Master in Artificial Intelligence (UPC-URV-UB)

Master of Science Thesis

Using Application-Specific Ontologies to Generate Grammar Rules in English, Hindi, and Spanish for a Web Interface

Piyush Paliwal

Advisor: Dr. Marta Gatius

September, 2013

Abstract

This thesis is concerned with the use of domain ontologies facilitating the generation of multilingual grammars, which furthermore can be integrated in the natural language understanding module of a communication system. In particular we work on grammars for supporting user queries when accessing the web in English, Hindi and Spanish in two scenarios: searching for a new medical specialist and looking for the information about cultural events in the city. Although there have been many works on communication systems supporting English and Spanish, this is not the case for Hindi language. For economical and cultural reasons there have not been many studies on the integration of Hindi language on communication systems. For this reason, our thesis also deals with the difficulty of working on a language for which not many studies have been done nor do existing resources exist.

In order to facilitate the generation of linguistic resources for the three languages in different domains we propose a clear separation of the conceptual and linguistic knowledge, as well as a separation of general and domain-restricted knowledge bases, being conceptual knowledge. Conceptual knowledge is represented in ontologies and is reused across the three languages. Linguistic knowledge is language specific. General knowledge consists in general conceptual concepts represented in a general ontology and general linguistic knowledge can be represented as general grammars rules that can be reused across domains. For developing the grammar rules for each domain and language we use grammatical framework (GF), a powerful tool for writing multilingual grammars. One of the main advantages of this formalism is that favours a clear separation of conceptual and syntactic knowledge involved in a particular grammar: it represent the conceptual knowledge in a module called abstract grammar and the syntactic details in a separate but related module called concrete grammar. We define abstract grammar from ontologies and they are base to further developing the concrete grammar.

Acknowledgements

I would like to thank, first and foremost, my thesis supervisor, Prof. Marta Gatius**,** for providing me fruitful guidance and persistent help. Her time-to-time suggestions during the planning and development of the project as well as writing the thesis have enriched the work-progress.

I am also grateful to all the professors from different disciplines with whom I interacted during the Master's and gained streamy knowledge.

Last but not the least, I can never forget the never-ending support accommodated by my family and close friends who, despite being miles away, have always motivated me.

Piyush Paliwal

Septembet, 2013

Barcelona, Spain

I hereby declare that this thesis, to the best of my knowledge, is solely my own work. Citations are provided when the work of others is used as for references.

Contents

Chapter 1

Introduction

1.1 Insights and Technical Background

With the constantly increasing demands for development in this emerging web society, the time has reached a point, where children are born with attractive machinery devices integrated with several assistive natural language (NL) interface systems. These NL interfaces are benchmarks that confer manifold assistance in many different activities: education systems, health assistances, or just for entertainment.

At the same time, more and more people are inclined towards the web usability, the research on web interfaces supporting different languages and modalities to assist users is growing rapidly. One of the main challenges in NL interface systems is to understand correctly the user's needs. This problem can be faced substantially using two different approaches: machine learning based methods, or conceptual and linguistic knowledge based method. The machine learning based approach needs a large corpora to learn but can be of great help in recognizing an incremental number of user's queries. The conceptual and linguistic based approach uses conceptual and linguistic knowledge that have to be developed by skilled professionals. Both approaches present several advantages and limitations. The machine learning based approach can perform worse if the data is not enough. On other hand, the knowledge based approach is expensive to develop, since an ample linguistic as well as domain knowledge have to be developed. Furthermore, developing the resources in different languages intensifies the workload. In this context, the use of domain ontologies representing the domain knowledge presents several advantages to simplify the complexity of dealing with multilinguality and multimodality.

Ontologies are formal representations of world-knowledge where the entities and relations are explicitly described. In a given context, the communication process often evolved the contents that are focused on a particular word-knowledge (i.e domain). This is how a child is taught to learn to speak and learn the language. Furthermore, when we try to communicate in different languages, conceptual knowledge is shared at the very first reasoning. This forces to separate the conceptual knowledge about the domain from the linguistic knowledge of a particular natural language. Additionally, several modules can be further defined separating the general conceptual knowledge from the domain specific knowledge.

This is where this Master thesis comes in, with the use of application- specific ontologies to facilitate the domain-restricted communication by separating the conceptual knowledge from the linguistic knowledge in three different languages: English, Spanish, and Hindi.

1.2 Thesis Objectives

The goal of this Master thesis is to use the application-specific ontologies to facilitate the two basic processes of NL interface systems: natural language understanding (NLU) and natural language generation (NLG).

The NL understanding is to be done by implementing the multilingual grammars in three different languages: English, Spanish, and Hindi. We work with two different domain scenarios: health domain and cultural events domain for each language. In particular, we have focused the first scenario where the user is looking for new medical specialists and the second scenario where the user searches practical information about cultural events in city.

The reason of working with two different domains is to meet our purpose of reusing the shared knowledge across domains and for that we organize the knowledge in a structured representation in ontologies. For implementing the grammars we use a tool Grammatical Framework (GF) [A. Ranta, 2004], a multilingual formalism. The GF grammar is composed of abstract syntax and a set of concrete syntaxes. The abstract syntax defined the meaning of user interventions and is same for all the languages, where the concrete syntax is languages dependent defined for each language. We define a correspondence between domain ontologies and abstract syntax and develop the concrete grammars using the abstract grammar.

We also describe the syntactic-semantic taxonomy (Gatius, 2001) adapted to health domain for NLG component. But this thesis will not detail this taxonomy at implementation level, but rather to provide a framework that can be easily extended and adapted to different domains and languages.

Evaluation is based on the measure of the reusability of grammars rules at different level of integration. The scope of this thesis is not to achieve a larger coverage of grammars that will recognize an arbitrary number of user queries, but rather construct the rules that can be reused or at least easily adapted across domains and languages.

Finally we experiment with some test examples from our developed multilingual grammar resources and demonstrate some applications thereon.

1.3 Thesis Layout

The layout of the thesis is as follows:

Chapter 2 State Of the Art

This chapter introduces the general architecture of a typical conversational interface system, describes the state-of-the-art of several assistive NL interface systems that use ontologies.

Chapter 3 Generating Grammar Rules using Application-Specific Ontologies

In Chapter 3 an introduction to the Grammatical Framework is given. Then the work done in this thesis is described for generating the grammar rules in two domain scenarios: health domain and cultural events domain.

Chapter 4 Natural language Generation

This chapter provides a brief description of the syntactic-semantic taxonomy adapted to health domain for facilitating the system answers generation.

Chapter 5 Grammar Rules Evaluation and Experimental Examples

Evaluation metrics of grammars developed are specified in this chapter. Then the experimental examples are demonstrated.

Chapter 6 Conclusion

The conclusions drawn from this thesis work are provided in this chapter.

Chapter 7 Future Direction

Finally, suggestions to guide the future work are given in this chapter.

Chapter 2

State of the Art

This chapter briefly reviews the most relevant web interaction systems that use ontologies to facilitate the developing process. Section 2.1 describes the general classical architecture of the web interface systems. Section 2.2 discusses about the most relevant ontology-based interface systems and their functionalities.

2.1 General Architecture of a Conversational Interface System

A typical web conversational interface system integrates several modules or components. Each of these components has unique functionality that must be carried out within a system. This section describes a generic architecture of a typical conversational system and their functionalities as defined in (Zue and Glass, 2000).

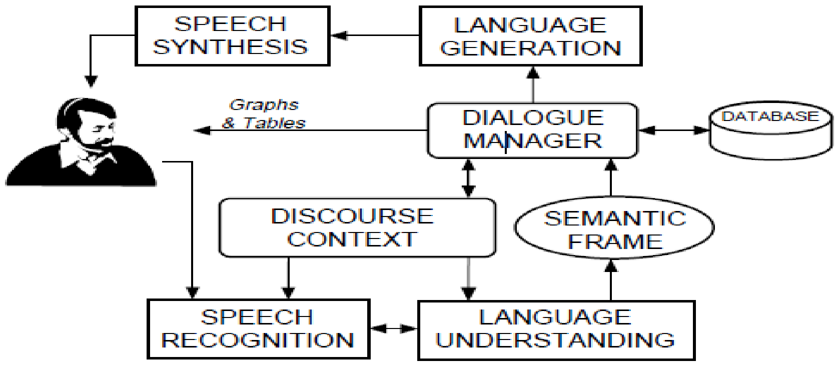


Figure 2.1: *Generic Block Diagram for a Typical Conversational System*

As shown in Figure 2.1, the component speech recognition (SR) is used to process the spoken input, i.e. to convert the speech signals into textual form. With the rapid advancements in speech technologies, SR has now shown to be effective against errors and disfluencies, like filled pauses, word fragments, and unknown words.

The language understanding (NLU) component extracts the meaningful information from the utterances. The utterances are either the output of speech recognition component or the typed-text input from the user, depending on whether the input modality is text-based or speech-based. This analysis is traditionally syntax-driven, that is, it takes into account all the words in an utterance. However, this approach presents several problems, like the user often gives grammatically incorrect sentences or incomplete sentences and in case of spoken input, the recognizer can give errors (J. Dowding et al., 1993). To deal with such problems, many works favor more semantic-driven approaches where the spotting keywords are enough to derive the relevant meanings using the robust parser (W. Ward, 1990).

