controlled language development. In our experiment we also realized that in many natural languages the type information i.e. the ontological classes affect the language generation.

1 Introduction

Developing large scale ontologies is always error-prone process. For instance, a predicate could be applied to arguments of wrong type. Even if its initial usage was correct, the type of some instance could be changed in the later development. After that, all spots where the instance was used have to be updated. Since this is a manual process it is very likely that errors will happen. The programming languages community was dealing with that from the very beginning of the existence of computer science and developed many different type systems. Unfortunately this problem is largely ignored in the ontology development community. For instance both KIF [1], OWL [2] and CycL [3] are untyped. We present an experiment about using a strictly typed language to encode ontologies. The language of choice is GF [4]. As a use case we took SUMO [5], the largest open-source ontology available today.

GF is a grammar formalism which makes distinction between abstract syntax and concrete syntax. The abstract syntax is a logical system based on Martin Löf's type theory [6] and describes the ontological relations in the domain. The concrete syntax relates every concept from the abstract syntax to a phrase in some concrete language. Since the abstract syntax is language independent it is possible to have several concrete syntaxes associated with one abstract syntax.

We developed conversion tool which converted the SUMO axioms from KIF to abstract syntax in GF (section 3). Since GF is also a framework for developing grammars, it was natural to develop natural language interface to SUMO (section 4). Indeed SUMO already has templates for natural language generation. They

were processed and automatically converted to concrete syntax in GF. The result is a controlled language which could be used to formulate new axioms in SUMO or to render existing axioms in natural language. We used only the English patterns because they have the largest coverage and the best quality. However as an experiment, a fraction of the controlled language was ported to French and Romanian. This revealed that the type system in the ontology is also important for the natural language generation. The complex morphological patterns in these languages would make it very difficult to produce fluent language using the original simple patterns in SUMO. Fortunately this is not a problem in GF.

2 The type system in SUMO-GF

We define the category of all classes to be *Class*:

$$\textbf{cat}\ \mathit{Class};$$

This is both syntactic category and semantic type. All classes in the ontology will then be values of type *Class*. All direct instances of class C will get type *Ind* C. When type checking we also need a way to say that some instance is of some sub-class of C, in this case we assign type *El* C:

$$\textbf{cat}\ \mathit{El}\ \mathit{Class};$$
$$\mathit{Ind}\ \mathit{Class};$$

The relation between this two types is encoded by the coercion:

$$\textbf{fun}\ \mathit{el} : (c_1, c_2 : \mathit{Class}) \to \mathit{Inherits}\ c_1\ c_2 \to \mathit{Ind}\ c_1 \to \mathit{El}\ c_2;$$

i.e. something is of class c_2 if it is a direct instance of c_1, and c_1 is a subclass of c_2. The reflexive - transitive closure of the subclass relation is modeled using the dependent type *Inherits*:

$$\textbf{cat}\ \mathit{Inherits}\ (c_1, c_2 : \mathit{Class});$$

Finally there is one more type *Formula* which is the return type for all predicates.

3 Mapping SUMO to SUMO-GF

A total of 17 modules from the SUMO ontology have been processed. Their area of coverage ranges from the most general facts (Merge - the core SUMO ontology) to the most specific domains like weapons of mass distruction. The over 10 000 objects and almost 7 000 relations have been translated automatically to GF. For example:

```
(instance FullyFormed DevelopmentalAttribute)
```

will become:

$$\textbf{fun}\ \mathit{FullyFormed} : \mathit{Ind}\ \mathit{DevelopmentalAttribute}.$$

In SUMO, functions and predicates are represented as intances of a descendent of *Relation*. The representation specifies the return type and the type of its arguments. This information yields automatically to the GF representation of the function/predicate. For example:

```
(instance address BinaryPredicate)
(domain address 1 Agent)
(domain address 2 Address)
```

becomes in GF:

$$\textbf{fun}\ \mathit{address} : \mathit{El\ Agent} \rightarrow \mathit{El\ Address} \rightarrow \mathit{Formula}$$

