Generating Multilingual Grammars from OWL Ontologies

Guillermo Pérez, Gabriel Amores, Pilar Manchón, and David González

Universidad de Sevilla
Seville, Spain
{gperez, jgabriel, pmanchon, dgmaline}@us.es

Abstract. This paper describes a tool which automatically generates productions for context–free grammars from OWL ontologies, using just a rule–based configuration file. This tool has been implemented within the framework of a dialogue system, and has achieved several goals: it leverages the manual work of the linguist, and ensures coherence and completeness between the Domain Knowledge (Knowledge Manager Module) and the Linguistic Knowledge (Natural Language Understanding Module) in the application.

1 Introduction

1.1 Automatic Grammar Generation

The problem of manually generating grammars for a Natural Language Understanding (NLU) system has been widely discussed. Two main approaches can be highlighted from those proposed in the literature: Grammatical Inference and Rule Based Grammar Generation.

The Grammatical Inference approach (http://eurise.univ-st-etienne.fr/gi/) refers to the process of learning grammars and languages from data and is considered nowadays as an independent research area within Machine Learning techniques. Examples of applications based on this approach are ABL [1] and EMILE [2].

On the other hand, the Rule Based approach tries to generate the grammar rules from scratch (i.e. based on the expertise of a linguist), while trying to minimize the manual work. An example of this approach is the Grammatical Framework [3], whose proposal is to organize the multilingual grammar construction in two building blocks: an abstract syntax which contains category and function declarations, and a concrete syntax which defines linearization rules. Category and function declarations are shared by all languages and thus appear only once, while linearization rules are defined on a per–language basis. Methods which generate grammars from ontologies (including ours) may also be considered examples of the Rule Based approach.

1.2 Generating Grammars from Ontologies

In the context of Dialogue Management, the separation of the Knowledge Manager (the module in charge of the domain knowledge) and the NLU module poses a number of advantages: the complexity of the linguistic components is reduced [4], the existing domain knowledge may be reused [4], reference resolution processes (i.e: anaphoric resolution, underspecification, presupposition, and quantification) are better defined [5] , and, finally, it helps the dialogue manager in keeping dialogue coherence [4].

However, the information contained in the Knowledge Manager module overlaps with information inside the NLU module. The key idea of this paper is that this redundancy can be used to automatically generate grammar rules from the relationships between the concepts described in the ontology. Thus, the fact that the concept "lamp" is linked to the concept "blue" through a "hasColor" relationship somehow implies that sentences like "the blue lamp" should be correct in this domain and therefore accepted by the NLU grammar.

The generation of linguistic knowledge from ontologies has been proposed previously. Russ et al. [6] proposed a method for generating context–free grammar rules from JFACC ontologies. Their approach was based on including annotations all along the ontology indicating how to generate each rule. They implemented a program that was able to parse the ontology and produce the grammar rules.

A second precedent of linguistic generation from ontologies can be found in [7], where the author claimed that the concepts of an OWL ontology could be used to generate the lexicon of the NLU module.

In this paper a new rule–based solution for generating grammars from ontologies will be described. Section 2 motivates and gives an overview of the solution hereby proposed. Section 3 describes how the configuration files have to be built. Section 4 shows an introduction to the algorithm used. Section 5 includes real showcases of the tool at work. Section 6 is a summary of the conclusions and future work.

2 Solution overview

The solution proposed here is close to that of [6] in the sense that we also parse the ontology for the rule generation. Nonetheless, it differs in two ways:

Firstly, our approach argues that the ontology should remain as–is, without any specific linguistic annotation. Although it is obvious that the ontology itself is not descriptive enough to generate the grammar rules without additional information, we consider it preferable to place this information in a separate configuration file which describes how the grammar rules should be generated. This approach is also more convenient for a dialogue system (where the linguistic information in the ontology would be useless and cumbersome), and more suitable from a reusability point of view.