The syntactic and semantic analysis of the utterances is not enough. A more in-depth analysis is needed, since they convey a context-independent meaning representation of the utterances. In order to have an effective communication, the system must also have ability to inherit information from the previous utterances that the user has followed so far. For example, if a user says, "*can you find me a doctor*?" followed with, "*I have some problems in my heart*" in a previous utterance, then the system must not provide all the different types of doctors, but the heart experts only, nor the system must further clarify with the user about the doctor's specialist. To deal with such issues, the system must consider context-information (one or more previous utterances) when analyzing the current utterance of the user. Such discourse-context information is thus maintained by a discourse-context component.

The natural language generation component (NLG) plays just an opposite role of NLU component, hence is used to generate the answers to the user query. This generation process can range from a very simple, either template based or simple grammar based (Glass et al., 1994), approaches to the sophisticated corpus-based methods (Oh, 2000). However, the simple approaches lack the generation expressivity in complex-domain applications. The output of NLG component is then fed to speech synthesizer, a text-to-speech synthesis (TTS), to provide a spoken response to the user's query. TTS systems, in their early stages of development, were rule-driven where the text input is first constituted into an abstract linguistic representation (Klatt, 1987). However, due to the lack of naturalness in such systems researchers have tended towards their further findings and the corpus-based approaches are one of their explorations (Sagisaka et al., 1992).

All these aforementioned components are linked to a dialogue manager (DM) which controls the flow and interaction between user and system.

2.2 Ontologies and Web Assistive Interface Systems

Developing the user friendly and collaborative interfaces is becoming crucial because the real world knowledge is immense and unstructured. In this direction, during the past years, ontologies have been used in several assistive natural language interfaces (NLIs) and have shown plausible improvements both in the development process and the usability of such interfaces.

The purpose of using ontologies is to identify and characterize the central concepts and relations in the real world, typically the ones used in a particular application (health services, travel, form-filling, etc.). It therefore facilitates the developing process.

Several directions conducted towards the use of ontologies have substantiated to be a greatest payoff in their quests. Some of the most relevant systems and architectures are revised in this section.

The SMARTWEB System

Our work of thesis proposes the use of ontologies for obtaining the linguistic resources where the ontologies are hand-crafted, whereas the system SMARTWEB incorporates some previously established ontologies in addition to some conceptual developed ontologies. Therefore, the system is reusing the existing knowledge available from some fundamental ontologies. Apart from the generic interface components, like NLU, NLG, DM, the system also consists of several specific modules (as will be described in the following introduction) which make the system more robust.

SMARTWEB is multimodal mobile interaction platform (D Sonntag et al., 2007) used for assisting the user when accessing the web services on Football World Cup in Brazil (in 1998) and in Germany (2006). The ontology used in this architecture, SWInto (SmartWeb Integrated Ontology) (Oberle, D et al, 2006), is an integrated ontology merging the two different fundamental established ontologies: DOLCE (Gangemi, A et al, 2002) and SUMO (Niles, I., Pease, A., 2001). These fundamental ontologies are sharing world knowledge for this application. In addition to these ontologies, two more ontologies are added: DiscOnt (discourse ontology), which pays attention to modeling the discourse interactions in a question-answering (QA) scenarios, and the SMARTMEDIA (media ontology), which is used to represent the multimodal information results. In a QA scenario, system supports flexible control flow allowing the clarification questions of web services.

Supporting multimodality functions in this particular architecture are verbal and non-verbal communications with the user. In order to access to the web, semantic representation formalism based on OWL-S and a service composition engine that exploits the semantic of user query are developed. Apart from this knowledge-intensive component (i.e, ontology), the system has other components: a speech interpretation component (SPIN), a modality fusion and discourse component (FADE), a system reaction and presentation component (REAPR), a natural language generation component (NIPSGEN). All of these processing components are integrated with a Java-based hub-and-spoke architecture (Reithinger, N., Sonntag, D, 2005).

ORAKEL System

This system is more focused on adapting the interfaces to different domains, for that the system mainly separates the conceptual knowledge into different lexicon databases. This thesis work is also focused on adaptation to different domains, but also work on multilinguality. In both systems the general conceptual knowledge unique to different domains are to be reused in all the domains.

ORAKEL system as discussed in (Cimiano et al., 2007) corroborates the claim of porting the natural language interfaces (NLIs) to new domains. The lexicon components are basically broken down in three separate lexicons consisting of domain-specific lexicon, domain-independent lexicon, and ontology lexicon. For generating the domain-specific lexicon, the user maps the linguistic expressions, such as verbs, adjectives, relational nouns that occur in a specific domain, to the corresponding domain-specific predicates. This mapping model is accomplished by using a FrameMapper and the aftermath of the mapping performed by a lexicon engineer is deployed to the system in an integrative manner. Therefore, the user is not supposed to being familiar with computational linguistics for generating the domain-specific lexicon. The domain-independent lexicon contains the meaning representation of the domain-independent words, like determiners, pronouns, prepositions, across the several domains. Such meaning representation is accredited with reference to the categories provided by a well-established ontology DOLCE (Masolo et al., 2003). The ontology lexicon is automatically derived from the domain ontology and contains the lexical entries and meaning representation of all the concepts and instances of the ontology. The benefit of having a modularized structure of the lexicon components is that the domain-independent lexicon can be used for every different domain.

Except these lexicon components, the system incorporates a query interpreter which constructs the meaning representation of the user's query through lambda expressions and it then transforms query into a First-Order-Logic (FOL)-like language. This FOL query further can be converted into any specified target language by a query converter component.

As for the results, a user study was carried out where the knowledge base containing the geographical facts, such as cities, states, rivers and highways, about Germany was used as a demonstration. Moreover, a case study was carried out at British Telecom to fulfill the aim of enhancing the search over a digital library.

Active Platform

This platform combines several Active ontologies which have several relational notations that can be of help in under/over specification. Similar proposal has been developed in this thesis work where onologies are made not so complex, but can comprise additional relational information.

Active platform, as introduced in (Guzzoni et al., 2006), provides an intelligent mobile assistance able to retrieve the online information for the user's query. This platform incorporates the use of one or more Active ontologies. The Active ontology is a formal representation of the domain knowledge defining the domain in terms of classes, attributes and their relations. The Active platform is java-based and has three components. The first component is the Active Editor which is used by the developers for modeling, deploying, and testing the applications. The second component is the Active Server which is a run-time engine that executes the Active programs. The third component is the Active Console that allows remote configuration of the Active Server.

In particular to the use of Active ontologies, Active framework uses two types of relational ontological notations: "is-a" classification relationships and "has-a" structural relationships. One of the classical ability of this framework is that it deals with the discourse context information, where the system when analyzing the current user sentence can inherit the previous utterances, thus providing an effective communication. Such type of communication is indeed appropriate in mobile communications because of the less bandwidth and interface complexity thereon.

The use of specialized Active ontology has shown to be conducive in the sense that it eases the process of registering and dynamically selecting the web services since the service categories are represented with the concepts and relationships of Active ontology.

PANTO System

The work of Wang et al. (2007) presents the PANTO, a portable natural language interface to ontologies, that aims at providing users the convenience in the sense that to map users' queries to formally defined ontologies. Consequently, improving the query' semantic interpretation. The PANTO system integrates the WordNet (Fellbaum, 1998) and string metrics algorithms (Cohen, 2003) that help the system in mapping NL queries to concepts and relations in ontology. It basically consists of three modules including ontology processing, query processing, and translator. First two modules process the user query input and using the lexicon database in ontology give the output in SPARQL formal query language. Inside query processing module, they system has a StanfordParser to generate the parse tress. The parse tree is then transferred to translator. The translator first transforms parse tree into QueryTriple which is mapped to OntoQuery by the help of lexicon in query processing module. The basic idea of translating the natural language queries into formal ones is to allow a deep parsing to observe the nominal-phrase pairs. The use of ontologies allows matching of nominal-phrase pairs with the facts that are like a triple form, i.e. <subject, predicate, object>, knowledge stored in ontology. In this triple form, the subject and object may be the concepts or instances in ontology, the predicate are most probably verbs, verb phrases, prepositional phrases, or nominal phrases. This triple-based analysis is very effective in interpreting the natural language queries.

GEMINI Platform

GEMINI (Generic Environment for Multilingual Interactive Natural Interfaces), an EC research project, is a generic platform which assists in developing the user-friendly multilingual and multimodal dialogue systems (Hamerich et al., 2004). The main contribution of GEMINI project is to minimize the development time, maintenances, and human efforts at providing the adaption to numerous modality services and languages. In this behalf, GEMINI project exploits the Application Generation Platform (AGP) as architecture that integrates set of assistants. The architecture is divided into three layers: framework layer, retrievals layer, and dialogue layer.

The framework layer has several assistants including the application description assistant (ADA), the data modeling assistant (DMA), and the data connector modeling assistant (DCMA). The user first provides the system all necessary information regarding, for instance, the type of modalities, his/her language preferences for dialogue communication. Then the application model is defined by consecutive assistants and the output is created in XML-based files. The retrievals layer consists of retrieval modeling assistant (RMA), which is independent to the modality and language, and is used to produce application flow. Finally the dialogue layer is modality and language dependent and consisting of modality extension assistant (MEA). This assistant enables the layer results the dialogue model for a specific modality. In case the chosen modality is speech, the resulting dialogue model is processed by a speech script generator to generate the VoiceXML scripts, or is processed by a screen script generator to generate the xHTML scripts, if the user chooses a web modality.

