University of Zurich
Department of Informatics

Diploma Thesis October 17, 2007

Ginseng Goes Italian

Adding Multilingualism to a Natural Language Search Engine

Ursula D'Onofrio
of Littau LU, Switzerland

Student-ID: 01-916-683
ursula@access.uzh.ch

Advisor: **Esther Kaufmann**

Prof. Abraham Bernstein, PhD
Department of Informatics
University of Zurich
http://www.ifi.uzh.ch/ddis

Abstract

The popularity of the Internet is growing steadily and with it the amount of data shared online. As a consequence, it is getting more and more difficult to find and organize the data on the Web. With the Semantic Web a way to structure this data has been provided, and to query it, SQL-like query languages were developed. Since these formal languages are not likely to be used by "standard" users, the development of natural language interfaces for querying Semantic Web knowledge bases has become a popular subject. The *Ginseng* project approaches this subject by offering a controlled natural language query interface for the Semantic Web to the user. In this thesis, the extension of the Ginseng application to a multilingual system as well as the implementation of a user interface for managing the semantic annotation of ontologies in Ginseng are presented.

Zusammenfassung

Die Beliebtheit des Internets wächts stetig und damit die Menge an online gespeicherten Daten. Eine Möglichkeit, Webinhalte strukturiert zu speichern, stellt das *Semantic Web* dar, welches es erlaubt, Metadaten zusammen mit Webinhalten abzuspeichern. Um die Daten aus dem Semantic Web abzufragen, wurden SQL-ähnliche formale Angfragesprachen entwickelt, welche aber wegen ihrer Komplexität nicht für jede/n zugänglich sind. Aus diesem Grund beschäftigen sich heute mehrere Projekte mit dem Zugriff auf *Semantic Web* Daten über natürlichsprachliche Benutzerschnittstellen. Darunter auch das Projekt *Ginseng*. In dieser Arbeit wird die Erweiterung der Applikation Ginseng zu einem mehrsprachigen System vorgestellt sowie die Entwicklung einer Benutzerschnittstelle für die semantische Annotation von Ontologien in Ginseng.

Table of Contents

1

Introduction

The popularity of the Internet is growing steadily and with it the amount of data shared online. As a consequence, it is getting more and more difficult to find and sort the data we need. How is it possible to separate important from less relevant information? Although some search engines, like *Google*[1] are able to achieve fairly good results, even better results could be achieved if the searched data was stored in a more structured way. The *Semantic Web* allows to store information along with metadata making it better processable by machines [Berners-Lee et al., 2001].

To query the Semantic Web SQL-like query languages were developed, which allow more specific queries than the full-text search used on the Internet. But how give a regular user, who does not have any specific knowledge, access to a formal query language? In the last few years various efforts have been made developing *Natural Language Interfaces (NLIs)* for Semantic Web data, which enable the user to query a knowledge base using natural language. Examples of projects which combine NLIs with Semantic Web knowledge bases are: *NaturalOWL* [Androutsopoulos and Galanis, 2007], *ORAKEL* [Ishikawa et al., 1986], and *Ginseng* [Bernstein et al., 2006].

This thesis is part of the Ginseng project. The previous versions of the application Ginseng were implemented for English natural language questions only. The first goal of this thesis was to examine if the Ginseng system could support two languages at the same time by integrating the Italian language into the Ginseng system. In the end the system should be able to answer English as well as Italian queries.

The second goal was to extend Ginseng with a tool for managing natural language information in the knowledge bases used by the system.

This thesis explains how these problems were solved, presents what the achieved results were, and their evaluation.

[1] http://www.google.com

The structure of the thesis is the following: After a short introduction to the Semantic Web, RDF, and OWL, Ginseng will be introduced. Then, the changes made to the Ginseng system to support multilingualism will be outlined as well as the implementation of the Synonym Editor. The results of the final evaluation will then be presented in the following section. Finally, future work and conclusions will complete the thesis.

2

The Semantic Web

The *Semantic Web* is an extension of the World Wide Web (WWW) [Berners-Lee et al., 2001]. However, in addition to the information displayed on a regular Web page of the WWW, the Semantic Web also stores semantic metadata. In the Oxford Dictionary the word "semantic" is defined as: "connected with the meaning of words." [Wehmeier, 2000]. Relating to the Semantic Web this means that the semantic metadata it stores represents the meaning of words, that is, it adds meaning to the information on the WWW.

Essentially, the purpose of the metadata is to make the Web content better accessible to computers. The data of the WWW is stored, displayed, and processed by machines, but it can only be interpreted, understood, and used by humans. This is why specific information for machines which is defined formally is needed to make the Web content machine-readable. [Breitman et al., 2007]

For the formal definition of the data the *Resource Description Framework (RDF)*, which will be introduced in the next section, was developed. RDF provides structures to define data objects and relations between them [Berners-Lee et al., 2001].

However, the Semantic Web not only adds metadata to the WWW, it also provides programs to query this data and intelligent agents [McCool, 2005], autonomous computer programs able to retrieve, interpret, and use the semantic information to perform their tasks.

For a more extensive introduction to the Semantic Web the reader is referred to [Breitman et al., 2007], [W3C, 2007], and [Manola and Miller, 2004].

2.1 The Resource Description Framework

The W3C has published a recommendation [Klyne and Carroll, 2004] for RDF, which provides a detailed description of RDF. In the next lines a summary of the recommendation will be provided to give the reader an idea of what RDF is.

The purpose of RDF is to specify a basic data model for the definition of metadata. This metadata, which is represented in an XML syntax, has to be machine processable and interchangeable between applications.

To include semantics, the data model has to be able to define objects and the relations between those objects. This is achieved with the help of *RDF Triples*, which describe subject-predicate-object relationships. This type of relationship can be explained with an example: The subject be "State", the predicate "hasCity" and the object "City". That means that "State" is the subject which has the property "hasCity" and "City" is the value of this property. Figure 2.1 shows the concept of the triples graphically.

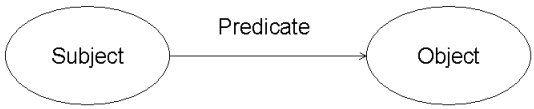

Figure 2.1: RDF Triple [Klyne and Carroll, 2004]

In figure 2.2 the example *State -> hasCity -> City* is visualized:

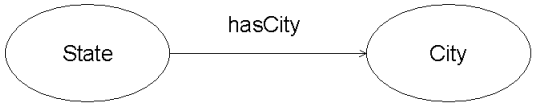

Figure 2.2: RDF Triple *State -> hasCity -> City*

RDF identifies its objects, called resources or properties, by URI references. Every document which uses a formal language based on RDF has a URI of the form:

`http://www.something.org/something/else#resourcename`.

The first part of the URI ends with the character # and is called *URI prefix*. The string after the character # contains the specific name of a resource or property. For example, all the URIs of the elements which are part of the RDF specification start with the prefix:

`http://www.w3.org/1999/02/22-rdf-syntax-ns#`

and end with strings like “type”, “resource”, and “value”. These URIs can be used in any document to refer to elements defined in the RDF specification.

The URI prefixes which end with the character # correspond to the namespace names identifying namespaces in XML.

> ”An XML *namespace* is a collection of names, identified by a URI reference [RFC2396], which are used in XML documents as element types and attribute names.” [Layman et al., 1999]

Since the namespace names are quite long abbreviations for them can be defined. These abbreviations are called namespace prefixes and are defined in the namespace declarations. Below, the declaration of the RDF namespace is shown:

```
xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
```

The string “xmlns” is an XML keyword and designates the start of a namespace declaration, “rdf” is the prefix of the declared namespace and `"http://www.w3.org/1999/02/22-rdf-syntax-ns#"` is the name of the namespace.

URIs and namespaces are essential to the Semantic Web as they provide a distributed and effective adressing scheme [Breitman et al., 2007].

2.1.1 RDF Schema

With RDF resources and subject-predicate-object relationships between them can be defined. However, it is not possible to describe classes and properties with RDF. This is what the W3C recommendation *RDF Schema (RDF-S)* [Guha and Brickley, 2004] is for. RDF-S extends RDF and specifies its vocabulary description language. It provides mechanisms for describing RDF properties and relationships between those properties. RDF-S is also able to describe groups of related resources, such as classes [Guha and Brickley, 2004] and hierarchies of classes [Breitman et al., 2007]. Examples of properties defined by RDF-S are [Guha and Brickley, 2004]:

subClassOf: Defines a hierarchical relation between classes and states that all the instances of a class are instances of another class.

subPropertyOf: Defines a hierarchical relation between properties and states that all the resources that are related by a property are related by another.

domain: Restricts the resources that have a property to be instances of a defined set of classes.

range: Restricts the values of a property to be instances of a defined set of classes.

Although resources and relationships can be defined with RDF and RDF-S, as the name says, they are just part of a framework and are not sufficient for the actual representation of Web content. This is why, various formal languages for the description of Web content were developed. These languages build on RDF and extend it to support more concepts. Examples are the *Ontology Inference Layer (OIL)*, the *DARPA Agent Markup Language (DAML)*, or the *Web Ontology Language (OWL)*. In this thesis the focus will be on OWL, as it is the language used in the Ginseng project.

2.2 The Web Ontology Language

Like RDF, OWL is a W3C recommendation. In the following, the information about OWL given in [McGuinness and van Harmelen, 2004] will be summarized. However, before getting to the details of OWL, the term "ontology" has to be clarified. In the field of computer science an ontology is a data model that defines concepts and relations among those concepts within a finite domain. The domain can be anything, e.g., the geography of the United States, the human body, or the university of Zurich. To describe an ontology, an ontology definition language like OWL is required.

A good explanation of the purpose of OWL is given in the W3C recommendation:

> "The OWL Web Ontology Language is designed for use by applications that need to process the content of information instead of just presenting information to humans. OWL facilitates greater machine interpretability of Web content than that supported by XML, RDF, and RDF Schema (RDF-S) by providing additional vocabulary along with a formal semantics." [McGuinness and van Harmelen, 2004]

OWL is based on RDF and provides more concepts and structures than RDF. In fact, OWL can not only be used to describe ontology schemas but also to define instances of classes defined in a schema. For example, if the class "City" is defined in an ontology, its instances "New York", "Los Angeles", and "Chicago" could be defined in OWL. An OWL ontology schema together with its instances represents an OWL knowledge base.

There are three versions of OWL: the sublanguages OWL Lite, OWL DL and OWL Full [McGuinness and van Harmelen, 2004]. OWL Lite is less expressive than OWL DL,

which is less expressive than OWL Full. Which of the three sublanguages to use, has to be decided by the author of the ontology depending on the complexity of the facts that need to be express.

To better understand how to define an ontology and its instances an example is presented below. All the definitions in the example are part of the ontology defined at *www.mooney.net/geo* [Tang and Mooney, 2001], which describes facts about the geography of the United States.

In the first listing the classes "State", "City" and "Capital" are defined. The marked keyword *Class* labels a class definition, whereas the *rdf:ID* defines the name of the class. Moreover, a hierarchical relationship between "City" and "Capital" is defined with the help of the RDF-S property *subClassOf*. The relationship states: A "Capital" is a subclass of a "City".

```
<owl:Class rdf:ID="State"></owl:Class>

<owl:Class rdf:ID="City"></owl:Class>

<owl:Class rdf:ID="Capital">
   <rdfs:subClassOf rdf:resource="#City"/>
</owl:Class>
```

Listing 2.1: Definition of the Classes "State", "City" and "Capital" at *www.mooney.net/geo*

Next, the *OWL datatype property* "cityPopulation" is defined for the class "City".

> "Datatype properties [are binary] relations between instances of classes and RDF literals and XML Schema datatypes" [Welty et al., 2004]

For each datatype property domain and range have to be defined. In listing 2.2 the domain of the property "cityPopulation" is the class "City", its range the XML Schema datatype "float". This means, the value of the property "cityPopulation" must be a number. This number will represent the population size of a city.

```
<owl:DatatypeProperty rdf:ID="cityPopulation">
   <rdfs:domain rdf:resource="#City"/>
   <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
</owl:DatatypeProperty>
```

Listing 2.2: Definition of the Datatype Property "cityPopulation" at *www.mooney.net/geo*

In the next listing the *OWL object property* "isCityOf" is displayed.

> "Object properties [are binary] relations between instances of two classes" [Welty et al., 2004].