Secondly, OWL has been chosen instead of JFACC for two reasons:

1. The use of OWL is widely spread and seems to be the basis for the future semantic web. This implies that large ontologies are likely to be available in the future. Our approach will help create dialogue applications more easily by simply downloading specific domain ontologies.
2. OWL is based on RDF, and therefore uses Subject–Property–Object triplets. This static structure of OWL is of great help because the algorithm can focus on handling properties, letting the linguist define how to create rules which apply to all the elements in their Domain (or Range). It should be pointed out that this choice is not just a change in the ontology format: the whole parsing algorithm is based on RDFs predefined structure.

As previously mentioned, our approach focusses on grammar rules generation: no automatic lexicon hierarchy generation has been considered. To ensure coherence between the lexicon and the grammar, the list of potential non–terminal types is extracted from the list of all the entities within the ontology. The linguist decides which entities from this list shall remain in the final dialogue application.

It should be pointed out that this approach is meant only as a way of helping the linguist, and therefore it does not provide a ready–to–use grammar. By using this tool, the grammar will be easier to generate and more consistent with the domain knowledge, but, in any case, the resulting grammar must be checked and completed manually in a second step.

The current implementation provides grammar rules in a self–defined format [8] , which basically consists of a left–hand symbol followed by an arrow and a list of right–hand symbols. This notation can be easily translated to most standards, such as BNF.

3 Configuration files

As outlined above, the linguist must define a configuration file which will be used in conjunction with the ontology in order to generate the grammar rules. In this configuration file, the linguist has to identify the properties that may appear in the grammar and the way in which their domain and range will be included in the associated rules. In order to do it, an easy XML syntax has been defined (see DTD below).

Basically, the linguist can define the generation rules by means of nested *forEach* loops handling the properties (and subproperties) of the ontology, and using variables to identify the elements from its domain and range.

```
<!DOCTYPE rulesList [
  <!ELEMENT rulesList (forEach+)>
  <!ELEMENT forEach (forEach|rule+)>
  <!ELEMENT rule (left,right)>
  <!ELEMENT left (#PCDATA)>
  <!ELEMENT right (#PCDATA)>

  <!ATTLIST forEach property CDATA #IMPLIED>
  <!ATTLIST forEach subPropertyOf CDATA #IMPLIED>
  <!ATTLIST forEach domain CDATA #IMPLIED>
  <!ATTLIST forEach range CDATA #IMPLIED>

  <!ATTLIST rule lang (ES|EN|FR) #REQUIRED>
]>
```

In order to better understand this structure as well as the objective of the tool, a selection of showcases including those relevant in the ontology, the configuration file, and the resulting grammar rules are shown in the following sections.

4 Overview of the algorithm

This section describes in some detail the functions in the algorithm, which consists of three major steps:

1. Parse the OWL ontology. The goal of this step is to generate an internal representation of the relevant ontological elements. This representation will in turn be used to make queries over the ontology.
2. Parse the configuration file. The objective here is to generate the list of all applicable rules.
3. Generate the output rules. In this step, the script goes through the previous list of applicable rules, substituting the reference to classes and properties by the corresponding Input Form from the ontology.

The first two steps described have been implemented through a finite state machine (FSM) illustrated in figure 4.

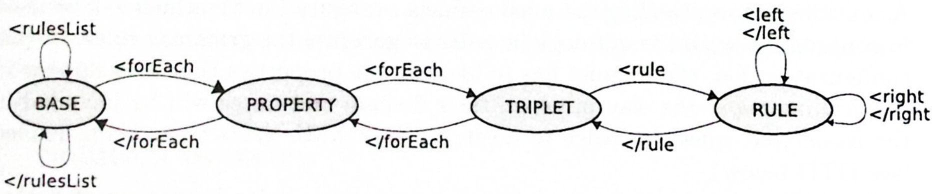

Fig. 1. FSM for the configuration file parser

For each state in the FSM, only one set of attributes can be parsed. These are mentioned in the previous DTD structure:

Base :
- No attributes are expected in this state.

Property :
- propertyRef: Indicates the word which refers to the property in the rule description.

  - subPropertyOf: Indicates a super property. All the sub properties of this one will be handled by the algorithm.

Triplet :
  - domainRef: Indicates the word which makes reference to the domain in the rule description.
  - rangeRef: Indicates the word which refers to the range in the rule description.