GDialogXML, a new abstract dialogue description language (Hamerich et al., 2003), is the object oriented language that supports the generation of dialogue applications within the GEMINI platform. This language makes use of the concepts for dialogue flow. The dialogue models written in GDialogXML generate the previously mentioned dialogue scripts (i.e. VoiceXML or XHTML scripts).

DIGUI System

The work of thesis is similar in many aspects to the work done for the DIGUI system. Both works convey the same message of using the ontologies for obtaining the linguistic resources for NLU, NLG components. The DIGUI system incorporates all the generic modules that the system needs to fulfill various tasks, whereas the current thesis work focuses on obtaining the linguistic resources for only two modules of interface system: NLU and NLG. In addition, a new language: Hindi and a new domain scenario: health assistive services have been studied using a new existing framework, Grammatical Framework, for writing grammars in our work.

DIGUI (DIalogues Guiding the User Interaction) system developed by (González, 2010) aims at providing the flexible and user friendly mixed-initiatives dialogue system that is adaptable to different web services and is accessible in different languages including Spanish, Catalan, and, English, and with the different modes of communication: text and speech. The system has four general components: natural language understanding (NLU), dialogue manager (DM), natural language generation (NLG), and task manager (TM).

The NLU component analyses the various types of concerns the user may pose to dialogue agent. This component traverses those concerns through the semantic grammars and lexicons that are obtained from the conceptual linguistic resources. The DM module uses the information state update (ISU) approach which controls the communication plans and decides the possible action that the system must follow at each turn by means of the set of rules. The DM also dynamically keep contacting with one or more TMs. The TM module advances the process of accessing the web services by the use of general schemas defined for this particular module. Adapting to the new web services is achieved by these general schemas.

The NLG component produces system answer to the user query. It uses the output of DM manager and decides how to produce the answer in natural way that must be followed for an effective communication. The developed system semi-automatically generates the main phrases by adapting the syntactic-semantic taxonomy (Gatius, 2001) to the particular application used. This taxonomy eases the adaption process to different domain scenarios. Several patterns are also described that can represent phrases in different way. Therefore the system can vary the same intended message in different forms resulting in a more flexible and friendlier communication.

As a summary, the system attains efficiency, adaptability, and multilinguality with the modular architecture described. The different linguistic resources such as language resources, domain resources are isolated, thus can very effectively facilitate the adaptation.

2.3 Natural Language Generation

Natural language answer generation (NLG) is an important discipline involved in several NL applications: dialogue systems, summary generators, systems generating documentation from programs, etc. Although several NLG problems are common in all applications the approaches to them may change considering the type of application.

In communication systems the NLG, although is the opposite process to NLU, it is much simpler. The answers are usually generated by mapping the output of DM to a natural text understandable by the human. Within the system as whole, the NLG must decide how to express the contents to user. The DM however may already have provided the dialogue moves, that is, the content to be presented to the user and its form (as a question, as a description, etc). In that case the NLG component only needs to organize those moves in human readable text. Many of the systems use grammar based rules or hand-crafted written texts that correspond with the dialogue moves obtained from the DM. In many other practical interface systems where not so complex reasoning about the previous utterances is needed (i.e., simple interfaces to databases), the DM and the NLG are not implemented as independent modules. In those systems the semantic meaning representation obtained from the NLU component is transformed into a query to the database and the results are presented to the user. The NL answer generation can be limited to a set of sentences ("*Those are the results*", "*No results are found*", "*Do you want ask anything else to the system*", etc.). For these systems the NLG process may be limited to a simple set of rules in charge to select a particular sentence from a previously defined set.

Research communication systems can also include more complex NLG that may generate automatically complex responses in any domain (and in different languages). NLG in those systems consists of two separated steps: first the generation of the content and then the linguistic realization (expression in language of the content). They can use several methods for this process: statistical- based, knowledge-based and also combining statistical and knowledge-based. In this thesis we have followed a particular approach to language generation, based on (Bateman, et al., 1994). This approach uses a syntactico-semantic ontology to relate conceptual knowledge in a conceptual ontology to the general linguistic structures needed for their realization.

In particular, in this thesis we use the work described in (Gatius,2001) that uses a syntactic-semantic taxonomy of the attributes of the concepts described in the ontology. Each attribute class is associated with one or more general forms to express the basic operations of filling and consulting the attributes represented by the class. The seven basic attribute classes in this taxonomy are associated with grammatical roles: participants (the three classes WHO_DOES, WHO_OBJECT, WHAT_OBJECT), being (the class IS), possession (the class HAS), descriptions and relationships between two or more objects (the class OF) and related processes (the class DOES). Subclasses are obtained from basic classes considering other information relevant for the linguistic realization of attributes. The OF class was subdivided into three classes: OF_PERSON, OF_OBJECT and OF_DESCRIPTION. The class OF_PERSON describes relations between the concept representing a person and one or more persons. The class OF_OBJECT represents relations between the concept and one or more objects. The class OF_DESCRIPTION represents qualities and circumstances related to the concept. The subclasses have been further sub classified considering specific linguistic details in the expression of the attributes in the class, such as having an associated verb or preposition. The class OF_DESCRIPTION was subdivided into the classes: OF_TIME, OF_PLACE, OF_MANNER, OF_CAUSE, OF_QUANTITY, OF_NAME, OF_TYPE. These subclasses represent attributes describing time, place, manner, cause, quantity, name and type respectively. For example, the class OF_QUANTITY describes attributes referring to quantities. Attributes in this class always involve the use of a unit of measure. The interrogative adverb cuánto/cuántos (how much/how many) appearing in the interrogation clauses expressing consult operations on these attributes is also included in the description of the class. In Spanish, attributes expressing a quantity have an associated verb (which corresponds to an associated adjective in English). They have been further subclassified considering specific linguistic details in the expression of the attributes in the class, such as having an associated verb or preposition. The main benefit of this taxonomy of attributes is that it is designed in a way to be reusable across various languages. In the Chapter 4, we describe this taxonomy of attributes is been used for the scenario in the health domain we have considered.

2.4 Summary of State-of-the-Art

In this chapter we briefly introduced about the fundamental architecture of the web interface system. The continuing section reviewed some of the most relevant web interface systems that use ontologies for facilitating the development process.

Next continued section described the natural language answer generation phenomenon from the system side. In particular, we focused on reviewing the syntactic-semantic taxonomy architecture which will also be used in this thesis work in Chapter 4 when facilitating the system answer generations in multiple languages and domain scenarios.

Chapter 3

Generating Grammar Rules using Application-specific Ontologies

Obtaining the linguistic resources for NL interface systems is not a trivial task especially when its impact will exhibit more useable, robust, efficient, and adaptive system. Application specific ontologies reduce the human efforts in acquiring the application-restricted knowledge resources in different languages. The same ontology is the basis for generating the linguistic rules in multiple language scenarios and hence can greatly help meet the underlying goals during the development process.

This chapter describes the work done in this thesis. After introducing the system architecture, we begin with Section 3.1 where a brief introduction to Grammatical Framework, a tool used in our work, has been given. In Section 3.2 the conceptual knowledge for health domain is formalized and described. Section 3.3 covers all the significant phases required for obtaining the grammars rules in health domain scenario. Section 3.4 and 3.5 cover all the conceptual knowledge defined for cultural events domain and the grammars rules obtained, respectively. Finally we summarize this chapter in Section 3.6.

This Figure shows a system that incorporates the ontologies knowledge, GF grammars (for NLU) and syntactic-semantic taxonomy (for NLG). Our current work does not use the taxonomy for NLG, we however provide the representation of taxonomy adapted to medical domain and that can be extended easily to different concepts appearing in different domains as well as to different languages. Our work is mainly focused on grammar development for NLU component of the system. For generating the system responses for a toy demonstration, we use the output of NLU component and look up into a query database and then generate the appropriate canned sentences. For NL understanding we have used Grammatical Framework (GF), a very well suited tool for facilitating the grammar developments. Now let us first introduce the basic concepts about this tool.