In this example the object property "isCityOf" defines a relation between the classes "City" and "State". As for datatype properties domain and range have to be specified for object properties.

```
<owl:ObjectProperty rdf:ID="isCityOf">
   <rdfs:domain rdf:resource="#City"/>
   <rdfs:range rdf:resource="#State"/>
</owl:ObjectProperty>
```

Listing 2.3: Definition of the Object Property "isCityOf" at *www.mooney.net/geo*

Finally, an instance of the class "City" is shown in the last listing of this section. Besides the label specifying the English name of the city represented by this instance, values for the properties "isCityOf" and "cityPopulation" are defined. For each class an infinite number of instances can be created.

```
<City rdf:ID="chicago">
   <rdfs:label xml:lang="en">chicago</rdfs:label>
   <isCityOf rdf:resource="#illinois" />
   <cityPopulation>3005172</cityPopulation>
</City>
```

Listing 2.4: Instance of the Class "City" at *www.mooney.net/geo*

In this chapter the Semantic Web, an extension of the WWW which stores metadata together with Web content, was introduced. The means for representing information on the Semantic Web are provided by RDF, RDF-S and OWL. Although also other ontology definition languages exist, we focused on OWL, as it is the ontology definition language used in Ginseng. Ginseng and its use of OWL will be introduced in the next chapter.

3

Ginseng

After introducing the basics of the Semantic Web, the actual subject of this thesis can be discussed, namely Ginseng. The acronym Ginseng stands for ***G**uided **I**nput **N**atural Language **S**earch **Eng**ine*. Ginseng is an application which enables the user to query a Semantic Web knowledge base using a controlled natural language interface. Although the user can ask natural language questions, he/she is not free to enter any question he/she wants into the interface. The system guides the user and suggests the words that can be used to complete sentences with the help of a pop-up list. By restricting the possible entries Ginseng ensures that only questions the system is able to process can be entered. The Ginseng user interface with the pop-up list that guides the user through the queries is shown in figure 3.1.

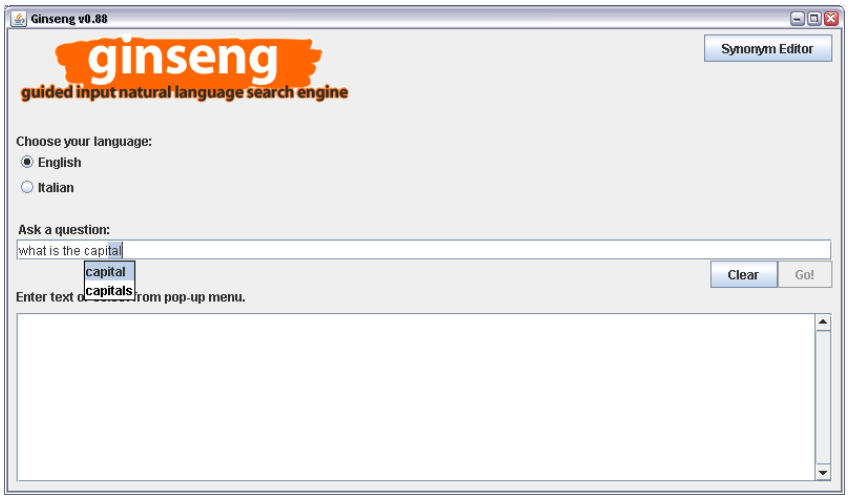

Figure 3.1: The Ginseng User Interface

To suggest possible completions of words Ginseng uses an ontology-independent grammar, which is dynamically extended with lexical information read from OWL knowledge bases loaded into the system at startup. The Ginseng grammar will be introduced in section 3.2.

The previous versions of Ginseng only provided the functionality to ask questions in English. Within this thesis Ginseng was extended to support multiple languages. The current implementation supports English and Italian, but more languages can be added easily. The details of the multilingual extension of Ginseng will be discussed in chapter 4. First, though more information on Ginseng will be given.

3.1 The Ginseng Architecture

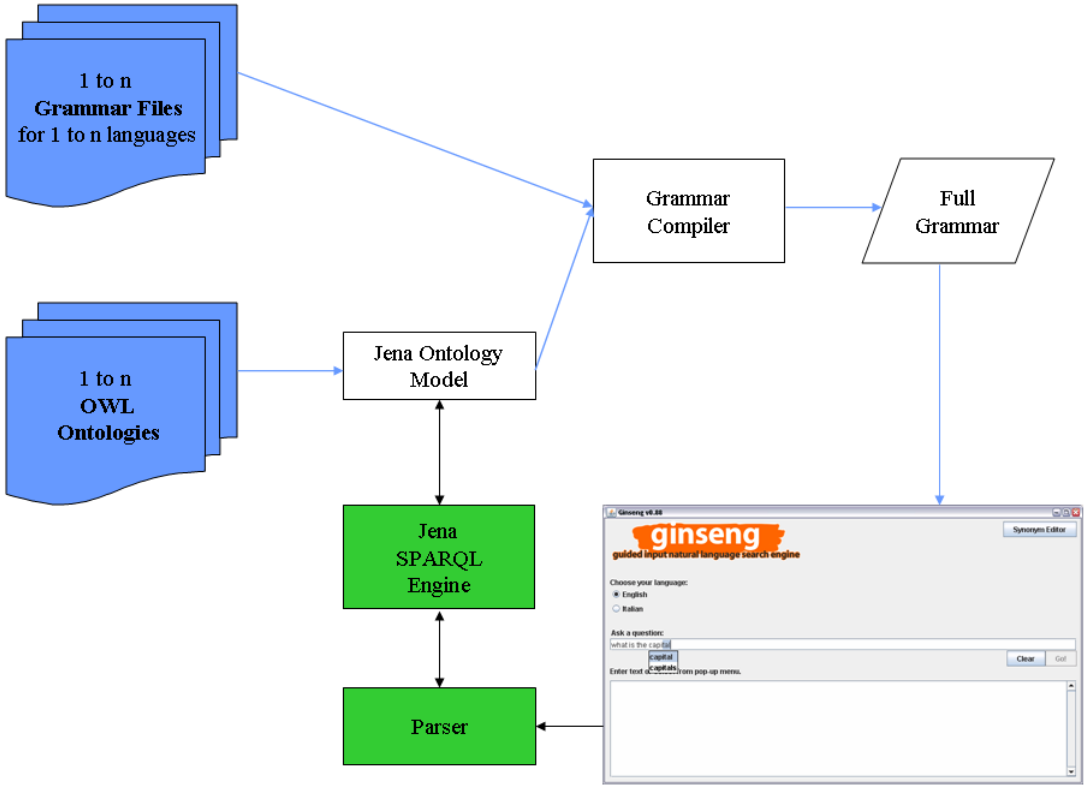

Figure 3.2: The Ginseng Architecture

> "Ginseng's architecture has three parts: a partially dynamically generated multi-level grammar, an incremental parser, and the ontology-access layer." [Bernstein et al., 2006]

The multi-level grammar, as well as the parser were developed within the Ginseng project, as an ontology-access layer, however, the open source framework *Jena*[1] was used.

Furthermore, to formally process the queries the *SPARQL Query Language* [Seaborne and Prud'hommeaux, 2007], which is designed to query RDF data, was applied. Besides these technical tools, Ginseng also needs data resources: on the one hand a grammar file for each supported language, on the other hand one or more OWL knowledge bases.

When started, Ginseng loads all the grammar files it finds from a predefined location into its *Thesaurus*, implemented by a tree data structure in Java. This results in the Thesaurus holding the static grammar rules for each language. Then, additional vocabulary items, which are combined with the static grammar rules, are extracted from the OWL knowledge bases to complete the Ginseng rule set. [Bernstein et al., 2006]

After this process, Ginseng is started and ready to accept natural language questions from the user. Besides the information for the composition of the natural language questions, the Ginseng grammar also contains elements for the composition of the SPARQL queries. Therefore, the grammar is not only used to guide the user through the natural language queries, but also to generate the SPARQL queries. The formal processing of the queries is then taken care of by Jena, which returns the query results to Ginseng in order to present them to the user. This process is described more precisely in [Bernstein et al., 2005], [Bernstein et al., 2006], and [Bernstein and Kaufmann, 2006].

The structure of the Ginseng architecture is visualized in figure 3.2. The blue arrows indicate the startup process, where the grammatical and lexical information is loaded from the input files. The green boxes, on the other hand, represent the resources used for the formal query processing, that is Jena and SPARQL.

In the following section the Ginseng grammar will be examined more closely. The parser and the query processing however, will not be discussed any further, as they are not central to this thesis. Detailed information on the parser and the Ginseng architecture in general can be found in [Kaiser, 2004].

[1] http://jena.sourceforge.net/

3.2 The Ginseng Grammar

The Ginseng grammar is ontology-independent, which means, it can be used with any knowledge base, as long as it supports SPARQL. That means, the knowledge bases used with the Ginseng grammar must be written in an ontology definition language that can be queried in SPARQL, such as OWL.

The grammar does not specify a set of rules by which sentences can freely be constructed, but it defines the syntax of a limited set of sentences in quasi BNF form.[2] This static sentence structure is completed by a dynamically generated vocabulary, extracted from the ontologies loaded in Ginseng, resulting in the final set of Grammar rules used in Ginseng. For each supported language Ginseng needs a separate grammar file. In listing 3.1 an extract from the English grammar is listed.

```
<START> ::= <SPO_OBJ> ?
            |SELECT <<SPO_OBJ>>
            |WHERE (<<SPO_OBJ>>)
            |<<SPO_OBJ>>

<SPO_OBJ> ::= list all <class>|?class|?class <<class>>

<class> ::= <NC>|<<NC>>|<rdf-syntax-ns#type> <<NC>>

<NC> ::= <NCc>|<<NCc>>|<<NCc>>

<NC> ::= <NCv>|<<NCv>>|<<NCv>>
```

Listing 3.1: Extract from the English Ginseng Grammar

Every question in Ginseng begins with the <START> rule, which defines the beginning of a natural language question and the basic structure of a SPARQL query: "SELECT ... WHERE...". Starting from the <START> rule and by replacing the *non-terminal symbols*[3] by other rules or *terminal symbols*[4], the final rule is built. Non-terminal symbols stand within the symbols "<" and ">" and have to be replaced by expressions that define the corresponding elements. In the expression <START> for example, the elements represented by <SPO_OBJ> have to be replaced by:

`list all <class>|?class|?class <<class>>`

[2] "Quasi BNF" because the symbol "|" is not used to express the logical "OR" but to separate the different parts of each rule.

[3] Non-terminal symbols are elements which have to be replaced by other elements (terminal or non-terminal.)

[4] Terminal symbols are strings which do not need to be replaced any further in a rule.

resulting in:

```
list all <class>?|SELECT ?class|WHERE (?class <<class>>)|?class.
```

The non-terminal symbols have to be replaced, until the expression only contains terminal symbols and non-terminal symbols which are not defined within the grammar file. These remaining non-terminal symbols have to be replaced by vocabulary items extracted from ontologies. In the example shown in listing 3.1 the replacing of non-terminal symbols has to be performed until terminal symbols and the elements <NCc> or <NCv> are left. In the definition of <class>, where the element <NC> has to be replaced, two different options are possible: <NCc> and <NCv>. When the rule set is built in Ginseng, both options are stored in the parse tree. Which of the two options is chosen and hence, which branch to follow in the parse tree, is determined by the users choice. (As the user choses the next words for his/her question on the user interface the branch to take in the parse tree is automatically determined.) Below, the final result of the rule composition with the elements out of listing 3.1 is shown:

```
<START> ::= list all <NCc> ?|
            SELECT ?class |
            WHERE (?class <rdf-syntax-ns#type> <<NCc>>)

<START> ::= list all <NCv> ?|
            SELECT ?class |
            WHERE (?class <rdf-syntax-ns#type> <<NCv>>)
```

Listing 3.2: Resulting Ginseng Grammar Rules

The remaining, non-terminal symbols that can not be substituted by any expression in the grammar, like <NCc> and <NCv> have to be replaced by the lexical items extracted from the loaded knowledge bases.