Rule :
  - lang: Indicates for which language the rule is valid.

5 Showcases

5.1 Sample Rules

The example below illustrates a common case in which the grammar rules will be generated. Our examples are taken from a smart–house domain in which the ontology describes both the hierarchy of devices in the house as well as the actions (or voice commands) which can be performed over those devices, such as "switch on the lamp in the kitchen". Thus, consider an ontology where a set of properties are grouped as subproperties of a general "hasDeviceCommand" property. These properties are shown graphically below:

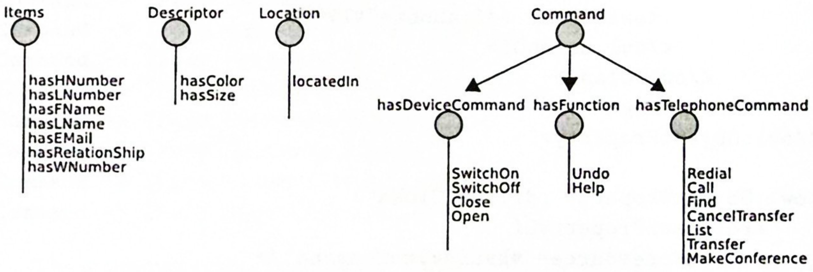

Fig. 2. Ontology Structure

In this showcase we are going to analyze the portion describing the device–related commands, whose XML equivalent is as follows:

```
<!-- hasDeviceCommand Subproperties -->

 <owl:ObjectProperty rdf:ID="SwitchOff">
  <rdfs:subPropertyOf
    rdf:resource="#hasDeviceCommand"/>
    <rdfs:domain rdf:resource="#System"/>
    <rdfs:range>
      <owl:Class>
```

```
        <owl:unionOf rdf:parseType="Collection">
           <owl:Class rdf:about="#Fan"/>
           <owl:Class rdf:about="#Heater"/>
           <owl:Class rdf:about="#Lamp"/>
           <owl:Class rdf:about="#Radio"/>
           <owl:Class rdf:about="#TV"/>
        </owl:unionOf>
     </owl:Class>
   </rdfs:range>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:ID="SwitchOn">
      <rdfs:subPropertyOf
          rdf:resource="#hasDeviceCommand"/>
      <rdfs:domain rdf:resource="#System"/>
      <rdfs:range>
          <owl:Class>
            <owl:unionOf
              rdf:parseType="Collection">
              <owl:Class rdf:about="#Fan"/>
              <owl:Class rdf:about="#Heater"/>
              <owl:Class rdf:about="#Lamp"/>
              <owl:Class rdf:about="#Radio"/>
              <owl:Class rdf:about="#TV"/>
            </owl:unionOf>
          </owl:Class>
      </rdfs:range>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:ID="Close">
      <rdfs:subPropertyOf
          rdf:resource="#hasDeviceCommand"/>
      <rdfs:domain rdf:resource="#System"/>
      <rdfs:range rdf:resource="#Blind"/>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:ID="Open">
      <rdfs:subPropertyOf
          rdf:resource="#hasDeviceCommand"/>
      <rdfs:domain rdf:resource="#System"/>
      <rdfs:range rdf:resource="#Blind"/>
  </owl:ObjectProperty>
```

In this particular case, the linguist has detected that all properties are actually actions, that is, they correspond to the "commands" to be performed by the system over all the elements in the range, that is, all the devices within the ontology. This can be easily expressed by the following configuration file:

```
<rulesList>
  <forEach property="Z" subPropertyOf="hasDeviceCommand">
    <forEach domain="X" range="Y">
      <rule lang="ES">
        <left>Command</left>
        <right>Z Y</right>
      </rule>
    </forEach>
  </forEach>
</rulesList>
```