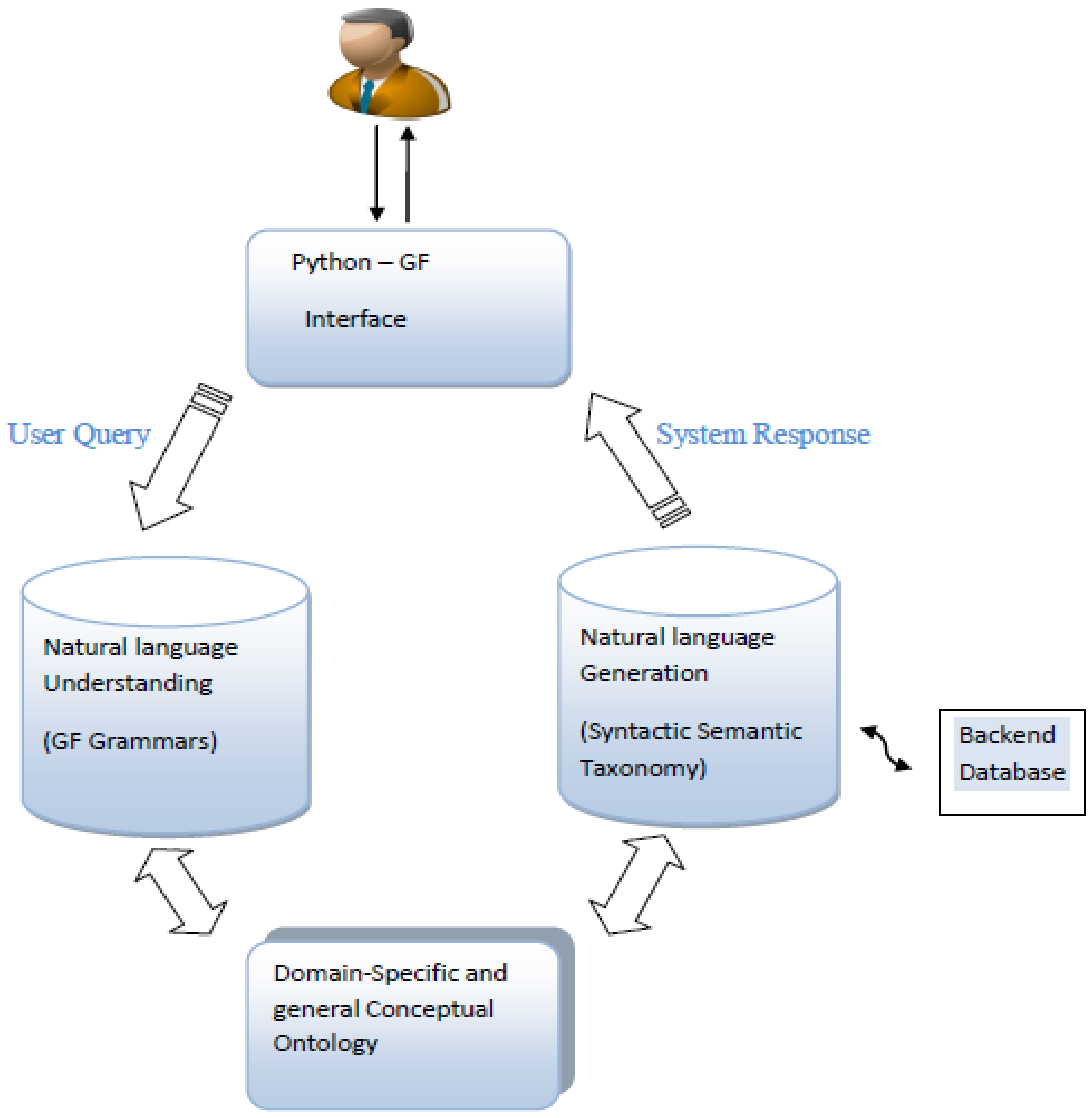

Python – GF
Interface
User Query
System Response
Natural language Understanding
(GF Grammars)
Natural language Generation
(Syntactic Semantic Taxonomy)
Backend Database
Domain-Specific and general Conceptual Ontology

Figure 3.0 *System Architecture*

3.1 Introduction to Grammatical Framework

This section provides an abridged introduction to the grammatical framework (GF) (A. Ranta, 2011), a powerful tool well suited for writing multilingual grammars and building applications thereon.

The GF is a functional programming language which was discovered based on the idea of constructive type theory (Martin-Löf, 1982). The main underlying feature of GF language is a clear separation of the grammars into two components: abstract syntax and concrete syntax. The abstract syntax has two parts: (i) a list of *cat* or *category declarations* introducing the main conceptual domain entities and their meanings and (ii) a list of *fun* or *function declarations* defining how the categories manipulate domain entities to form their semantics to be used in communication process. The concrete syntax consisting of (i) a list of *lincat* or *linearization type* assigning a linearization type to each category in *cat* and (ii) a list of *lin* or *linearization* telling how the categories and functions are linearized in a particular natural language.

A single abstract grammar can be defined for a particular application domain whereas concrete grammars can be as many as of natural languages involved. The abstract grammar is a language-independent component and only domain knowledge is required to define it. On other hand, each concrete grammar would need a linguist expert dealing with naturally evolved complexities in target natural language (i.e. variations in word orders, different linearization types, agreement features, etc.). However, it is worth noting that the inherent engineering features and functionalities of GF greatly help human experts ease the generation of rules in each language because of the shared syntactic rules in abstract grammar.

The format of cat and fun in abstract syntax:

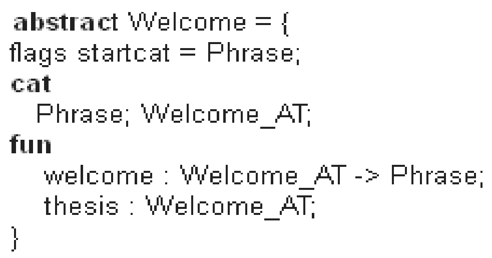
cat C1, C2, …Cn	There are total n categories
fun f1 : T1, f1 : T2, …., fm : Tm	There are total m functions (so-called syntactic rules), each of them has function type (for e.g. f1 is a function of type T1)

The format of lincat and lin in contract syntax:

lincat C1 = L1, C1 = L2, … Cn = Ln	There are total n categories, each of them has a linearization type, for e.g. category C1 has a linearization type L1
lin f1 = t1, f2 = t2, …, fm = tm	There are total m functions (so-called linearization or syntactic rules), each of them has a linearization function, for e.g. function f1 has linearization function t1

There must be at least one category (i.e. n≥1) and one function (i.e. m≥1) defined in abstract syntax and so must be in concrete syntax the lincat and lin, since they are equal in number of that of the abstract cat and fun, respectively. The n and m may differ in number, though. Following these can lead us to one working grammar application.

Let us illustrate a discernible toy example expressed by grammars: an abstract grammar (Figure 3.1.1), a concrete English grammar (Figure 3.1.2), and a concrete Spanish grammar (Figure 3.1.3) that are capable to parse a demo sentence "*Welcome to my thesis*" in both languages. The Figures 3.1.4 and 3.1.5 show the parse tree representation for English and Spanish, respectively.

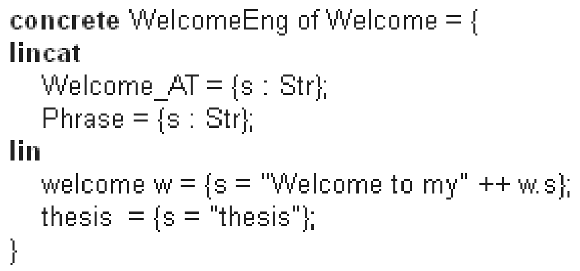


```
abstract Welcome = {
flags startcat = Phrase;
cat
  Phrase; Welcome_AT;
fun
   welcome : Welcome_AT -> Phrase;
   thesis : Welcome_AT;
}
```

Figure 3.1.1 *demo abstract Welcome grammar*

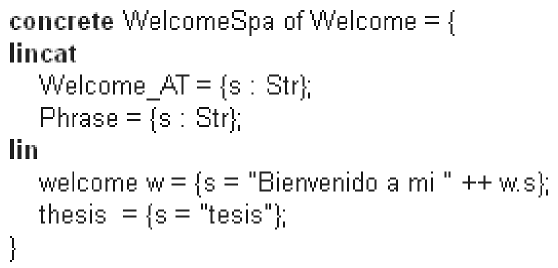
The abstract grammar represents its main parts described earlier. The "*Phrase*" and "*Welcome_AT*" are two categories and the two functions named "welcome" and "thesis" describe how categories are manipulated. A startcat flag declaration defines the default start category (i.e. Phrase) to be used in parse tree.

The concrete grammars for English and Spanish are relating the linguistic formulation in concerned languages to the abstract grammar. All the non-linguistically constructed syntactic rules in abstract syntax are to be shared by all the concrete syntaxes, while the rules in concrete syntaxes alter depending on the language and their formulation (i.e., the lexicon or other morphological syntactic rules), for e.g., "*thesis*" in English and "*tesis*" in Spanish.



```
concrete WelcomeEng of Welcome = {
lincat
   Welcome_AT = {s : Str};
   Phrase = {s : Str};
lin
   welcome w = {s = "Welcome to my" ++ w.s};
   thesis  = {s = "thesis"};
}
```

Figure 3.1.2 *demo concrete English Welcome grammar*



```
concrete WelcomeSpa of Welcome = {
lincat
   Welcome_AT = {s : Str};
   Phrase = {s : Str};
lin
   welcome w = {s = "Bienvenido a mi " ++ w.s};
   thesis  = {s = "tesis"};
}
```

Figure 3.1.3 demo concrete Spanish Welcome grammar

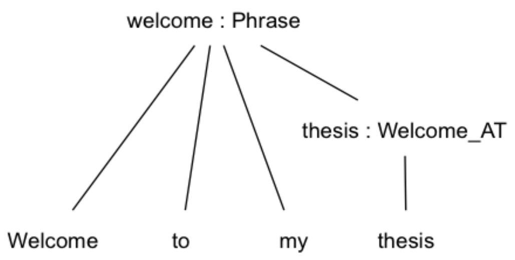


Figure 3.1.4 parse tree structure for English grammar

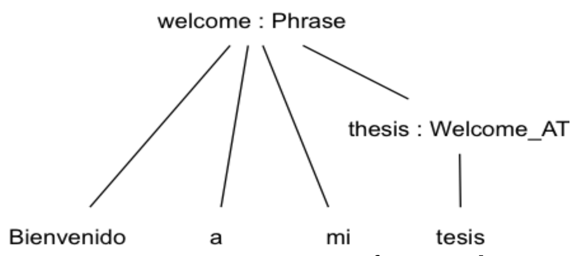


Figure 3.1.5 parse tree structure for Spanish grammar

With GF, we can use grammars for several purposes as follow:

- **Linearization and Parsing**

  Linearization is a mapping of tree-like representation of abstract syntax (i.e. syntactic functions) to strings. Inversely, the parsing of any natural language sentence recognized by concrete grammar is a mapping of strings to tree-like syntactic functions.