3.3 Enhanced Ontologies for Ginseng

Any OWL ontology can be loaded into Ginseng, although the usability of the system can be increased by using ontologies to which additional lexical information was added. This information appears in the ontologies in the form of properties, called *Ginseng Phrases* or *Synonyms*. These properties are defined in the Ginseng namespace and their values are used, when loaded into Ginseng, to complete the static grammar rules. The Ginseng namespace defines the following classes:

- ignore
- phrase
- interrogative

If a class appears to have the *ignore* property it means that its classname, that is its RDF ID, should not be used to build grammar rules. This is necessary, because the names of the defined resources are not always meaningful. The *phrase* property designates a synonym of the classname, the *interrogative* property defines text elements that can be used to compose rules for sentences starting with interrogative words or other interrogative sentence beginnings.

To build the final rule set for Ginseng, for every class, datatype property, object property, and instance a grammar rule is generated [Bernstein et al., 2006] and added to the Thesaurus to complete the static grammar rules (see section 3.1).

The following examples show how synonyms can be used to enrich OWL ontologies. To see the difference between ontologies with and without Ginseng Phrases, compare the definitions in the listings 3.4 to 3.6, to the ones in listing 3.3, which shows "standard" OWL definitions without Ginseng Phrases:

```
<owl:Class rdf:ID="City"></owl:Class>

<owl:DatatypeProperty rdf:ID="cityPopulation">
   <rdfs:domain rdf:resource="#City"/>
   <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
</owl:DatatypeProperty>

<owl:ObjectProperty rdf:ID="isCityOf">
   <rdfs:domain rdf:resource="#City"/>
   <rdfs:range rdf:resource="#State"/>
</owl:ObjectProperty>
```

Listing 3.3: Definition of OWL Classes and Properties without Ginseng Phrases

In the next listing the class "City" is defined including two synonyms: "cities" and "metropolis". In this case there is no need for the *ignore* property because the classname is meaningful and suited for the construction of grammar rules. Note that also the plural form "cities" is defined as a Ginseng Phrase of the class "City". Since Ginseng does not implement the pluralization of words the plural forms have to be stored as Ginseng synonyms.

```
<owl:Class rdf:ID="City">
   <ginseng:phrase xml:lang="en" rdf:value="cities"/>
   <ginseng:phrase xml:lang="en" rdf:value="metropolis"/>
</owl:Class>
```

Listing 3.4: Definition of the Class "City" with Ginseng Phrases

Next, the declaration of the datatype property "cityPopulation" is shown. Since the property name "cityPopulation" is not meaningful, it can not be used as a vocabulary item in any grammar rule. Thus, the *ignore* property must be used. Furthermore, the *interrogative* property "how big" is defined. It states that questions which start with "how big" and which refer to the class "City" will query for the population of a city. For example, it could be used to construct the question: "How big is Chicago?".

```
<owl:DatatypeProperty rdf:ID="cityPopulation">
   <rdfs:domain rdf:resource="#City"/>
   <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
   <ginseng:ignore rdf:value="id text"/>
   <ginseng:phrase xml:lang="en" rdf:value="population"/>
   <ginseng:phrase xml:lang="en" rdf:value="size"/>
   <ginseng:interrogative xml:lang="en" rdf:value="how big"/>
</owl:DatatypeProperty>
```

Listing 3.5: Definition of the Datatype Property "cityPopulation" with Ginseng Phrases

The last listing shows the definition of the object property "isCityOf". In this declaration too, the *ignore* property is needed to keep the property name "isCityOf" from appearing in a grammar rule.

```
<owl:ObjectProperty rdf:ID="isCityOf">
   <rdfs:domain rdf:resource="#City"/>
   <rdfs:range rdf:resource="#State"/>
   <ginseng:ignore rdf:value="id text"/>
   <ginseng:phrase xml:lang="en" rdf:value="is city of"/>
   <ginseng:phrase xml:lang="en" rdf:value="lies in"/>
   <ginseng:phrase xml:lang="en" rdf:value="is the city in"/>
   <ginseng:phrase xml:lang="en" rdf:value="are the cities in"/>
   <ginseng:phrase xml:lang="en" rdf:value="is located in"/>
   <ginseng:phrase xml:lang="en" rdf:value="is located in the state"/>
   <ginseng:phrase xml:lang="en" rdf:value="lie in"/>
</owl:ObjectProperty>
```

Listing 3.6: Definition of the Object Property "isCityOf" with Ginseng Phrases

In this chapter Ginseng, a natural language query interface for OWL knowledge bases, was introduced. The primary focus was on the Ginseng grammar and the construction of the rule set for Ginseng, as they are essential for the understanding of the following chapters. As from now the results accomplished within this thesis will be described and how they were achieved. Thus, in the next section the integration of Italian as a query language into the Ginseng system will be presented.

4

Multilingual Support for Ginseng

Originally, Ginseng was implemented to accept and answer English natural language questions only. However, since the latest version, Ginseng can be used with two languages: English and Italian. In the future, it might support more languages, as they can be added easily.

In this chapter the changes made to the Ginseng system to support the Italian language will be presented. First, the modification of the implementation of Ginseng itself will be exposed, then, the changes made to the ontologies to support multiple languages will be presented, and finally, the newly created Italian grammar will be introduced.

4.1 Changes to the Implementation

The original implementation of Ginseng was changed in three ways: First, a static *Language Manager* class that keeps track of the languages currently supported by the tool had to be implemented. Additionally, the Language Manager also provides several static methods to retrieve the language names or their abbreviations from the data.

Second, the *Thesaurus* class had to be extended. Before, it was implemented with one tree, holding only English grammar rules. Now, there is one tree per language, holding the corresponding rules and a *getTree* method was implemented, which takes a language name as a parameter and returns the correct tree of grammar rules.

Third, all the existing methods which are part of the process of retrieving Ginseng Phrases from ontologies had to be extended. Each of these methods had to be modified in such a way that when a string declaring a Ginseng property is parsed, not only the

synonym, but also the language label is retrieved and stored. In order that when the Ginseng Phrases are used to build grammar rules, the program knows which language to combine that phrase with.

Of course, the user interface too had to be adapted. To enable the user to choose the language he/she likes for querying a knowledge base, a group of optionboxes was added to the interface, the two options being "English" and "Italian".

If more languages need to be added to Ginseng the following implementation changes are necessary:

1. In the Language Manager the list of supported languages has to be updated; moreover, its methods must be adapted to return the correct names and abbreviations for the new languages.

2. For each new language an additional tree must be added to the Thesaurus; the getTree method must be adapted to return the correct tree for the newly added languages.

To the user interface no changes are needed when a new language is added: the program reads the number and the names of the supported languages from the Language Manager and autonomously creates the required checkboxes on the interface.

4.2 Changes to the Ontology

The ontologies used in Ginseng were introduced in section 3.3, along with the Ginseng Phrases. It was mentioned that the annotation of ontologies with synonyms can increase the usability of the Ginseng system. To benefit from this increase in usability in the multilingual Ginseng system too, Ginseng Phrases have to be specified in the used ontologies for all the supported languages. Since we integrated Italian as a natural query language into the system, Italian phrases were added to each class and property of the ontology used at that time in Ginseng.

Having Ginseng Phrases in different languages in the class and property definitions, it became necessary to define a way to designate the language of each synonym. For this purpose, an additional XML-tag was added to the declaration of the Ginseng Phrases: *xml:lang="en"*. Listing 4.1 shows the definition of the class "City" for which English as well as Italian phrases are defined. It is also visible in this listing that the class "City" has the property *ginseng:ignore*. When more than one language is used in an ontology, the classname must not be used to build the grammar rules in Ginseng. Otherwise, the

classname would appear in the rule set of all the supported languages. In listing 4.1, for example, without the **ignore** property the word "city" would come up in the pop-up menu for English and Italian questions, although "city" is not an Italian word. This is why the *ginseng:ignore* property has to be added to all the classes and properties of multilingual ontologies for Ginseng.

```
<owl:Class rdf:ID="City">
   <ginseng:ignore rdf:value="id text"/>
   <ginseng:phrase xml:lang="en" rdf:value="city"/>
   <ginseng:phrase xml:lang="en" rdf:value="cities"/>
   <ginseng:phrase xml:lang="en" rdf:value="metropolis"/>
   <ginseng:phrase xml:lang="it" rdf:value="citta"/>
   <ginseng:phrase xml:lang="it" rdf:value="metropoli"/>
</owl:Class>
```

Listing 4.1: An OWL Class with English and Italian Ginseng Phrases

No changes and additions were made to the instance declarations in multilingual knowledge bases. This means, the instance names used for building the final grammar rule set in Ginseng are the same for all the supported languages. However, since the XML-language tag (xml:lang) could also be used for instances, in the future also instances could be named in various languages. This would be advantageous, because it would make the vocabulary used in the natural language questions more consistent. For example in the knowledge base defined at *www.mooney.net/geo* for the class "State" the instance "northDakota" is specified:

```
<State rdf:ID="northDakota">
  <rdfs:label xml:lang="en">North Dakota</rdfs:label>
  <abbreviation>nd</abbreviation>
  <statePopulation>70700</statePopulation>
  <stateArea>652700</stateArea>
  <statePopDensity>0.10831929</statePopDensity>
</State>
```

Listing 4.2: Instance of the Class "State" at *www.mooney.net/geo*

The English label of this instance is "North Dakota", which is the name of the state represented by this instance. The Italian name of this state, however, is "Dakota del Nord". To add this information to the instance definition the following line would have to be added to it:

```
<rdfs:label xml:lang="it">Dakota del Nord</rdfs:label>
```

If the language label is only defined in English, in Ginseng it appears in English and Italian questions. This is not a big problem when using geographic data, but with other data which has more language-specific names, the mix of words read from the grammar files and lexical information extracted from knowledge bases could become hard to understand. Thus, it will probably be unavoidable in the future to introduce language-dependent instance labels in Ginseng or to have language-specific knowledge bases anyway. To support instances with language-dependent labels Ginseng's implementation would have to be adapted.

4.3 The Italian Grammar

To allow natural language queries in Italian, an Italian grammar had to be developed. The goal was to define the syntax for the same types of queries, as the English grammar. Therefore, the English grammar was used as a guideline. Below, three types of questions are listed together with both an English and an Italian example:

Querying for a property of a class:
English: How big is California?
Italian: Quanto è grande la California?

Queries asking to list all the instances of a class:
English: List all cities.
Italian: Elenca tutte le città.

Queries to find out the type of a class or property:
English: What is Alabama?
Italian: Cos'è l'Alabama?

Although some rules can easily be translated from English to Italian, it is not always possible to do so. The main difference between the English and the Italian grammar is that in the Italian one more non-terminal symbol had to be introduced. Most of them specify Italian pronouns. This is due to the fact that in English, wh-words[1] and definite and indefinite articles show no grammatical gender (feminine and masuline) and number

[1] Wh-words are words used in questions such as "which" or "what".

Word Category	English	Italian
Wh-words	where	dove
	what	cosa
		quale
		quali
	which	quale
		quali
	how many	quanti
		quante
	how much	quanto
		quanta
	how <adj>	quanto è <adj>
	how <adj>	quanto sono <adj>
Definite Article	the	il
		lo
		la
		l′
		i
		gli
		le
Indefinite Article	a/an	un
		uno
		una
		un′

Table 4.1: Comparison of English and Italian Interrogative Words and Determiners Used in the Ginseng Grammar

English Presposition	Italian Presposition
of	di
	del
	dello
	della
	dell′
	dei
	degli
	delle

Table 4.2: Comparison of English and Italian Prepositions Used in the Ginseng Grammar

(singular and plural), in Italian they do. In table 4.1 a comparison of English and Italian interrogative words and determiners is displayed.

The same holds for prepositions, which in Italian are often used in combination with articles. See table 4.2 for a comparison of the English preposition "of" with the corresponding Italian combination of the preposition "di" with the pronouns of each number and gender. A similar table could also be drawn for the prepositions: "on", "in", "at", "from", and "with".

4.3.1 A Practical Example

To show how the English and the Italian grammar are related the composition of an English and an Italian grammar rule will be compared. Therefore, an English and an Italian natural language question which both lead to the same SPARQL query will be examined. First, the English question, which was also used as an example in section 3.2, will be discussed.

```
<START> ::= <SPO_OBJ> ?
            |SELECT <<SPO_OBJ>>
            |WHERE (<<SPO_OBJ>>)
            |<<SPO_OBJ>>

<SPO_OBJ> ::= list all <class>|?class|?class <<class>>

<class> ::= <NC>|<<NC>>|<rdf-syntax-ns#type> <<NC>>

<NC> ::= <NCc>|<<NCc>>|<<NCc>>
<NC> ::= <NCv>|<<NCv>>|<<NCv>>
```

Listing 4.3: Example of an English Ginseng Grammar Rule

The combination of the rules in listing 4.3 results in the English natural language question: "List all *classname*." Whereas *classname* stands for the name of a class defined in the an ontology; in this case a classname defined at *www.mooney.net/geo*. There, *classname* could be replaced, for example, by "cities", "states", or "lakes". Thus, *classname* is the only variable element in the sentence "List all *classname*.".