Now, once the application is run indicating the appropriate configuration file, the following results are obtained [1]

```
Command -> IForm_SwitchOff IForm_Fan
Command -> IForm_SwitchOff IForm_Heater
Command -> IForm_SwitchOff IForm_Lamp
Command -> IForm_SwitchOff IForm_DimmerLamp
Command -> IForm_SwitchOff IForm_Radio
Command -> IForm_SwitchOff IForm_TV
Command -> IForm_SwitchOn IForm_Fan
Command -> IForm_SwitchOn IForm_Heater
Command -> IForm_SwitchOn IForm_Lamp
Command -> IForm_SwitchOn IForm_DimmerLamp
Command -> IForm_SwitchOn IForm_Radio
Command -> IForm_SwitchOn IForm_TV
Command -> IForm_Close IForm_Blind
Command -> IForm_Open IForm_Blind
```

The grammar rules obtained are semantically driven, without purely linguistic items like "Noun" or "Preposition". This is typically the case of dialogue systems grammars.

It should be noticed that even with this toy ontology, sixteen grammar rules have been generated using just two nested *forEach* loops.

5.2 Capturing Multimodality

Now let us assume the same scenario (i.e. the same ontology) but including multimodal entries; namely voice and pen inputs. Following Oviatt's results [9], it may be expected that mixed input modalities (voice: *switch this on*, pen: `clicks`

[1] The prefix "IForm" stands for "Input Form", used to identify non-terminal symbols in our self-defined format [8] . This prefix has no bearing on the algorithm: any other token could be used.

on the lamp icon) may also include alternative constituent orders, that is, different from the voice only input. The NLU module may therefore receive inputs such as: "lamp switch on" (verb at the end). [2]

This new set of rules can be easily accounted for by adding just one rule to the configuration file:

```
<rulesList>
   <forEach property="P"
            subPropertyOf="hasDeviceCommand">
      <forEach domain="X" range="Y">
         <rule>
            <left>Command</left>
            <right>P Y</right>
         </rule>
         <rule>
            <left>Command</left>
            <right>Y P</right>
         </rule>
      </forEach>
   </forEach>
</rulesList>
```

The new output will be the same as before but including these new rules:

```
Command -> IForm_Fan IForm_SwitchOff
Command -> IForm_Heater IForm_SwitchOff
Command -> IForm_Lamp IForm_SwitchOff
Command -> IForm_DimmerLamp IForm_SwitchOff
Command -> IForm_Radio IForm_SwitchOff
Command -> IForm_TV IForm_SwitchOff
Command -> IForm_Fan IForm_SwitchOn
Command -> IForm_Heater IForm_SwitchOn
Command -> IForm_Lamp IForm_SwitchOn
Command -> IForm_DimmerLamp IForm_SwitchOn
Command -> IForm_Radio IForm_SwitchOn
Command -> IForm_TV IForm_SwitchOn
Command -> IForm_Blind IForm_Close
Command -> IForm_Blind IForm_Open
```

5.3 Capturing Multilinguality

Due to the structural differences among human languages, different rules must be generated for different languages.

[2] Note that linguistically speaking this order is also possible in English in topicalized or left-dislocated constructions such as "The lamp, switch it on".

For example, in order to indicate the location of a given device, "the kitchen light" is said in English, whereas in Spanish the constituent order changes: "la luz de la cocina" (literally, the light of the kitchen).

Once the target language has been chosen, specific language rules may be generated.

Consider the following fragment taken from the ontology previously shown, describing which elements can be affected by the property "locatedIn":

```
<owl:ObjectProperty rdf:ID="locatedIn">
    <rdfs:domain>
        <owl:Class>
            <owl:unionOf
                rdf:parseType="Collection">
                <owl:Class rdf:about="#Lamp"/>
                <owl:Class rdf:about="#Radio"/>
                <owl:Class rdf:about="#Heater"/>
             </owl:unionOf>
         </owl:Class>
     </rdfs:domain>
     <rdfs:range>
        <owl:Class>
         <owl:unionOf
          rdf:parseType="Collection">
            <owl:Class rdf:about="#Bedroom"/>
            <owl:Class rdf:about="#Kitchen"/>
            <owl:Class rdf:about="#Hall"/>
            <owl:Class rdf:about="#LivingRoom"/>
         </owl:unionOf>
        </owl:Class>
      </rdfs:range>
</owl:ObjectProperty>
```