- **Translation**

  The Sentences can be translated into other languages within the developed multilingual grammar resources. Here translation from language L1 to L2 is just parsing in L1 and linearization to L2.

- **Guiding the user access the resources**

  The system can guide the user next possible acceptable words, resulting in a friendlier conversation.

- **Random generation of acceptable sentences**

  All possible sentences can be generated randomly that furthermore can be used, for e.g., for deriving the multilingual linguistic corpora.

We study these features to some extent in Section 3.3 and in Chapter 5 where the results from our developed grammar resources have been analyzed.

3.2 Medical Domain: Knowledge Representation

The domain knowledge representing the various types of concerns the user may pose to the system can be explicitly formalized in ontologies. The ontologies in comparison to some other semantic formalisms, e.g., database models and frames are richer and a flexible way of representing domain concepts. In most known formalisms they comprise a hierarchical representation where the terms are related usually in a “*is-a*” form. However this is not the only limiting form of relations, but can, beyond, contains other forms such as “*part-of*” relations overcoming the problem of under/over specifications phenomenon encompassing in most of the communication processes, and other relations for similarity measures dealing with synonyms. The ontologies also magnify the reusability of domain knowledge whereby making the human life easier while adapting the resources to other domains. Besides, the use of ontologies favors the integration of knowledge from widely used web services.

This section transmits the essence of domain ontologies representing the medical domain knowledge we have considered to work on. In particular, we have focused on a scenario where the user is looking for new medical specialists and can express his/her therapeutical circumstances to the system. We first study the user needs and till feasible dimensions we describe the domain-specific knowledge by the class **Medical_Concept** and its subclasses (as shown in Figure 3.2.1), and general knowledge about time and space is represented as the general concepts “**Unit_Of_Time**” and “**Space**” and their subclasses (as shown in Figure 3.2.2). The main goal of this separation is to favour the reusability of such general concepts across domains.

The domain-specific entities represented in the medical domain ontology are as follows:

- Medical concepts are subdivided into three categories: **Medical_Resources**, **Body_Part**, and **Disease**.

- The class **Body_Part** is included because in informal conversation the most common way of asking for a specialist or revealing the disease types can implicitly be given by indicating a specific part of body where the assistance is required.

- Including the user’s disease types by **Disease** class can be essential in case the user already knows his/her disease types rather expressing the body parts betimes.

- The class **Medical_Resources** has been subclassified into **Human_Resources** and **Equipment** subclasses. The **Equipment** class can have subclasses like **Hospital**, **Pharmacy**, **Clinics**, and other kinds of medical facilities where users can reach according to their exigency and preferences.

- **Human_Resources** can be described by subclasses representing the particular types of human experts involved (doctors, nurses, directors, secretaries, etc.). In our particular implementation we have focused on doctor specialists. We describe the main attributes related to the **Doctor** class that are involve in dialogues such as *visit_at_equipment* (system may answer the user needs by answering the equipment the doctor is supposed to visit at), *treat_for_body_part_of* (to express the concerned body part the doctor treats for), *visit_at_zone* and *visit_at_hour* (for their visiting schedule and location), *name* (name of the doctor).

This ontology has been constructed such that a number of user exemplary queries can be recognized:

- My ear hurts. Can I get a doctor, please?
- I prefer a small clinic.
- Is (are) there any cardiologist(s) near Barcelona centre?
- I am looking for an ophthalmologist.
- I am feeling severe pain in my elbow(s), which specialist should I look for?

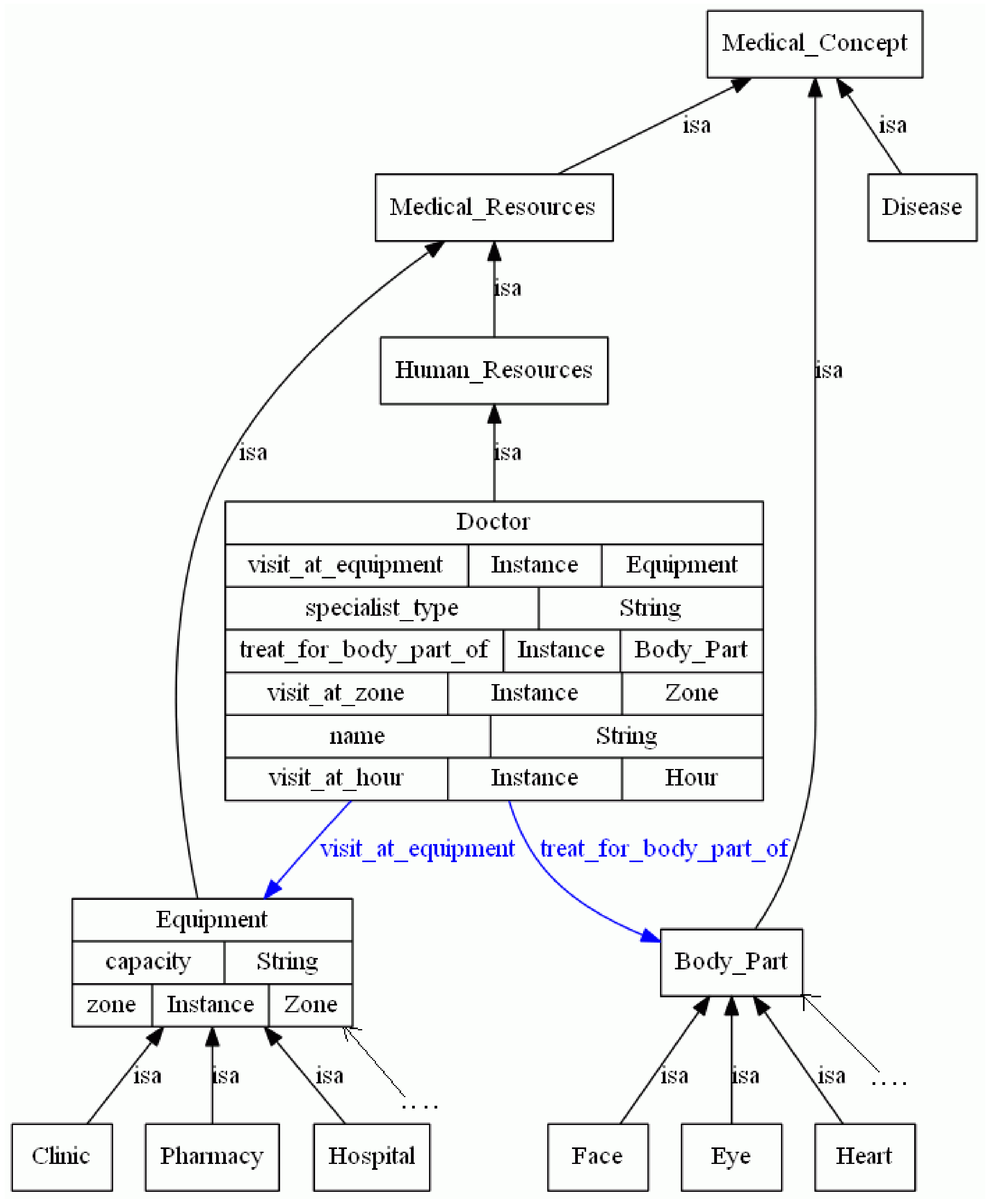


Figure 3.2.1: *Medical Domain Conceptual Knowledge Representation*

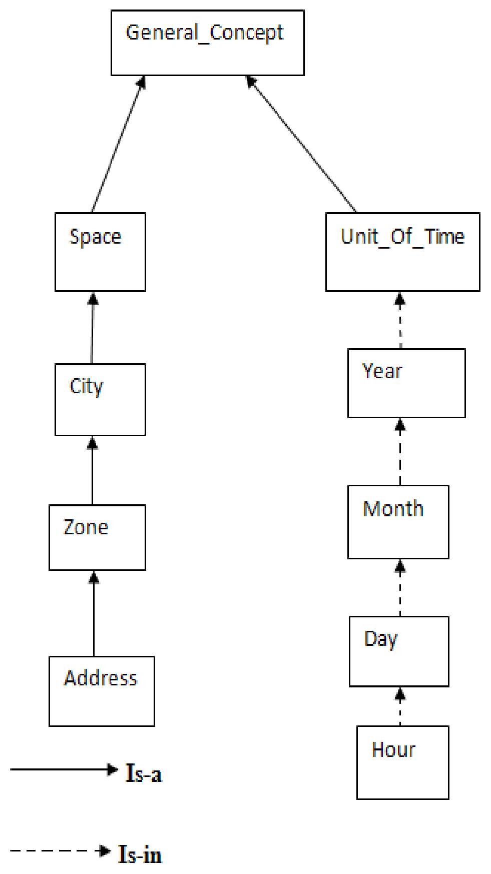


Figure 3.2.2: *General Conceptual Knowledge Representation*

3.3 Medical Domain: Developing the Grammar Rules using Grammatical Framework

In previous section we summarized the ontology representation for medical domain. This section details how we define the grammar rules for understanding the user needs during the communication process. The grammar rules are developed using the information represented in the ontology along with the grammatical framework. Generated grammar rules are incorporated into the natural language understanding (NLU) component of the system.