Below, the grammar rules for the composition of the Italian translation of the sentence "List all *classname*." are listed.

```
<START> ::= <SPO_OBJ> ?
            |SELECT <<SPO_OBJ>>
            |WHERE (<<SPO_OBJ>>)
            |<<SPO_OBJ>>
```

```
<SPO_OBJ> ::= elenca <tuttoPl> <articoloPl> <class>
              |?class |?class <<class>>

<NC> ::= <NCc>|<<NCc>>|<<NCc>>
<NC> ::= <NCv>|<<NCv>>|<<NCv>>

<tuttoPl>   ::= tutti
<tuttoPl>   ::= tutte

<articoloPl>  ::= i
<articoloPl>  ::= gli
<articoloPl>  ::= le

<class> ::= <NC>|<<NC>>|<rdf-syntax-ns#type> <<NC>>
```

Listing 4.4: Example of an Italian Ginseng Grammar Rule

The list of Italian rules for the discussed example is longer than the English one. As mentioned previously in this section, the Italian grammar needs more non-terminal symbols. In this example the Italian word for "all" ("tutti" or "tutte") and the definite plural article ("i", "gli" or "le") have to be adapted to the gender and number of the selected classname. Since the classname is selected last in the sentence, it is the users responsibility to choose the grammatically correct forms of words. Theoretically, the following set of sentences could result from the combination of the rules in listing 4.4:

- Elenca tutti i *classname*.
- Elenca tutti gli *classname*.
- Elenca tutti le *classname*.
- Elenca tutte i *classname*.
- Elenca tutte gli *classname*.
- Elenca tutte le *classname*.

Obviously, not all of these sentences are grammatically correct. The valid ones are only the following:

- Elenca tutti i *classname*.
- Elenca tutti gli *classname*.
- Elenca tutte le *classname*.

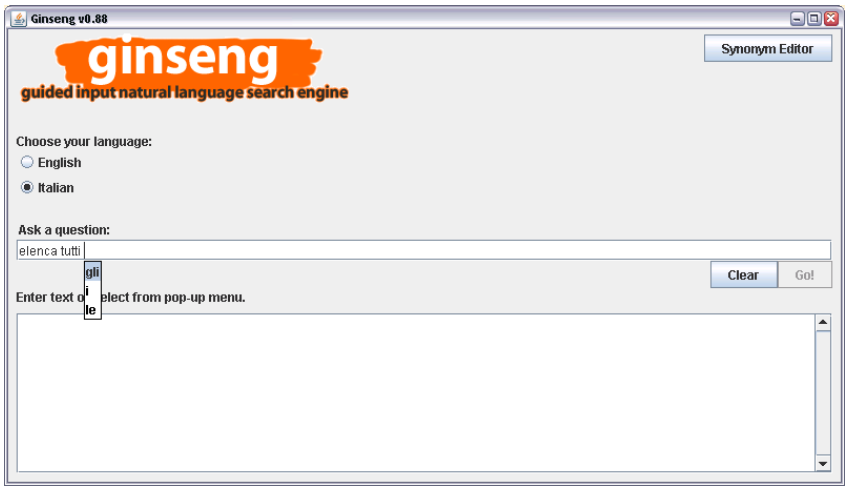

Figure 4.1: Choosing the Right Italian Article in the Ginseng User Interface

The user chooses the words he/she wants to use to construct a sentence on the Ginseng user interface with the help of a pop-up list. The situation where the user has to choose the right form of the Italian definite article is shown in figure 4.1.

In general, Ginseng does not "know" the grammatical gender and number of the words it uses. Thus, even if the word which determines the grammatical form of the other words in a sentence would appear before the words that needed to be adapted, Ginseng could not do the adaption autonomously. Ginseng does not store any grammatical information about its synonyms. However, if more languages in which some words take forms based on number and gender of other words are integrated, it might be helpful to include grammatical information in the definitions of Ginseng Phrases.

4.4 Limitations

The current implementation of Ginseng can not process ontologies which contain special characters, such as accents or umlaute. In Italian accents are special characters used to indicate the emphasized pronunciation of the last syllable of a word, as in "città" or "località".

The reason for this limitation is that the OWL files used in Ginseng use the *UTF-8* character set, which does not encode an international character set. In the header of a document based on XML a character set can be specified. If this specification is omitted UTF-8 is used as the default encoding. [Bray et al., 2006] This is the case in Ginseng. At the moment the header of the OWL file *www.mooney.net geo.owl* is:

```
<?xml version="1.0"?>
```

To support special characters this header should be changed to:

```
<?xml version="1.0" encoding="iso-8859-1"?>
```

This header specifies the character ISO-8859-1, also known as *Latin-1*. It encodes the special characters of most Western European languages, including Italian, French, and German. Another problem arises, though, if languages which need a character set different from Latin-1 are integrated into Ginseng, for example, Central European languages, such as Croatian and Polish. Therefore, it should be examined if the current implementation of Ginseng could support languages which need different character sets. In general, the issue of language encodings should be considered more in detail in the future, as it could constitute a barrier to multilingualism in Ginseng. First, though, it has to be decided which encodings to use and to what extent Ginseng should be a multilingual system. After all, if only Western European languages are integrated the issue of using several different character encodings at the same time is not relevant anymore.

In conclusion, it is important to mention that to keep the current system consistent it was decided not to use accents in the Italian grammar either. Thus, if a character encoding which supports accents is employed in the future the grammar should be updated to contain these symbols.

In this chapter the changes made to Ginseng to support the Italian language were presented. Besides the extension of the implementation of the tool itself, also the way an ontology has to be adapted to support an additional language was shown. Furthermore, the Italian grammar was introduced and its differences to the English grammar shown. So far Ginseng was extended to support English and Italian, but it can be extended by following the steps presented in this chapter to support further languages, as long as a grammar file can be defined for them. The next chapter introduces a new feature added to Ginseng for managing the Ginseng Phrases in the used ontologies.

5

The Ginseng Synonym Editor

If multilingual ontologies are used in Ginseng the number of synonyms increases with every added natural language. Listing 5.1 for example, shows the definition of the object property "isCapitalOf" for which 20 English and Italian Ginseng Phrases are specified.

```
<owl:ObjectProperty rdf:ID="isCapitalOf">
  <rdfs:domain rdf:resource="#Capital"/>
  <rdfs:range rdf:resource="#State"/>
  <ginseng:ignore rdf:value="id text"/>
  <ginseng:phrase xml:lang="en" rdf:value="is capital of"/>
  <ginseng:phrase xml:lang="en" rdf:value="of"/>
  <ginseng:phrase xml:lang="en" rdf:value="is the capital of"/>
  <ginseng:phrase xml:lang="en" rdf:value="is the capital city of"/>
  <ginseng:phrase xml:lang="en" rdf:value="are capitals of"/>
  <ginseng:phrase xml:lang="en" rdf:value="are the capitals of"/>
  <ginseng:phrase xml:lang="en" rdf:value="are capital cities of"/>
  <ginseng:phrase xml:lang="en" rdf:value="is OBJ located"/>
  <ginseng:phrase xml:lang="it" rdf:value="del"/>
  <ginseng:phrase xml:lang="it" rdf:value="dello"/>
  <ginseng:phrase xml:lang="it" rdf:value="della"/>
  <ginseng:phrase xml:lang="it" rdf:value="dell'"/>
  <ginseng:phrase xml:lang="it" rdf:value="e la capitale del"/>
  <ginseng:phrase xml:lang="it" rdf:value="e la capitale dello"/>
  <ginseng:phrase xml:lang="it" rdf:value="e la capitale della"/>
  <ginseng:phrase xml:lang="it" rdf:value="e la capitale dell'"/>
  <ginseng:phrase xml:lang="it" rdf:value="sono le capitali del"/>
  <ginseng:phrase xml:lang="it" rdf:value="sono le capitali dello"/>
  <ginseng:phrase xml:lang="it" rdf:value="sono le capitali della"/>
  <ginseng:phrase xml:lang="it" rdf:value="sono le capitali dell'"/>
</owl:ObjectProperty>
```

Listing 5.1: Definition of the Class "City" with Ginseng Phrases

If synonyms for additional languages had to be defined for the property shown in listing 5.1 the list of Ginseng Phrases would become even longer. Thus, with a growing number of supported languages the number of synonyms increases significantly. As a consequence ontology files become very large and hard to manage. This is why the *Synonym Editor* was developed, a user interface for managing Ginseng Phrases of ontologies in Ginseng. It provides the means for adding, editing, and removing synonyms in class and property definitions. At the same time, it offers a well-organized overview of the classes and properties and of their synonyms. Obviously, the editor is not meant for users who only need to query the knowledge bases but rather for users with more access rights and advanced knowledge about ontologies, such as administrators.

5.1 The Features of the Synonym Editor

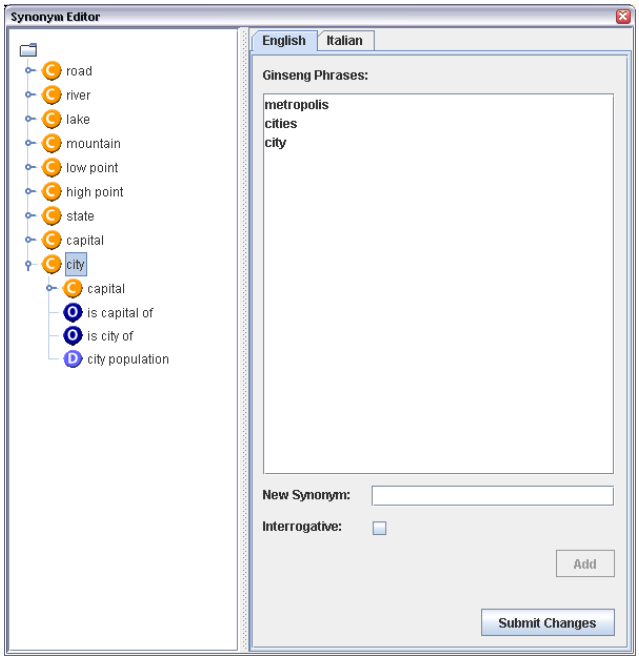

Figure 5.1: The Ginseng Synonym Editor

Figure 5.1 displays the user interface of the Synonym Editor. The left side of the editor window contains a tree, which visualizes the classes and properties of ontologies loaded in Ginseng, organized in a "typical way". The root of the tree is labeled with the URI prefix of the ontology it represents. In Figure 5.1 the URI prefix is *www.mooney.net/geo*. Attached to the root, the OWL classes are listed. The classes are parent nodes to their

subclasses and to the properties which have them as their domains.

When a class or a property is selected in the ontology tree, its Ginseng Phrases appear in the list of synonyms on the right-hand side of the window. Here, two tabs are visible one labeled "English", the other labeled "Italian". Each tab shows the synonyms in the corresponding language. Figure 5.1 shows the Synonym Editor with focus on the English tab, in figure 5.2 the focus is on the Italian tab. In this part of the window the synonyms can be edited. The following operations can be performed for a selected tree node: adding, deleting, and removing synonyms. These operations are described in detail below.

Add new synonyms: To add a new Ginseng Phrase to a class or property selected in the tree the user has to enter the desired synonym into the textfield *New Synonym* and check the *Interrogative* checkbox in case he/she is specifying an interrogative phrase. Then, by clicking the *Add* button, the new Ginseng Phrase is added to the list of synonyms. Figure 5.2 shows the Synonym Editor just before the Italian synonym "località" is added to the class "City". Beneath figure 5.2, listing 5.2 displays the definition of the class "City", before the synonym "località" is added to it.