The multilingual configuration file which captures the structural differences mentioned above would be the following:

```
<rulesList>
   <forEach property="P"
    subPropertyOf="Location">
      <forEach domain="X" range="Y">
         <rule lang="ES">
            <left>X</left>
            <right>X P Y</right>
         </rule>
        <rule lang="EN">
            <left>X</left>
            <right>Y X</right>
        </rule>
      </forEach>
   </forEach>
</rulesList>
```

Now, if only English grammar rules are to be generated, the application must be run with the option "-lang=EN", which obtains the following results:

```
IForm_Lamp -> IForm_Bedroom IForm_Lamp
IForm_Lamp -> IForm_Kitchen IForm_Lamp
IForm_Lamp -> IForm_Hall IForm_Lamp
IForm_Lamp -> IForm_LivingRoom IForm_Lamp
IForm_Radio -> IForm_Bedroom IForm_Radio
IForm_Radio -> IForm_Kitchen IForm_Radio
IForm_Radio -> IForm_Hall IForm_Radio
IForm_Radio -> IForm_LivingRoom IForm_Radio
IForm_Heater -> IForm_Bedroom IForm_Heater
IForm_Heater -> IForm_Kitchen IForm_Heater
IForm_Heater -> IForm_Hall IForm_Heater
IForm_Heater -> IForm_LivingRoom IForm_Heater
```

6 Conclusions and future work

In this paper a novel rule-based approach to automatic grammar generation has been described. The solution proposed is based on OWL ontologies and provides linguists with an easy way to take advantage of the information contained within ontologies. This information extraction process will also be easier for the linguist if the ontology has been designed keeping in mind that grammars will be generated from it.

The solution proposed has achieved the expected goals: the linguist can generate a good number of rules from a simple configuration file and, by having the rules directly generated from the ontologies, domain knowledge and linguistic knowledge coherence and completeness is ensured. In addition, a rapid prototyping of new grammars for the speech recognizer and the NLU module is obtained by the same mechanism.

Future research areas include the generation of unification-based grammar rules and dialogue rules, and an evaluation of the usefulness of the tool with larger OWL ontologies.

7 Acknowledgements

This work was done under the TALK research project, funded by EU FP6 [ref. 507802] and the "Multilingual Management of Spoken Dialogues" project, funded by the Spanish Ministry of Education under grant TIC2002-00526.

References

[1] Zaanen, M.V.: Abl: Alignment–based learning. In: Proceedings of the 18th International Conference on Computational Linguistics (COLING), Saarbrücken (2000)

[2] Williem Adriaans, P.: Language Learning from a Categorial Perspective. PhD thesis, Amsterdam University (1992)

[3] Ranta, A.: Grammatical framework. a type–theoretical grammar formalism. The Journal of Functional Programming **14** (2004) 145–189

[4] Milward, D., Beverige, M.: Ontology–based dialogue systems. In: IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems. (2003)

[5] Quesada, J.F., Amores, G.: Knowledge–based reference resolution for dialogue management in a home domain environment. In Johan Bos, M.E., Matheson, C., eds.: Proceedings of the sixth workshop on the semantics and pragmatics of dialogue (Edilog). (2002) 149–154

[6] Russ, T., Valente, A., MacGregor, R., Swartout, W.: Practical experiences in trading off ontology usability and reusability. In: Proceedings of the Knowledge Acquisition Workshop (KAW99), Banff, Alberta (1999)

[7] Estival, D., Nowak, C., Zschorn, A.: Towards ontology-based natural language processing. In: RDF/RDFS and OWL in Language Technology: 4th Workshop on NLP and XML, Barcelona, Spain, ACL (2004)

[8] Amores, G., Quesada, F.: Episteme. Procesamiento del Lenguaje Natural **21** (1997) 1–16

[9] Sharon Oviatt, S.L., DeAngeli, A., K., K.: Integration and synchronization of input modes during multimodal human–computer interaction. In: Proceedings of Conference on Human Factors in Computing Systems: CHI '97. (1997)