The generation of natural language answers to user queries will be described in Chapter 4 where we obtain the linguistic resources semi-automatically from domain ontologies using a syntactic-semantic taxonomy. For now let us emphasize particularly on how we developed the grammars for the NLU component.

Many practical rule-based NL interface systems make use of ontologies while writing the grammar rules in a particular application. This is because the domain-specific rules are implicitly organized in ontologies. Although the rules are written manually by using the conceptual knowledge represented in ontologies, reducing the human efforts required is one of the central goals of our work.

As time is a limited resource, we try to write the grammar rules in a structured way that makes them partially reusable in different domains. This is possible because of the modular architecture and the multiple inheritance property of GF. We now will go more in-depth to study the richer features of GF used in our implementation.

As said earlier we first analyzed the possible user needs when creating the medical ontology. We then included the domain entities therein to abstract grammar as in Figure 3.3.1.1. From these entities the semantic rules that can match a feasible number of user utterances are constructed. Though in abstract syntax the rules do not take into account any linguistic intimations about a particular language, those will however be developed in concrete syntax. Such modular architecture therefore indeed helps in development process of grammar rules later on in each language. We propose to deploy a partially reusable system that contains a clear separation of domain-specific knowledge from the general conceptual entities. Hence the grammar rules supporting those general knowledge entities are represented separately in general grammar, i.e. the **Time** and **Zone** grammars as in Figure 3.3.1.2 and can be inherited into domain-specific grammar.

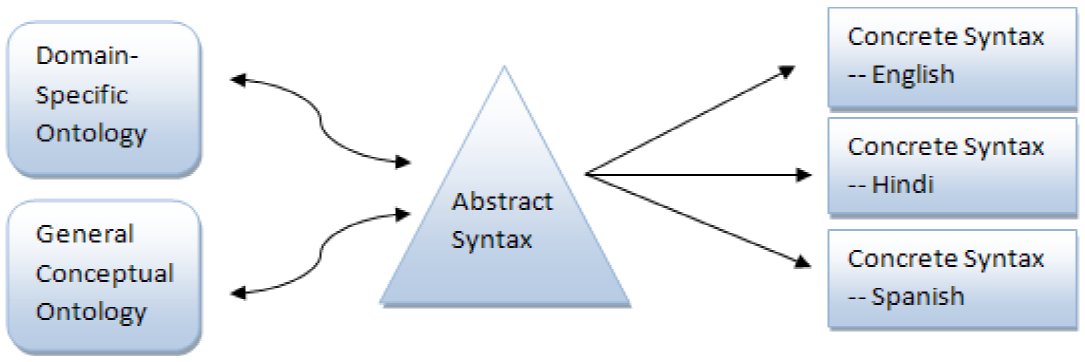

Domain-Specific Ontology
General Conceptual Ontology
Abstract Syntax
Concrete Syntax -- English
Concrete Syntax -- Hindi
Concrete Syntax -- Spanish

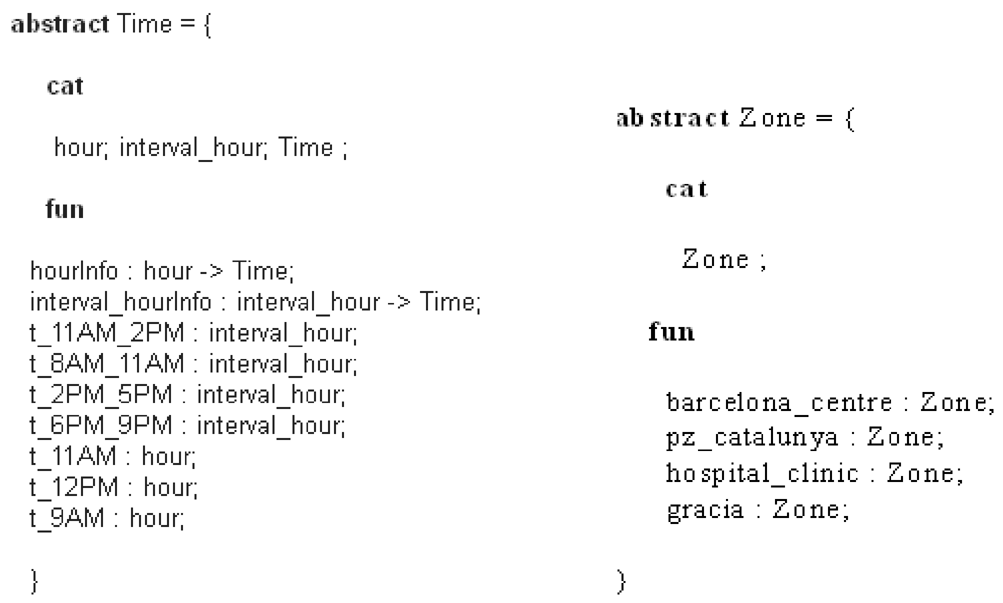
Figure 3.3.1 *Correspondence between Ontology and Grammars*

3.3.1 Abstract grammar for Health Domain

```
abstract HealthDomain = Zone, Time ** {

flags startcat = Comment;

cat

Body_Part;
Capacity;
Comment;
Description;
Description_1;
EquipCapacity;
Equipment;
Specialist;

fun

  userComment : Description -> Comment;
  userComment_specialistInquiry : Description_1 -> Comment;
  equipmentInquiry : EquipCapacity -> Equipment -> Description;
  illnessInfo : Body_Part -> Description;
  Specialist_with_Time : Description_1 -> Time -> Description;
  specialistInquiry : Specialist -> Zone -> Description_1;
  cardiologist : Specialist;
  dentist : Specialist;
  dermatologist : Specialist;
  ophthalmologist : Specialist;
  clinic : Equipment;
  hospital : Equipment;
  pharmacy : Equipment;
  equipCapacity : Capacity -> EquipCapacity;
  small : Capacity;
  big : Capacity;
```

```
   ear : Body_Part;
   elbow : Body_Part;
   head : Body_Part;
   eye : Body_Part;
   face : Body_Part;
   heart : Body_Part;
   skin : Body_Part;
   stomach : Body_Part;
   leg : Body_Part;
   teeth : Body_Part;
}
```

Figure 3.3.1.1 *The Domain-Specific Abstract grammar (for Health Domain)*



```
abstract Time = {

   cat

    hour; interval_hour; Time ;

   fun

  hourInfo : hour -> Time;
  interval_hourInfo : interval_hour -> Time;
  t_11AM_2PM : interval_hour;
  t_8AM_11AM : interval_hour;
  t_2PM_5PM : interval_hour;
  t_6PM_9PM : interval_hour;
  t_11AM : hour;
  t_12PM : hour;
  t_9AM : hour;

  }
```

```
abstract Zone = {

   cat

     Zone ;

   fun

    barcelona_centre : Zone;
    pz_catalunya : Zone;
    hospital_clinic : Zone;
    gracia : Zone;

}
```

Figure 3.3.1.2   *domain-independent Time and Zone abstract grammar*

In introductory Section 3.1, we explained that the abstract grammar only represents the semantics of the domain concepts and how they are formulated, it does not deal with language-specific details. In other words, the purpose of abstract syntax is to describe in a semantic form the most probable concepts the user may pose to the system. We therefore have used domain entities described in ontology and defined them in GF abstract syntax.

Correspondence between the domain ontology and the abstract grammar:

1. correspondence between cat (categories of abstract grammar) and domain ontology

Abstract categories correspond to the combination of lexical and syntactical categories.

- The lexical categories represent the concept classes, subclasses plus relevant attributes of those concepts in ontology.
- The syntactical categories are defined considering the combination of ontology attributes and concepts involved in users sentences and are then constructed from one or more lexical categories and/or syntactical categories in the right side of the function rules.

2. correspondence between fun (functions of abstract grammar) and domain ontology

This type of correspondence is maintained differently in the following two conditions:

- If there is only one category on right side of the function rule, it must be a lexical category. Given the lexical category is an attribute of any concept class in ontology, the function name associated with the rule corresponds to attribute's value. Given the lexical category is a concept class in ontology, the function name associated with the rule corresponds to its subclass or instance.
- If there is more than one category on right side of the rule, the abstract function name is just a keyword defining the function rule with that particular name.

In abstract grammar, we then have a list of lexical categories including Specialist, Equipment, Capacity, Body_Part, hour and interval_hour (categories inherited from Time grammar), and Zone (category inherited from Zone grammar), and a list of syntactical categories including EquipCapacity, Description, Description_1, Time (category inherited from Time grammar) and Comment.

The *Zone* and *Time* abstract grammars are inherited in domain-specific grammar using multiple inheritance property of GF language. This separation of grammars forces the domain experts to keep focus only on building grammar rules for the entities appearing in domain-specific ontologies, while the general grammars can be defined by a domain non-specialist from the general conceptual ontologies.