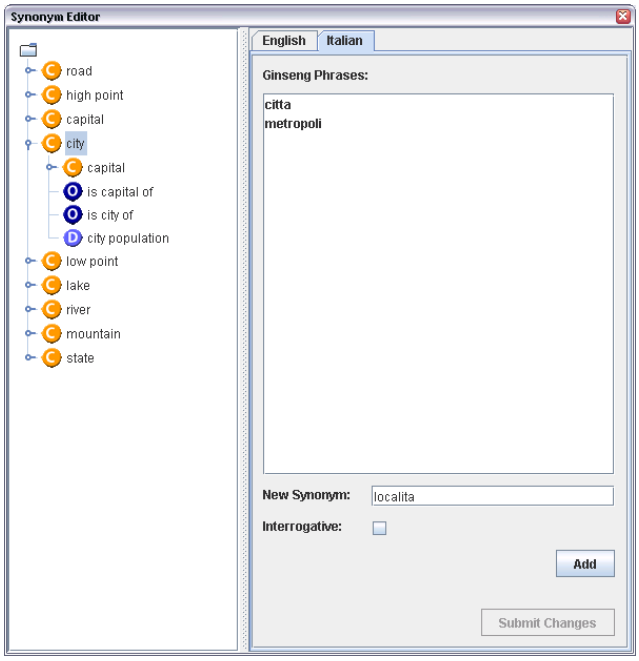

Figure 5.2: The Ginseng Synonym Editor Before Adding a New Synonym for the Class "City"

```
<owl:Class rdf:ID="City">
   <ginseng:ignore rdf:value="id text"/>
   <ginseng:phrase xml:lang="en" rdf:value="city"/>
   <ginseng:phrase xml:lang="en" rdf:value="cities"/>
   <ginseng:phrase xml:lang="en" rdf:value="metropolis"/>
   <ginseng:phrase xml:lang="it" rdf:value="citta"/>
   <ginseng:phrase xml:lang="it" rdf:value="metropoli"/>
</owl:Class>
```

Listing 5.2: Definition of the OWL Class "City"

After the new synonym is entered and the *Add* button clicked, the new synonym appears in the synonym list. In Figure 5.3 the synonym "località" is now listed among the Italian synonyms of the class "City". The effect this operation has on the OWL file holding the definition of that same class is shown in listing 5.3. Now, the synonym "località" is listed as a Ginseng Phrase in the class definition.

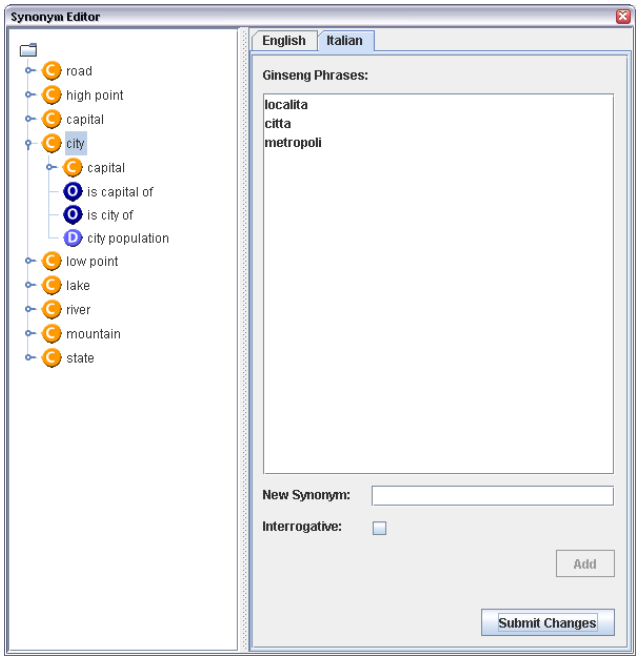

Figure 5.3: The Ginseng Synonym Editor After Adding a New Synonym to the Class "City"

```
<owl:Class rdf:ID="City">
   <ginseng:ignore rdf:value="id text"/>
   <ginseng:phrase xml:lang="en" rdf:value="city"/>
   <ginseng:phrase xml:lang="en" rdf:value="cities"/>
```

```
   <ginseng:phrase xml:lang="en" rdf:value="metropolis"/>
   <ginseng:phrase xml:lang="it" rdf:value="citta"/>
   <ginseng:phrase xml:lang="it" rdf:value="metropoli"/>
   <ginseng:phrase xml:lang="it" rdf:value="localita"/>
</owl:Class>
```

Listing 5.3: The OWL Class "City" with the Additional Synonym "località"

Edit existing synonyms: To edit a synonym the user needs to right-click on it in the synonym list. From the appearing pop-up menu the *Edit* entry has to be selected. A dialog window shows up, in which the chosen synonym can be modified. The figures 5.4 and 5.5 show the pop-up menu and the dialog for editing synonyms.

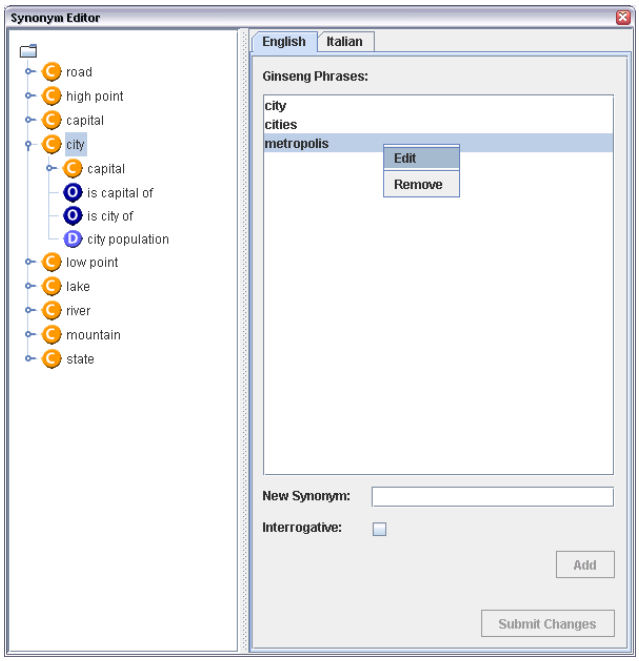

Figure 5.4: The Ginseng Synonym Editor Showing the Pop-up Menu

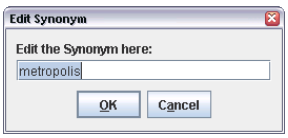

Figure 5.5: Edit Dialog of the Ginseng Synonym Editor

Remove existing synonyms: To delete a Ginseng Phrase from the synonym list the user needs to right-click on the synonym he/she wants to remove. From the appearing

pop-up menu the *Remove* entry has to be selected. The *Remove* menu item appears on the same pop-up menu as the *Edit* item. It is displayed in figure 5.4.

To save the changes made to ontologies in the Synonym Editor the *Submit Changes* button, located at the bottom of the window, is used. When this button is clicked, the changes made in the Synonym Editor are written to the OWL file which the ontology was loaded from at startup.

When the Synonym Editor is closed, the changes not yet saved in the OWL file are written to it. Then, all grammar rules stored at that point in the Thesaurus are cleared and new rules are generated from the grammar files and the edited OWL files. Thus, the user is immediately able to work with the edited knowledge base.

The Synonym Editor provides a tool for the maintenance of Ginseng synonyms. As it is not meant for any user, in the future two separate versions of Ginseng should exist. One, for administrators, with access to all the features of the application and one for restricted users, who will only be able to access the main Ginseng user interface and to query knowledge bases.

6

Evaluation

To measure the results achieved within this thesis a pilot evaluation was conducted. Therefore, an experiment was set up and given to five subjects. The purpose of this evaluation was to show how a bigger test series could be conducted with more subjects to test the quality of the newly added Italian grammar and the usability of the Synonym Editor.

6.1 The Test Data and the Users

As a knowledge base the file "www.mooney.net geo.owl", which contains facts about the geography of the United States, was used. The grammar tested was the Italian grammar for Ginseng, stored in the file "standard_italian.ggf".

Five people took part in this experiment, four women and one man. All of them had some experience using computers, two of them had professional knowledge. They were all native Italian speakers. None of the five subjects knew Ginseng before the evaluation.

6.2 The Experiment

The Experiment had two parts, in the first, the quality of the Italian grammar was tested, in the second, the user-friendliness of the Synonym Editor. All the written material given to the users during the experiment, such as instructions and questionnaires, can be found in appendix B.

The subjects tested the system independently from each other having only a set of written instructions and a computer running Ginseng to their disposal. They had to solve their tasks alone, observed by a person sitting next to them.

Before getting to their actual assignments, the test users had to read a description of the Ginseng system. Then, to make themselves familiar with the system, they had to enter two given questions into the Ginseng user interface, one in English and one in Italian.

6.2.1 Testing the Italian Grammar

After getting to know the query interface the users were informed about the content of the knowledge base they were querying with Ginseng. Their first task was then, to freely formulate five questions they would like to ask the system about the geography of the United States in Italian and to enter them into a text editor. They did not have to enter the questions into the system themselves. Afterward, we entered the queries into the system to see how many of the questions formulated by the users could be processed by Ginseng.

The Results

In table 6.1 the questions the users formulated are listed. In the third column the value "yes" designates that the question could be entered into Ginseng and that the answer returned for that question was correct. "No" means, Ginseng was not able to process the question. The ratio in the last column shows how many of the five questions suggested by the respective user were answered by Ginseng. The total ratio, over all the 25 questions is 9/25, which corresponds to 36.0%. Those questions can be denoted as *semantically tractable* according to the definition in [Popescu et al., 2003].

For the questions that were semantically tractable, we also computed recall[1] and precision[2], as defined in [Baeza-Yates and Ribeiro-Neto, 1999], by checking for how many of them Ginseng returned correct results. Recall as well as precision were 100%. That means, all the questions that had resulted semantically tractable were answered correctly by Ginseng.

6.2.2 Testing the Synonym Editor

First, the users had to read a description of the functionality of the Synonym Editor. Then, to test the Synonym Editor, they were asked to select a given class in the ontology tree

[1] "Recall is the fraction of the relevant documents [...] which has been retrieved[...]." [Baeza-Yates and Ribeiro-Neto, 1999]

[2] "Precision is the fraction of the retrieved documents [...] which is relevant." [Baeza-Yates and Ribeiro-Neto, 1999]

Subject	Question	Successful	Ratio
1	Qual'è la montagna più alta?	no	1/5
	Quanto è grande la California?	yes	
	In quale parte dell'America si trova il monte Alverstone?	no	
	Qual'è lo stato più grande?	no	
	Qual'è lo stato più montuoso?	no	
2	Quanti fiumi ci sono negli Stati Uniti?	yes	2/5
	Quanto è alta la montagna più alta degli Stati Uniti?	no	
	Dove si trova il fiume Mississippi?	no	
	Quanto è grande il lago Michigan?	yes	
	Quante città ha lo stato di New York?	no	
3	Dov'è situato il monte Alverstone?	no	1/5
	Da dove nasce il Mississippi?	no	
	Quale città bagna il Mississippi?	no	
	In quale stato è situato il monte Alverstone?	no	
	Quanti stati ci sono in America?	yes	
4	Qual'è il fiume più lungo d'America?	no	3/5
	Quanti abitanti ha la città di New York?	yes	
	Qual'è la capitale del Minnesota?	yes	
	Quanti fiumi ci sono in America?	yes	
	Come si chiama lo stato più a nord?	no	
5	Qual'è il punto più alto degli Stati Uniti?	no	2/5
	Quanti laghi ci sono nello stato del Texas?	no	
	C'è una citt con più di due millioni di abitanti nello stato dell'Ohio?	no	
	Quanti stati confinano con il Colorado?	yes	
	Quali sono gli stati che confinano con la Virginia?	yes	

Table 6.1: Questions Users Would Like to Ask Ginseng

Subject	1	2	3	4	5	average
SUS Score	72.5	90	87.5	82.5	80	82.5

Table 6.2: SUS Scores for the Ginseng Synonym Editor

and to perform the following operations on them:

- add a new synonym
- edit an existing synonym
- submit the changes to the OWL file

After making these changes, the test users were asked to close the Synonym Editor and to enter a predefined query, containing a previously edited synonym, into the Ginseng query interface. To measure the user experience the subjects finally compiled a usability questionnaire [Brooke, 1996]. The collected results are presented and evaluated in the next section.