In addition to this, SUMO also features axioms which specify the behaviour of relations and their connection to classes and instances. These axioms often use quantification. For example:

```
(=> (instance ?P Wading)
    (exists (?W) (and (instance ?W BodyOfWater) (located ?P ?W))))
```

Since GF is a strongly typed language, all variables should be declared with their corresponding type. We provide a simple type inference algorithm that searches for the usage of each variable, and assigns it the most general type that would make the axiom type-check. The corresponding axiom in GF would be:

$$\mathit{forall\ Wading}\ (\backslash P \rightarrow (\mathit{exists\ BodyOfWater}\ (\backslash W \rightarrow \mathit{located}\ (\mathit{el}\ P)\ (\mathit{el}\ W))))$$

The SUMO declarations of objects and relations constitute the SUMO-GF abstract syntax, while the other axioms are GF trees that are used for testing the natural language generation and for experiments in reasoning with the ontology.

During the translation of SUMO to GF, we discovered a number of small inconsistencies from the original ontologies like mismatches between instances and classes, usage of undefined objects and usage of functions used with wrong number of arguments. This represents almost 8% of the total number of axioms from SUMO and was determined automatically during the type checking phase.

4 Verbalization

The SUMO distribution provides natural language generation for 10 languages, via a set of string templates which are assembled together in order to construct phrases. For a language without sophisticated morphology, such as English, this turns out to be a satisfactory solution. On the other hand, for languages that feature gender agreement or case declension for nouns, the templates render naive constructions, that are not grammatically correct for most non-trivial cases. The usage of GF guarantees syntactically correct constructions.

Moreover, due to the design of the type system, variables from quantified formulae are assigned the gender of their corresponding type. This is a common feature of Romance languages, and would be difficult in an untyped setting.

Our work provides natural language generation for the two biggest modules *Merge* and *Mid-level-ontology* and two domain specific: *Elements* - featuring chemical substances and *Mondial* - featuring countries and cities of the world. A total of almost 7 000 objects and 500 relations from SUMO were verbalized in English. This process is done automatically for objects and semi-automatically

for relations, and uses the GF resource grammar [7] for English. The automatic process takes advantage of the camel case representation of SUMO concepts. For example, *BodyOfWater* will be rendered as "body of water" and parsed by GF as a noun phrase. For the two domain specific ontologies, the information is extracted from SUMO predicates that assert the connection of an entry with its name. The procedure used to verbalize functions and predicates is semi-automatical as it assumes building whole sentences and assumes more interaction from the user regarding the place and order of the arguments in the phrase.

As a result, our approach renders verbalization of a large number of entries from the ontology, with a high rate of automation, ensuring syntactical correctness of the generated phrases. For example :

For every unique list LIST, every positive integer NUMBER2 and every positive integer NUMBER1, we have that if the element with number NUMBER1 in LIST is equal to the element with number NUMBER2 in LIST, then NUMBER1 us equal to NUMBER2.

5 Conclusion

Our experience shows that using strongly typed language is promising from both knowledge engineering and linguistic point of view.

References

1. Ganesereth, M.R., Fikes, R.E.: Knowledge interchange format. Technical Report Logic-92-1, Stanford University (June 1992)
2. Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D.L., Patel-Schneider, P.F., Stein, L.A.: OWL web ontology language reference (February 2009)
3. Cycorp: The syntax of CycL (March 2002)
4. Ranta, A.: Grammatical Framework: A Type-Theoretical Grammar Formalism. The Journal of Functional Programming **14(2)** (2004) 145–189
5. Niles, I., Pease, A.: Towards a standard upper ontology. In: FOIS '01: Proceedings of the international conference on Formal Ontology in Information Systems, New York, NY, USA, ACM (2001) 2–9
6. Martin-Löf, P.: Constructive mathematics and computer programming. In Cohen, Los, Pfeiffer, Podewski, eds.: Logic, Methodology and Philosophy of Science VI. North-Holland, Amsterdam (1982) 153–175
7. Ranta, A.: The GF resource grammar library. Linguistic Issues in Language Technology **2**(2) (2009)