Here we give a thumbnail description about types of categories and their relations with ontology.

Body_Part	lexical category represents the concept Body_Part in ontology and the functions associated with this category express the subclasses of Body_Part (e.g. ear, heart)
Capacity	lexical category represents the attribute capacity of Equipment concept in ontology and the functions associated with this category express the values of attribute capacity (e.g. small, big)
Equipment	lexical category represents the concept Equipment in ontology and the functions associated with this category exprese the subclasses of Equipment (e.g. clinic, hospital)
EquipCapacity	syntactical category expresses the phrase describing the information related to lexical category: Capacity (e.g. small, big)
Description_1	syntactical category expresses the phrase describing the information related to lexical categories Specialist and Zone (e.g. "*I am looking for an eye specialist near plaza catalunya*")
Description	syntactical category expresses the phrase describing the information related to a syntactical category Description_1 and Time (i.e. information about Specialist, Zone and Time all in same phrase) and to lexical categories: Body_Part, and Equipment (e.g. "*I am looking for a cardiologist near gracia around 9AM*" "*My legs are hurting*" "*I prefer a small clinic*")
Comment	syntactical category expresses the complete phrase (i.e. any possible examples expressed by categories Description_1 and Description)

One or more functions (syntactic rules) can be defined for expressing the associated meaning of the rule in right side. For example, small and big are the functions expressing the attribute values of a lexical category Capacity and illnessInfo is the function name for the rule in its right side (since right side contains more than one category and at least one of them is a syntactical category, the function name, illnessInfo, is just a keyword). The function userComment constructs the syntactical category Comment which is set to be a start category (by flags startcat) to be used in parse tree. The defined syntactic rules are intended to capture the semantics of the user query and to form a parse tree.

To further understand the implementation of abstract syntax ideally, let us formalize a particular syntactic rule,

```
fun
        equipmentInquiry : EquipCapacity -> Equipment -> Description;
```

Where

- *equipmentInquiry* is the name a function that represents the user's query about the equipment and its capacity as in "is there a big hospital"

- *EquipCapacity* is a syntactical category constructed from the lexical category *Capacity* from the rule *equipCapacity : Capacity* -> *EquipCapacity*;

- *Equipment* is a lexical category constructed itself in each of associated functions *clinic*, *hospital*, and *pharmacy*

- *Description* is a syntactical category constructed from a lexical category *Equipment* and a syntactical category *EquipCapacity*.

3.3.2 Concrete grammars for Health Domain

We previously made an indispensable attempt to explain how the abstract grammar abstracts away all the language-specific details. The shared syntactic rules in abstract grammar are needed to represent the content of user's interventions. In this subsection we describe how they are related to the actual surface forms in a particular natural language. In particular, we describe our implementation of concrete grammars for health domain in Spanish, English and Hindi languages. The complete grammars are provided in Appendix.

The linguistic realizations or linearizations of the functions (**fun**) in abstract grammar are assigned to corresponding linearization functions (**lin**) in the concrete grammar. The linearization type (**lincat**) is assigned to each category (**cat**) in abstract grammar. We now will explain one rule and the fragments of grammars needed to show the syntax of the language. We will only give example sentences for the rest of the rules. The full grammars can however be found in Appendix.

3.3.2.1 <The rule 1>

The syntactic rule from the abstract syntax:

- fun *equipmentInquiry* : EquipCapacity -> Equipment -> Description -- [equation 1]

The concrete rules using the abstract rule in [equation 1] can be constructed by a linguistic expert, with the ambition of parsing a set of example sentences correctly. To the context of the following example sentences, the concrete grammar rules have been constructed in our implementation:

Context 1 Sentences corresponding to template format:

{i prefer/ i would like to see} a {small/big} {hospital/clinic/pharmacy}

- I prefer a small clinic

  The abstract/parse tree of above sentence would look like:
  => userComment (equipmentInquiry (equipCapacity small) clinic)

- I would like to see a big clinic
- I prefer a big pharmacy
- I would like to see a big hospital
- ….

Context 2 Sentences corresponding to template format:

is there {-/any/some} {small/big/-} {hospital/clinic/pharmacy}

- is there any small clinic
- is there some big pharmacy
- is there big hospital
- ……

Context 3 Sentences corresponding to template format:

are there {-/any/some} {small/big/-} {hospitals/clinics/pharmacies}

- are there any small hospitals
- are there any small pharmacies
- are there some small clinics
- are there some pharmacies
- are there clinics
- ……

Agreement type:

In all the sentences above, there is a verb (is/are) agreement with the equipment singularity or plurality, for instance, sentences starting with "is" will end up with singular equipment, like hospital/pharmacy, and sentences starting with "are" will end up, for instance, with pharmacies.

(A) Linearization rules and description in English concrete grammar

Linearization rule for the function (in [equation 1]) in English concrete grammar is

- lin *equipmentInquiry* eqc eq = description -- [equation 1.1a]

Where

```
description = {s = variants {give_equip_info_start ++ "a" ++ eqc.s ++ eq.s ! Sg;
 "is there" ++ var_equip ++ variants {[]; eqc.s} ++ eq.s ! Sg;
"are there" ++ var_equip ++ variants {[]; eqc.s} ++ eq.s ! Pl}}          -- [equation 1.2a]
```

In above equations, the *equipInquiry* is the name of the rule that corresponds to the function in abstract grammar [equation 1] and to the lin in concrete grammar. The eqc is EquipCapacity (it can be written with any short name for convenience, because the order can be recognized from function rule in abstract syntax), eq is Equipment, and description is Description. The above [equation 1.2a] is telling how a syntactical category description has been constructed from a lexical category eq (Equipment), and a syntactical category eqc (EquipCapacity). Along with these two categories, this description includes several GF constructions with variables and mechanisms tying them properly, and that should need the following brief explanations to serve our purpose for wider audience:

- The word variants allows any expression enclosed inside curly braces. Each expression must end in ‘;’. The use of ‘[]’ allows an empty expression.

- The symbol ‘++’ differs from ‘+’. The former adds the two strings with space between them while the latter adds the two strings without space, i.e. gluing them for suffix or prefix operations (we will see the use of ‘+’ symbol later in the grammar).

- Syntactic and semantic agreements are performed representing the categories as records and tables. They symbol ‘.’ is used to access a record field and the symbol ‘!’ is used to access a table value.

-

Now we detail the types of categories. The linearization types of categories EquipCapacity and Description are of record type with one field which has an object s of string type.

- lincat Description = {s : Str}
- lincat EquipCapacity = {s : Str}

The linearization type of category Equipment consists of a record type with two fields. The one field consists of an object s of table-type structure: {agreement_ param => Str}. This can be read as: *a table from agreement_param to String*, where agreement_ param is a parameter. The second field consists of an object b of type boolean, where boolean is a parameter.

- lincat   Equipment = {s : agreement_ param => Str; b : boolean}
- param   agreement_ param = Sg | Pl
- param   boolean = T | F

Each parameter has values to be used in linearization rules that can be selected using “!” as in [equation 1.2a], when using eq.s ! Sg, this indicates the rule will select the agreement_ param = Sg for Equipment. For example, it can be used to establish the singular and plural agreement of nouns with respect to determiners. For example, the given rule in [equation 1.1a] is assigned Equipment type using the outcome of other rules named clinic, hospital, or pharmacy (see full English grammar in Appendix).

- hospital = check_agreement_Equip "hospital" T

- pharmacy = check_agreement_Equip "pharmacy" F

For e.g., with clinic, we have an associated linearization rule:

- clinic = check_agreement_Equip "clinic" T   -- [equation 1.3a]

Since GF is a functional programming language, we can also define operations functions (oper) which allow a single function to be called for a number of agreements dealing with them. The check_agreement_Equip (operation function as in [equation 1.3a]), var_equip and give_equip_info_start in [equation 1.2a] are three operation definitions to be used for checking various kinds of agreements.

- oper   give_equip_info_start = variants {"i prefer"; "i would like to see"}

- oper   var_equip = variants {[]; "any"; "some"}

-
  ```
  oper  check_agreement_Equip : Str-> boolean -> {s : agreement_param => Str; b : boolean}
                  =  \x,y -> {s = table {
                                      Sg => x;
                                      Pl => case y of {
                                                   F => "pharmacies";
                                                   T => x + "s"
                                                    }
                                        };
                                 b = y};
  ```

This operation check_agreement_Equip has a type Str->Boolean->{s : agreement_param => Str; b : boolean}, and it can be read as "*an operation function named check_agreement_Equip reads the two object inputs of string type and boolean type, and outputs the record type object enclosed inside curly braces*". It will, when calling from clinic [equation 1.3a] which passes a string "clinic" and the value T of parameter boolean, equate the x = "clinic" and y = T, then the table checks if agreement_param associated with lincat Equipment is used either Sg or Pl. If it is Sg, the function returns x, in vice versa the function returns x adding the suffix "s" to it (if y = T) or returns pharmacies (if y = F). This was defined to agree on, for e.g., when y = T which is associated with clinic, the plural of "clinic" would become "clinics".