The Results

To evaluate the usability of the Synonym Editor we used the *System Usability Scale* developed by [Brooke, 1996]. The users had each answered 10 questions about the usability of the system (see page 50). For those questions we computed the final SUS scores of the single users. These scores are listed in table 6.2. It can be seen that the scores are all rather high, the average score over all the users is 82.5. This means, the subjects who tested the Synonym Editor did not have trouble using it and thought it was quite user friendly. It must be said, though, that two of these people have professional knowledge of computers. However, this is acceptable, since the editor is intended for administrators. Nevertheless, if a bigger evaluation of the system is made, for it to be representative, users with different levels of expertise with computers should be chosen.

6.3 Conclusions

In the first part of the evaluation the quality of the Italian grammar was tested by asking the users to freely formulate Italian questions. The acceptance of this questions in Ginseng was then examined. We found that only 36.0% of these questions were semantically tractable. The reason for this is that on the one hand some sentence structures are missing in the grammar and on the other hand that not all required synonyms are defined

in the ontology which was queried during the evaluation. Hence, to improve the usability of the system not only more rules need to be added to the Italian grammar, but also additional synonyms should be specified in used ontologies.

For the semantically tractable questions recall and precision, which both turned out to be 100%, were computed. This means, Ginseng processed and answered all the accepted questions correctly. Regarding the tested Italian grammar this implies that the grammar rules used by Ginseng to process those questions are formally correct. Thus, it can be concluded that the Italian grammar is not complete, however formally correct.

In the second part of the experiment the usability of the Synonym Editor was examined. Therefore, the subjects were asked to edit the synonyms of a given class with the help of the Synonym Editor. Thereafter they had to fill out a questionnaire for retrieving their SUS Scores for the Synonym Editor. The resulting average SUS score was 82.5, which is a rather high value. This value represents a high usability of the Synonym Editor. Also the observations made while the users were working with the editor showed that the subjects did not have any problems using the interface and easily understood how to perform the required operations after only a short introduction to Ginseng. Thus, from this evaluation the Synonym Editor results to be user friendly without major flaws.

This evaluation shows that especially Ginseng's functionality for querying OWL knowledge bases could be improved. This could be achieved by defining more grammar rules and by adding more synonyms to its ontologies. More ways to extend and enhance Ginseng will be suggested in the following chapter.

7

Future Work

7.1 Answer Generation

One enhancement of Ginseng could be to present natural language answers to the user instead of plain numbers or names, as it is the case now. This task should originally have been solved within this project with the help of the software NaturalOWL [Androutsopoulos and Galanis, 2007]. However, after a closer examination of Ginseng and NaturalOWL, it was clear that NaturalOWL would rather be suitable for other purposes, such as showing descriptions of OWL classes or properties on demand.

Like the ontologies used in Ginseng, the ones used in NaturalOWL store natural language information. NaturalOWL does not use simple expressions, like Ginseng does with the Ginseng Phrases, but it stores complete descriptions of the classes or properties in an ontology for one or more languages.

NaturalOWL is not suited for the answer generation in Ginseng, which needs to answer questions like: "How many states are there in the United States?", because the user does not wish to see a description of the United States but he/she only needs to know the number of states. In this case an appropriate answer would be: "There are 50 states in the United States." Assuming Ginseng will provide natural language answers in the future, NaturalOWL could be used to generate descriptions out of a Ginseng answer on demand. For example, in the sentence "There are 50 states in the United States." the user could click on the expression "United States" to read a description of the United States generated by NaturalOWL.

It must be said, though, that with the integration of NaturalOWL into Ginseng the maintenance effort for ontologies would increase. Because in that case, not only the Ginseng Phrases would have to be maintained but also the linguistic elements for Natu-

ralOWL. Furthermore, some redundancy in the stored lexical information would be introduced. For example, for the class "City" the English Ginseng synonyms are: "state" and "states", the Italian ones are: "stato" and "stati". For NaturalOWL the following information could be added: "*instancename* is a state." and "L'*instancename* è uno stato.". In general, it can be said that there is a trade-off between additional information and maintenance effort. This is why before integrating NaturalOWL into Ginseng, the advantages of the resulting application should be considered in relation to the effort needed to add and maintain the extra data.

7.2 Enhancements to the Ginseng User Interface

The Ginseng user interface, consisting of the Ginseng main window and the Synonym Editor already allow the user to query OWL knowledge bases and to edit the Ginseng Phrases of an ontology in an easy way. However, to improve the user experience the user interface could be adapted to the natural language chosen by the user for querying. Most likely, a user who enters queries in Italian also likes seeing the labels and messages that appear on the interface in Italian. This addition would make the user interface more consistent when a language is chosen for the queries.

A possible way to implement this extension in Java would be to create property classes for the supported languages storing all the labels and messages that are supposed to appear on the corresponding interface.

This feature would not add any new functionality to the interface, but it could improve the appearance of Ginseng and the user experience.

7.3 Extension of the Multilingualism

With this thesis the proof was given that Ginseng can support multiple natural languages, as long as a static grammar can be defined for them. At the moment English and Italian are part of the Ginseng system, but it would be interesting to add more languages. It should probably be possible to add French and Spanish easily. Like Italian, they both are Latin languages, which makes their grammatical structure similar. For other languages, like German for example, a more extensive examination is required to check if and how those languages can be integrated into the system.

8

Conclusions

In this thesis Ginseng was introduced and shown that the system can support the languages English and Italian at the same time, using the same concepts and how this can be done by combining static grammars with dynamic vocabulary extracted from ontologies. Moreover, the implementation of the Synonym Editor, an interface for managing semantic annotations of ontologies in Ginseng, was presented. The multilingual system as well as the Synonym Editor were evaluated and suggestions for future work were made. In the following, some interesting points on the multilingual Ginseng system will be discussed.

In the latest version Ginseng includes an English as well as an Italian query interface. In theory, Ginseng can support an infinite number of languages, as long as a static grammar and Ginseng Phrases can be defined for them. Thus, for each language that has to be integrated into Ginseng it must be examined if, with a static grammar and Ginseng Phrases, meaningful questions can dynamically be composed. Essentially, questions that can be used to query knowledge bases through Ginseng. From the similarities in the structures of the Italian, French, and Spanish grammars it can be assumed that it is possible to add French and Spanish to Ginseng too. Regarding the issue on character encoding addressed in chapter 4.4 French and Spanish, which both use special characters, require the same character set as Italian. Thus, the same limitations apply for Italian, French, and Spanish.

Ginseng could support an infinite number of languages - meaning natural languages - but with every added language the number of Ginseng Phrases listed in the included ontologies grows. As a consequence, it gets harder to manage the synonyms of these ontologies in a regular text editor. This is why the Synonym Editor, an interface for managing Ginseng synonyms, was developed. With the support of multiple languages the

Synonym Editor has become an essential feature. Not only because of the increasing amount of synonyms, but also because in the multilingual Ginseng system the semantic annotation of ontologies is not optional anymore: While in the monolingual Ginseng system the class and property identifiers can be used as vocabulary items, in the multilingual Ginseng implementation this is not allowed. In the multilingual case vocabulary items are defined by Ginseng Phrases only and the *ignore* property is used to keep the resource name from appearing in any generated rule. Thus, as soon as more than one language is used in Ginseng, semantic annotations are required. As a solution to the problem of multilingual ontologies it was also suggested to have a knowledge base for each supported language. However, this would not be the correct approach, since this would cause redundant information to be stored in different documents, which could lead to inconsistencies and errors.

Today most knowledge bases are written in English. To test Ginseng with Italian knowledge bases we looked for some on the Internet, but we could not find any useful data. Furthermore, even in projects of Italian research groups knowledge bases are usually defined in English. In fact, the language in which classes and properties are named should not be relevant when using ontologies with NLIs. Natural language information for NLIs should be stored in form of additional semantic information in ontologies, like it is done with the Ginseng Phrases. As suggested in [Androutsopoulos et al., 2005], for example, OWL would become more powerful if standards for the specification of natural language information in OWL ontologies were defined. That way, maybe, semantic annotations would become more common. Furthermore, knowledge bases would already contain at least a part of the needed linguistic information, which would make the process of annotating a knowledge base shorter. In various projects, such as *ORAKEL* [Ishikawa et al., 1986] and *NaturalOWL* [Androutsopoulos and Galanis, 2007] natural language is combined with ontologies and in each of these project the linguistic information is added in a different way. Although the projects are different from each other, some redundant work is being done. With the definition of a standard this redundancy could be minimized. To facilitate and encourage the addition of natural language information to ontologies, tools, like the Synonym Editor, but independent from any other application, should be developed.

In the introduction of this thesis two goals were defined: The first goal was to extend Ginseng to support Italian as a query language. This was achieved by modifying the the implementation of Ginseng, adding Italian synonyms to ontologies and by developing an Italian grammar. Not only was the goal achieved, but it was also shown how more languages can be integrated into the system, although character encoding issues might

need to be addressed.

The second goal was to develop an interface for managing Ginseng Phrases in Ginseng. This task was solved by adding a new window, which provides the features for adding, removing, and editing synonyms, to the Ginseng user interface. In the evaluation the Synonym Editor was rated as user-friendly. Furthermore, with the introduction of multilingualism in Ginseng the Synonym Editor has become an essential tool, which helps managing the Ginseng Phrases.

This thesis has shown that the concepts developed in Ginseng can be applied for a multilingual implementation too. It would certainly be interesting, in the future, to add more languages to the system to find if there are more language-dependent limitations to the multilingualism in Ginseng.

A

The Italian Grammar

```
## Ginseng General Grammar Description File for Italian

<ginseng:lang="it">

RDQL <START> ::= <SPO_CLOSED> ?|SELECT <<SPO_CLOSED>>|WHERE (<<
   SPO_CLOSED>>) |<<SPO_CLOSED>>
RDQL <START> ::= <SPO_ENUM> ?|SELECT <<SPO_ENUM>>|WHERE (<<
   SPO_ENUM>>) |<<SPO_ENUM>>
RDQL <START> ::= <SPO_OBJ> ?|SELECT <<SPO_OBJ>>|WHERE (<<SPO_OBJ
   >>) |<<SPO_OBJ>>

<SPO_CLOSED> ::= [<articolo_sing>] <NI> <VIU> |?what|?what <<VIU
   >>|AND (?what eq <<NI>>)

<SPO_OBJ> ::= elenca <tutti> <articolo_pl> <class>|?class|?class
    <<class>>
<SPO_OBJ> ::= elenca <articolo_pl> <class> che <V_O>|<<class
   >>|<<class:1>> <<class>>) (<<class:1>> <<V_O>>|<<class>>

<SPO_OBJ> ::= dove <V_O>|?where|?where <<V_O>>

<SPO_OBJ> ::= cos' <articolo_indet> <NC>|?class|?class <http://
   www.w3.org/2000/01/rdf-schema#subClassOf> <<NC>>
<SPO_OBJ>  ::= cos' <articolo_sing> <NI>|?inst|<<NI>> <http://
   www.w3.org/1999/02/22-rdf-syntax-ns#type> ?inst

################################################
```

```
<SPO_OBJ> ::= quale <class> c'|<<class>>|<<class:1>> <<class
   >>|<<class>>
<SPO_OBJ> ::= quali <class> ci sono|<<class>>|<<class:1>> <<
   class>>|<<class>>
<SPO_OBJ> ::= cosa <V_O>|?what|?what <<V_O>>
<SPO_OBJ> ::= che cosa <V_O>|?what|?what <<V_O>>
<SPO_OBJ> ::= cosa <V_O> and <V_O2>|?what|?what <<V_O>>) (?what
   <<V_O2>>
<SPO_OBJ> ::= che cosa <V_O> and <V_O2>|?what|?what <<V_O>>) (?
   what <<V_O2>>

<SPO_OBJ> ::= quali sono <articolo_pl> <class> <
   prep_articolata_di> <instance>|<<class>>|<<class:1>> <<
   instance>>) (<<class:1>> <<class>>|<<class>>
<SPO_OBJ> ::= quali sono <articolo_pl> <class> <V_O>|<<class
   >>|<<class:1>> <<class>>) (<<class:1>> <<V_O>>|<<class>>
<SPO_OBJ> ::= come si chiamano <articolo_pl> <class> <V_O>|<<
   class>>|<<class:1>> <<class>>) (<<class:1>> <<V_O>>|<<class>>
<SPO_OBJ> ::= come si chiama <articolo_sing> <class> <V_O>|<<
   class>>|<<class:1>> <<class>>) (<<class:1>> <<V_O>>|<<class>>
<SPO_OBJ> ::= elenca <tutti> <articolo_pl> <class> <V_O>|<<class
   >>|<<class:1>> <<class>>) (<<class:1>> <<V_O>>|<<class>>

<SPO_OBJ> ::= quale <class> ha <articolo_indet> <Measure>|<<
   class>>|?measure <<Measure>> <<class:1>>) (<<class:1>> <<
   class>>|<<class>>
<SPO_OBJ> ::= quale <class> ha <articolo_indet> <Measure> di <
   Subj>|<<class>>|<<Subj>> <<Measure>> <<class:1>>) (<<class
   :1>> <<class>>|<<class>>

<SPO_OBJ> ::= quale <class> <V_O>|<<class>>|<<class:1>> <<class
   >>) (<<class:1>> <<V_O>>|<<class>>
<SPO_OBJ> ::= quali <class> <V_O>|<<class>>|<<class:1>> <<class
   >>) (<<class:1>> <<V_O>>|<<class>>
<SPO_OBJ> ::= quale <class> <V_O> e <V_O2>|<<class>>|<<class:1>>
    <<class>>) (<<class:1>> <<V_O>>) (<<class:1>> <<V_O2>>|<<
   class>>
<SPO_OBJ> ::= qual' <articolo_sing> <Measure> <
   prep_articolata_di> <Subj>|?what|<<Subj>> <<Measure>> ?what
<SPO_OBJ> ::= quali sono <articolo_pl> <Measure> <
   prep_articolata_di> <Subj>|?what|<<Subj>> <<Measure>> ?what
<SPO_OBJ> ::= qual' <articolo_sing> <Measure> <
   prep_articolata_di> <Subj>|?what|<<Subj>> <<Measure>> ?what
```

```
<SPO_OBJ> ::= <MInt> <VIU>|?what,<<MInt>>|?what <<VIU>>) (?what
   <<MInt>>
<SPO_OBJ> ::= <MInt> <articolo_sing> <class> <instance>|<<MInt
   >>|?instance <<instance>>) (?instance <<class>>) (?instance
   <<MInt>>
<SPO_OBJ> ::= <MInt> <articolo_sing> <class> <prep_articolata_di
   > <instance>|<<MInt>>|?instance <<instance>>) (?instance <<
   class>>) (?instance <<MInt>>
<SPO_OBJ> ::= <MInt> <articolo_sing> <instance>|<<MInt>>|?
   instance <<instance>>) (?instance <<MInt>>
################################################

<SPO_OBJ> ::= <Measure> <prep_articolata_di> <Subj>|?what|<<Subj
   >> <<Measure>> ?what
<SPO_OBJ>  ::= <Measure> <prep_articolata_di> <class> <Subj>|<<
   class>>|<<Subj>> <<Measure>> <<class:1>>) (<<class:1>> <<
   class>>|<<class>>
<SPO_OBJ>  ::= <Measure> <prep_articolata_di> <Subj> <class>|<<
   class>>|<<Subj>> <<Measure>> <<class:1>>) (<<class:1>> <<
   class>>|<<class>>

<SPO_ENUM>  ::= <pron_interrog_quant> <instance> ci sono|?
   instance|?instance <<instance>>
<SPO_ENUM>  ::= <pron_interrog_quant> <class> <V_O>|<<class>>|<<
   class:1>> <<class>>) (<<class:1>> <<V_O>>|<<class>>
<SPO_ENUM>  ::= <pron_interrog_quant> <class> <V_O> e <V_O2>|<<
   class>>|<<class:1>> <<class>>) (<<class:1>> <<V_O>>) (<<class
   :1>> <<V_O2>>|<<class>>
<SPO_ENUM>  ::= <pron_interrog_quant> <class> ci sono|<<class
   >>|<<class:1>> <<class>>|<<class>>

<instance>  ::= <NI>|?instance|<http://www.w3.org/2002/07/owl#
   sameAs> <<NI>>

<class> ::= <NC>|<<NC>>|<http://www.w3.org/1999/02/22-rdf-syntax
   -ns#type> <<NC>>
<class> ::= <adj> <NC>|<<NC>>|<http://www.w3.org/1999/02/22-rdf-
   syntax-ns#type> <<NC>>) (<<NC:1>> <<adj>>|<<adj>>

## phrase|variable|subject
<NC> ::= <NCc>|<<NCc>>|<<NCc>>
<NC> ::= <NCv>|<<NCv>>|<<NCv>>

<Subj> ::= <NI>|-|<<NI>>
```

```
<Subj> ::= <instance>|?objinst|<<instance:1>> <<instance>>) (<<
   instance:1>>
<Subj> ::= <articolo> <class> <prep_articolata_di> <instance>|?
   objinst|<<instance:1>> <<class>>) (<<instance:1>> <<instance
   >>) (<<instance:1>>
<Subj> ::= <articolo> <instance> <class>|?objinst|<<instance:1>>
    <<class>>) (<<instance:1>> <<instance>>) (<<instance:1>>
<Subj> ::= <NU>|-|?anything
<Subj> ::= <articolo> <class> che <V_O>|<<class>>|<<class:1>> <<
   class>>) (<<class:1>> <<V_O>>) (<<class:1>>

## phrase|nothing|object
<Obj> ::= <class> <V_O>|<<class>>|<<class:1>>) (<<class:1>> <<
   V_O>>|<<class>>
<Obj> ::= <instance>|?objinst|<<instance:1>>) (<<instance:1>> <<
   instance>>
<Obj> ::= <articolo> <class> <prep_articolata_di> <instance>|?
   objinst|<<instance:1>>) (<<instance:1>> <<class>>) (<<
   instance:1>> <<instance>>
<Obj> ::= <articolo> <instance>|?objinst|<<instance:1>>) (<<
   instance:1>> <<instance>>
<Obj> ::= <NU>|-|?anything
<Obj> ::= <articolo> <class> che <V_O>|<<class>>|<<class:1>>)
   (<<class:1>> <<class>>) (<<class:1>> <<V_O>>
<Obj> ::= <articolo> <class> <V_O>|<<class>>|<<class:1>>) (<<
   class:1>> <<class>>) (<<class:1>> <<V_O>>


## phrase|nothing|predicate+object
<V_O> ::= <VCO>|-|<<VCO>>
<V_O> ::= <VCU>|-|<<VCU>>
<V_O> ::= e <VIO>|-|<<VIO>>
<V_O> ::= e <VIU>|-|<<VIU>>
<V_O> ::= ha <VHO>|-|<<VHO>>
<V_O> ::= ha <VHU>|-|<<VHU>>
<V_O> ::= sono <VIO>|-|<<VIO>>
<V_O> ::= sono <VIU>|-|<<VIU>>
<V_O> ::= hanno <VHO>|-|<<VHO>>
<V_O> ::= hanno <VHU>|-|<<VHU>>

<V_O2> ::= <V_O>|-|<<V_O>>

## phrase|nothing|nothing
<NU> ::= chiunque|?anyone|?anyone
```

```
<NU> ::= qualsiasi cosa|?anything|?anything
<NU> ::= ovunque|?anywhere|?anywhere

## articoli e preposizioni
<prep_articolata_di> ::= di
<prep_articolata_di> ::= del
<prep_articolata_di> ::= dell’
<prep_articolata_di> ::= dello
<prep_articolata_di> ::= della
<prep_articolata_di> ::= dei
<prep_articolata_di> ::= degli
<prep_articolata_di> ::= delle

<prep_articolata_in> ::= in
<prep_articolata_in> ::= nel
<prep_articolata_in> ::= nell’
<prep_articolata_in> ::= nello
<prep_articolata_in> ::= nella
<prep_articolata_in> ::= nei
<prep_articolata_in> ::= negli
<prep_articolata_in> ::= nelle

<prep_articolata_su> ::= su
<prep_articolata_su> ::= sul
<prep_articolata_su> ::= sull’
<prep_articolata_su> ::= sullo
<prep_articolata_su> ::= sulla
<prep_articolata_su> ::= sui
<prep_articolata_su> ::= sugli
<prep_articolata_su> ::= sulle

##articoli
<articolo>   ::= <articolo_sing>
<articolo>   ::= <articolo_indet>

<articolo_sing>  ::= il
<articolo_sing>  ::= l’
<articolo_sing>  ::= lo
<articolo_sing>  ::= la

<articolo_pl>  ::= i
<articolo_pl>  ::= gli
<articolo_pl>  ::= le
```

```
<articolo_indet>  ::= un
<articolo_indet>  ::= un'
<articolo_indet>  ::= uno
<articolo_indet>  ::= una

##pronomi interrogativi qualitativi
<pron_interrog_qual> ::= cosa
<pron_interrog_qual> ::= quale
<pron_interrog_qual> ::= qual'
<pron_interrog_qual> ::= quali

##pronomi interrogativi quantitativi
<pron_interrog_quant> ::= quanti
<pron_interrog_quant> ::= quante

<tutti>   ::= tutti
<tutti>   ::= tutte
```

Listing A.1: The Complete Italian Grammar for Ginseng

B

Evaluation Material

This appendix contains all the instructions, task description and questionnaires used for the evaluation of Ginseng presented in chapter 6. Following documents are displayed:

Pages 50 to 53: These pages show the instructions and the descriptions of the tasks assigned to the users.

Page 49: System Usability Scale

Page 54: Questionnaire for retrieving personal information from users

Since the evaluation was conducted in German, all the questionnaires appear to be written in German.

SUS – Fragebogen zu Benutzerfreundlichkeit und Verwendbarkeit

Kreuzen Sie bitte Zutreffendes an. Bitte **nur 1 Kreuz** pro Frage.

		vollkommen einverstanden	einverstanden	weder noch	nicht einverstanden	ganz und gar nicht einverstanden
1.	Ich denke, dass ich das System und dessen Abfragesprache gerne öfters benutzen würde.	☐	☐	☐	☐	☐
2.	Ich finde, dass das System und dessen Abfragesprache unnötig kompliziert waren.	☐	☐	☐	☐	☐
3.	Ich denke, dass das System und dessen Abfragesprache einfach zu benutzen waren.	☐	☐	☐	☐	☐
4.	Ich denke, dass ich die Hilfe einer Fachperson benötigen würde, um das System und dessen Abfragesprache benützen zu können.	☐	☐	☐	☐	☐
5.	Ich finde, dass die verschiedenen Aufgaben des Systems und der Abfragesprache gut integriert sind.	☐	☐	☐	☐	☐
6.	Ich denke, dass es zu viele Widersprüchlichkeiten in diesem System und der Abfragesprache gibt.	☐	☐	☐	☐	☐
7.	Ich könnte mir vorstellen, dass die meisten Leute sehr schnell lernen würden, mit dem System und der Abfragesprache umzugehen.	☐	☐	☐	☐	☐
8.	Ich finde, dass das System und die Abfragesprache sehr mühsam waren.	☐	☐	☐	☐	☐
9.	Ich fühlte mich sehr selbstsicher bei der Benutzung des Systems und der Abfragesprache.	☐	☐	☐	☐	☐
10.	Ich musste viele Dinge lernen, bevor ich mit dem System und der Abfragesprache umgehen konnte.	☐	☐	☐	☐	☐

University of Zurich
Department of Informatics

Dynamic and Distributed
Information Systems

Untersuchung des natürlichsprachlichen Suchsystems Ginseng

Ihre Aufgaben

Zunächst werden Sie sich fünf Fragen überlegen, die Sie an das System stellen möchten. Danach werden Sie selbst eine Frage in Ginseng eingeben, um mit dem System vertraut zu werden.
Im zweiten Teil des Experiments werden Sie über die Benutzerschnittstelle die Datenbasis des Systems mit vorgegebenen Elementen erweitern.
Sämtliche Instruktionen sind auf diesen Blättern vorhanden; die Fragen können Sie am Computer eingeben. Lesen Sie zuerst die Einführung in die Benutzung des Systems.

Am Schluss füllen Sie einen **Fragebogen mit 10 Fragen** aus, mit welchem Sie die Nützlichkeit und die Benutzerfreundlichkeit des Systems und dessen Abfragesprache beurteilen können.

Ganz am Schluss des Experiments bitten wir Sie noch, **7 Fragen** zur Ihrer Person zu beantworten.

Bitte beachten Sie, dass wir beim Experiment **nicht** Ihr Können testen, sondern dass Sie für uns das Suchsystem beurteilen. Wir untersuchen **tatsächlich** die Benutzerfreundlichkeit des Suchsystems und nicht Ihr Verhalten.

Das Suchsystem: Ginseng

Bitte lesen Sie jetzt diese Seite als Einführung zum Suchsystem.

Sie sehen nun auf dem Bildschirm vor Ihnen das Suchsystem Ginseng. Der Name steht für *Guided Input Natural Language Search Engine*. Dieses Suchsystem beantwortet Fragen zur Geografie der USA auf Englisch und auf Italienisch.

Bei diesem Suchsystem können **nur Fragen eingegeben werden, die vom System vorgeschlagen werden und die den Wörtern folgen, die in den aufgezeigten Listen stehen**. Die Listen ermöglichen Ihnen, ganze Sätze zu bilden. Wenn Sie im weissen Textfeld nach „Ask a question“ zu tippen beginnen, werden Ihnen die Listen mit Wörtern angezeigt, die Sie eingeben können. Tippen Sie beispielsweise ein „w“ ein, dann erscheinen alle mit „w“ beginnenden möglichen Wörter. Je mehr Buchstaben Sie pro Wort tippen, desto mehr öffnet sich die Liste mit möglichen Wörtern oder schränkt sich die Liste ein.

Sie können **ein Wort vollständig eintippen** oder **mit der Maus auf ein Wort in der Liste klicken** oder **mit der Pfeiltaste nach unten zu dem Wort gehen**, das Sie eingeben möchten, und die **Eingabetaste (Enter) drücken**.

Ist ein Wort eingegeben, werden Ihnen die darauffolgenden erlaubten Wörter angezeigt. Sie können erneut ein Wort auswählen. Das System führt Sie so durch die Formulierung einer Frage hindurch.

Taucht ein **Fragezeichen** in der Liste auf, so kann damit ein Satz abgeschlossen werden. Nach dem Auswählen des Fragezeichens wird eine Frage vom System beantwortet und die Antworten im unteren weissen Feld angezeigt.

Mit der Zurück-Taste (*Backspace*) können Sie im Satz zurückgehen und Wörter wieder löschen, um den Satz beispielsweise anders zu formulieren.

Abfragen machen mit Ginseng

Damit Sie sich mit dem Suchsystem Ginseng vertraut machen können, bitten wir Sie, zuerst eine Frage als Probedurchgang an das System zu stellen. Formulieren Sie dazu die folgende Frage in die Abfragesprache des Systems um, und lassen Sie die Frage vom System beantworten, indem Sie am Schluss Ihrer Frage das Fragezeichen auswählen:

What is the capital of Arizona?

Wechseln Sie nun zur italienischen Sprache, indem Sie das Feld „**Italian**“ aktivieren und geben Sie dieselbe Frage auf Italienisch ein:

Quanti abitanti ha la California?

1. Aufgabe

Das System enthält Informationen über die amerikanische Geografie. Die Inhalte decken Elemente wie Städte, Staaten, Flüsse, Berge, Seen und Strassen ab, sowie deren Eigenschaften wie z.B. die Breite der Flüsse oder die Höhe der Berge. Zudem sind auch die Beziehungen zwischen den Elementen gespeichert, beispielsweise „Die Stadt Los Angeles befindet sich im Staat Kalifornien.“

Formulieren Sie nun fünf Fragen, die Sie an das Suchsystem stellen möchten und schreiben Sie diese in den Texteditor. Sie müssen die Fragen nicht in Ginseng eingeben.

2. Aufgabe

In dieser Aufgabe werden Sie den Synonymeditor testen. Gehen Sie dabei wie folgt vor:

- Klicken Sie im Hauptfenster von Ginseng oben rechts auf den Knopf „Synonym Editor“, um den Editor zu starten.
- Klicken Sie auf den Baum links im Fenster. Klicken Sie dabei auf verschiedene Begriffe und machen Sie sich mit dem Baum vertraut.
- Auf der rechten Seite des Editors erscheinen die Synonyme für das Element, das jeweils im Baum selektiert ist.
- Die rechte Seite des Synonymeditors enthält zwei Tabs, je eins für die englische und eins für die italienische Sprache. Jedes Tab enthält die Liste der Synonyme in der entsprechenden Sprache.
- Der Editor erlaubt es, über das Textfeld „New Synonym“ und dem „Add“-Knopf Synonyme hinzuzufügen. Beim Rechtsklicken auf en Synonym öffnet sich das Menu, um Synonyme zu editieren oder zu entfernen.
- Fügen Sie nun zum Element **city population** die Synonyme **citizens** für das Englische und **cittadini** für das Italienische hinzu.
- Korrigieren Sie nun für das Element **lake** das Synonym **lagho** zu **lago**.
- Klicken Sie auf die Schaltfläche **Submit Changes,** um die vorgenommenen Änderungen dauerhaft zu speichern.
- Schliessen Sie den Synonymeditor.
- Geben Sie nun die italienische Frage aus Aufgabe 1 mit dem neuen Synonym ein:
  Quanti cittadini ha la California?

Persönliche Angaben

Bitte geben Sie uns zum Abschluss noch die folgenden Angaben zu Ihrer Person.

Die Informationen werden für rein statistische Zwecke und nicht für individuelle Schlussfolgerungen benutzt.

1. Alter: ☐ Jahre

2. Geschlecht: ☐ weiblich ☐ männlich

3. Beruf: ☐

		professionell	gut	mittel	schlecht	gar keine
4.	Meine Kenntnisse in Linguistik (Sprachwissenschaft) sind:	☐	☐	☐	☐	☐
5.	Meine Kenntnisse in Informatik sind:	☐	☐	☐	☐	☐
6.	Meine Kenntnisse in Bezug auf formale Abfragesprachen (z.B. SQL, RDQL, SPARQL etc.) sind:	☐	☐	☐	☐	☐
7.	Meine Englischkenntnisse sind:	☐	☐	☐	☐	☐
8.	Meine Italienischkenntnisse sind:	☐	☐	☐	☐	☐

C

Content of the CD-Rom

Path	Content
/Ginseng088/src	Source code of Ginseng
/Ginseng088/bin	Compiled Ginseng source
/Ginseng088/lib	Libraries required for Ginseng
/Ginseng088/doc	Javadoc
/Ginseng088/grammars	English and the Italian Ginseng grammars
/Ginseng088/owl	RDF and OWL files for Ginseng, including *www.mooney.net geo.owl*
/Ginseng088/survey	Material for surveys
/Ginseng088/test	Textfiles with containing English test queries
/Ginseng088/ginseng_v088.jar	The jar file containing a runtime version of Ginseng
/Ginseng088/run.bat	Batch file which starts ginseng_v088.jar
/Ginseng088/readme.txt	Readme file
/Thesis/Abstract.pdf	English Abstract
/Thesis/Zusammenfassung.pdf	German Abstract
/Thesis/GinsengGoesItalian.pdf	File containing this document
/Evaluation/EinfuehrungInstruktionen.pdf	Instructions for the Evaluation
/Evaluation/susFragebogen.pdf	SUS questionnaire
/Evaluation/personenFragebogen.pdf	Questionnaire for retrieving personal information about users

Table C.1: Overview of the Content of the CD-Rom

List of Figures

List of Tables

List of Listings

Bibliography

[Androutsopoulos and Galanis, 2007] Androutsopoulos, I. and Galanis, D. (2007). Generating multilingual descriptions from linguistically annotated owl ontologies: The naturalowl system. In *11th European Workshop on Natural Language Generation (ENLG 2007)*, Schloss Dagstuhl, Germany.

[Androutsopoulos et al., 2005] Androutsopoulos, I., Kallonis, S., and Karkaletsis, V. (2005). Exploiting owl ontologies in the multilingual generation of object descriptions. In *10th European Workshop on Natural Language Generation (ENLG 2005)*, pages 150–155, Aberdeen, UK.

[Baeza-Yates and Ribeiro-Neto, 1999] Baeza-Yates, R. and Ribeiro-Neto, B. (1999). *Modern Information Retrieval*. Addison-Wesley Longman, Harlow, UK.

[Berners-Lee et al., 2001] Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The semantic web. *Scientific American*, 284(5):34–43.

[Bernstein and Kaufmann, 2006] Bernstein, A. and Kaufmann, E. (2006). Gino - a guided input natural language ontology editor. In *5th International Semantic Web Conference (ISWC 2006)*, pages 144–157. Springer.

[Bernstein et al., 2005] Bernstein, A., Kaufmann, E., and Kaiser, C. (2005). Querying the semantic web with ginseng: A guided input natural language search engine. In *15th Workshop on Information Technology and Systems (WITS 2005)*, pages 45–50.

[Bernstein et al., 2006] Bernstein, A., Kaufmann, E., Kaiser, C., and Kiefer, C. (2006). Ginseng: A guided input natural language search engine for querying ontologies. In *Jena User Conference, Bristol, UK*.

[Bray et al., 2006] Bray, T., Maler, E., Yergeau, F., Sperberg-McQueen, C. M., and Paoli, J. (2006). Extensible markup language (XML) 1.0 (fourth edition). W3C recommendation, W3C. http://www.w3.org/TR/2006/REC-xml-20060816.

[Breitman et al., 2007] Breitman, K., Casanova, M., and Truszkowski, W. (2007). *Semantic Web: Concepts, Technologies and Applications*. Springer London.

[Brooke, 1996] Brooke, J. (1996). Sus - a "quick and dirty usability scale. In Jordan, P., Thomas, B., Weerdmeester, B., and McClelland, A., editors, *Usability Evaluation in Industry*. Taylor and Francis, London.

[Guha and Brickley, 2004] Guha, R. V. and Brickley, D. (2004). RDF vocabulary description language 1.0: RDF schema. W3C recommendation, W3C. http://www.w3.org/TR/2004/REC-rdf-schema-20040210/.

[Ishikawa et al., 1986] Ishikawa, H., Izumida, Y., Yoshino, T., Hoshiai, T., and Makinouchi, A. (1986). A knowledge-based approach to design a portable natural language interface to database systems. In *Proceedings of the Second International Conference on Data Engineering*, pages 134–143, Washington, DC, USA. IEEE Computer Society.

[Kaiser, 2004] Kaiser, C. (2004). Ginseng - a natural language user interface for semantic web search. Diploma thesis, University of Zurich, Zurich, Switzerland.

[Klyne and Carroll, 2004] Klyne, G. and Carroll, J. J. (2004). Resource description framework (RDF): Concepts and abstract syntax. W3C recommendation, W3C. http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/.

[Layman et al., 1999] Layman, A., Hollander, D., and Bray, T. (1999). Namespaces in XML. first edition of a recommendation, W3C. http://www.w3.org/TR/1999/REC-xml-names-19990114.

[Manola and Miller, 2004] Manola, F. and Miller, E. (2004). RDF primer. W3C recommendation, W3C. http://www.w3.org/TR/2004/REC-rdf-primer-20040210/.

[McCool, 2005] McCool, R. (2005). Rethinking the semantic web, part 1. *IEEE Internet Computing*, 9(6):88–87.

[McGuinness and van Harmelen, 2004] McGuinness, D. L. and van Harmelen, F. (2004). OWL web ontology language overview. W3C recommendation, W3C. http://www.w3.org/TR/2004/REC-owl-features-20040210/.

[Popescu et al., 2003] Popescu, A.-M., Etzioni, O., and Kautz, H. (2003). Towards a theory of natural language interfaces to databases. In *8th Intl. Conf. on Intelligent User Interfaces*, pages 149–157, Miami, FL.

[Seaborne and Prud'hommeaux, 2007] Seaborne, A. and Prud'hommeaux, E. (2007). SPARQL query language for RDF. Candidate recommendation, W3C. http://www.w3.org/TR/2007/CR-rdf-sparql-query-20070614/.

[Tang and Mooney, 2001] Tang, L. R. and Mooney, R. J. (2001). Using multiple clause constructors in inductive logic programming for semantic parsing. In *12th European Conf. on Machine Learning (ECML-2001)*, pages 466–477, Freiburg, Germany.

[W3C, 2007] W3C (2007). W3c world wide web consortium. `http://www.w3.org/`.

[Wehmeier, 2000] Wehmeier, S., editor (2000). *Oxford Advanced Learner's Dictionary of Current English*. Oxford University Press, sixth edition.

[Welty et al., 2004] Welty, C., McGuinness, D. L., and Smith, M. K. (2004). OWL web ontology language guide. W3C recommendation, W3C. http://www.w3.org/TR/2004/REC-owl-guide-20040210/.