But as not every noun is to be suffixed “s”, special attentions are needed, for e.g., in the case of “pharmacy”, the parameter value associated is F, hence after calling the function, y = F and function returns not the “pharmacys” but “pharmacies” instead. This T and F are therefore just indicators assigned with a particular lexicon entry telling the rule how the categories should be appeared with morphological analysis and agreements.

(B) Linearization rules and description in Spanish concrete grammars

Same like in English where we needed a referential corpus of sentences, to the context of the following example sentences, the concrete grammar rules for Spanish have been constructed in our implementation:

Context Sentences corresponding to the following template formats:

{ { yo/-} prefiero / {me/-} gustaria } {a/-} un hospital {grande/pequno}

{ { yo/-} prefiero / {me/-} gustaria } {a/-} una {clinica/farmacia} {grande/pequna}

hay algun hospital {grande(s)/pequno(s)}

hay algunas {clinicas/farmacias} {grandes/(pequnas}

hay algunos hospitales {grandes/pequnos}

- ➢ yo prefiero a un hospital pequno
- ➢ prefiero a una clinica pequna
- ➢ gustaria una farmacia grande
- ➢ hay algunas clinicas pequnas
- ➢ hay algunos hospitales grandes
- ➢ ………

In all sentences above there is a determiner type - noun singularity/plurality – adjective feminine/masculine agreement.

Linearization rule for the function in [equation 1] in Spanish concrete grammar is

- lin *equipmentInquiry* eqc eq = description -- [equation 1.1b]

Where

```
description = {s = variants {

  give_equip_info_start  ++ variants{"a";[]} ++ eq.s!Sg1 ++ eqc.s ! eq.b1;
"hay" ++ eq.s ! variants {Sg; Pl} ++ eqc.s ! eq.b1

                    }

  };                                      -- [equation 1.2b]
```

The linearization type of category Description is of record type with one field which has an object s of String type.

- lincat Description = {s : Str}

The linearization type of category Equipment is of record type with two fields. The one field has an object s, which is of table-type structure: {agreement_param => Str}, this can be read as: *a table from agreement_param to String*, where agreement_param is a parameter. The other field has an object b1 of type boolean1, where boolean1 is a parameter. The linearization type of EquipCapacity is of record type with single field which has an object s of a table-type structure: {boolean1 => Str}, reading: *a table from boolean1 to String*.

- lincat Equipment = {s : agreement_param => Str; b1 : boolean1}
- lincat EquipCapacity = {s : boolean1 => Str}
- param agreement_param = Sg | Pl | Sg1
- param boolean1 = FEM | MASC

These parameters, in same fashion as in English grammar, are used to deal with agreements needed in Spanish. For example, clinica is a feminine noun in Spanish. Then to use it as a plural or singular, we define an operation check_agreement_Equip [equation 1.4b] which will, when calling from [equation 1.3b], assign y = FEM and x = "clinica", then the operation checks whether agreement_param is set to Sg, Pl or Sg1. Depending on that it adds suffix, prefixes and perform other morphological operations defined in this particular operation. For example, in [equation 1.2b] we set eq.s ! Sg1 ++ eqc.s ! eq.b1, this indicates that we want an equipment that fixes agreement_param to Sg1, and in equipment capacity (eqc) the boolean1 type from equipment is used (eq.b1). So if clinica has been used for equipment, the associated boolean1 type is FEM in lincat, so the output of operation check_agreement_Equip would be: "una clinica".

- lincat clinic = check_agreement_Equip "clinica" FEM   -- [equation 1.3b]
- lincat hospital = check_agreement_Equip "hospital" MASC
- lincat pharmacy = check_agreement_Equip "farmacia" FEM

```
•   oper  check_agreement_Equip : Str -> boolean1 -> {s : agreement_param => Str; b1 : boolean1}  =
                                   \x,y -> {s = table {

                                     Sg1 => case y of {

                                            FEM => "una" ++ x;

                                           MASC => "un" ++ x

                                                };

                                     Pl => case y of {

                                          FEM => "algunas" ++ x + "s";

                                          MASC => "algunos" ++ x + "es"

                                                  };

                                      Sg => case y of {

                                           FEM => "alguna" ++ x;

                                           MASC => "algun" ++ x

                                               }

                                   };

                            b1 = y};                                         -- [equation 1.4b]

•   oper   give_equip_info_start  = variants {variants {"yo"; []} ++ "prefiero";

                                                 variants {"me"; []} ++ "gustaria"};
```

(C) Linearization rules and description in Hindi concrete grammars

To the context of the following example sentences, the concrete grammar rules for Hindi have been constructed in our implementation:

Context Sentences corresponding to the following template formats:

मैं एक {बड़ा / छोटा} हॉस्पिटल {पसंद / मैं जाना पसंद} {करूँगा/ करुँगी} है

मैं एक {बड़ी / छोटी} {फार्मेसी / क्लीनिक } {पसंद / मैं जाना पसंद} {करूँगा/ करुँगी} है

क्या वाहा कोई {बड़ा / छोटा} {हॉस्पिटल्स / हॉस्पिटल} है

क्या वाहा कोई {बड़ी / छोटी} {फार्मसिया / फार्मेसी / क्लीनिक } है

The exemplary queries:

- मैं एक बड़ा हॉस्पिटल मैं जाना पसंद करूँगा
- मैं एक छोटी फार्मेसी पसंद करुँगी
- क्या वाहा कोई बड़ी फार्मसिया है
- ……………

All sentences have noun singularity/plurality with adjective feminine/masculine agreements.

Linearization rule for the function (in [equation 1]) in Hindi concrete grammar is:

- lin *equipmentInquiry* eqc eq = description

Where

```
description = {s = variants {"मैं"  ++ "एक" ++ eqc.s ! eq.b1 ++ eq.s ! Sg

          ++ variants {"पसंद"; "मैं जाना पसंद"} ++ variants {"करूँगा"; "करुँगी"};

            "क्या  वाहा  कोई"++ eqc.s ! eq.b1 ++ eq.s ! variants{Sg; Pl} ++ "है"}}
```

The linearization types of categories Description, Equipment, EquipCapacity are same like in Spanish grammar.

- lincat Description = {s : Str}
- lincat Equipment = {s : agreement_param => Str; b1 : boolean1}
- lincat EquipCapacity = {s : boolean1 => Str}

However, the agreement_param parameter has only two values defined in Hindi grammar. In Spanish grammar it contained three values instead. We defined this because of the different grammatical structure in language.

- param   agreement_param = Sg | Pl
- param   boolean1 = FEM | MASC

Unlike the Spanish and English grammar we have not defined any operations for checking agreement for Equipment, we rather provide all the necessary structure inside each lincat associated with Equipment. For e.g.,

```
clinic  = {s = table {
                    Sg => variants {"क्लीनिक"; "क्लिनिक"};
                    Pl => variants {"क्लीनिक"; "क्लिनिक"}
                         };
                b1 = MASC}



hospital  = {s = table {
                    Sg => "हॉस्पिटल";
                    Pl => "हॉस्पिटल्स"
                       };
                b1 = MASC}



pharmacy  = {s = table {
                    Sg => "फार्मेसी";
                    Pl => "फार्मेसिया"
                      };
          b1 = FEM};
```

In above lines of code we explicitly set the clinic and hospital to be masculine nouns, and pharmacy to be a feminine noun in Hindi. And we express how each of them varies in singular (Sg) and plural (Pl) case. In Spanish and English we did that using the operation definitions after passing the parameters. That helps if we have many rules performing the same kinds of morphological analysis, for e.g., if many words are needed to be suffixed by “s”, calling them from operations would cost less. But in Hindi, for this particular rule, the morphological analysis is very different in each of the words (hospital, pharmacy, and clinic). So we better express their variability in lincat definitions. However, in many of the other rules in Hindi, as in defined complete Hindi grammar in Appendix, the operation definitions have been used to develop the grammar at lower cost.

3.3.2.2 <The rule 2>

The syntactic rule from the abstract syntax:

- fun illnessInfo : Body_Part -> Description

The linearization rule corresponding to the above syntactic rule in concrete grammars enable the system recognizes some of the following sentences:

Spanish	**English**	**Hindi**
me duele la cara	My face hurts/suffers My face is hurting/suffering	मेरा चेहरा दुख रहा है
me duele el ojo	My eye hurts/suffers My eye is hurting/suffering	मेरी आंख दुख रही है
me duelen los ojos	My eyes hurt/suffer My eyes are hurting/suffering	मेरी आँखे दुख रही है
me duele la cabeza	My head hurts/suffers My head is hurting/suffering	मेरा सिर दुख रहा है मेरा मस्तिष्क दुख रहा है मेरा मस्तक दुख रहा है
me duelen las piernas	My legs are hurting/suffering My legs hurt/suffer	मेरे पैर दुख रहे है
me duelen los codos	My elbows hurt/suffer My elbows are hurting/suffering	मेरी कोहनी दुख रही है

Slight changes were needed for representing agreements when translating rule illness from English to Spanish. This is because in Spanish number agreement concerns on verb-determiner-noun altogether, whereas in English only verb and noun agreement was necessary for this particular rule. For example, in the sentence "*my eye hurts*" the verb form will be "hurt" or "hurts" depending if the noun is eyes or eye. In Spanish in the sentence "*me duele el ojo*" there is an agreement between verb duele or duelen, the determiner whether el or los, and the noun whether ojo or ojos. Feminine and masculine of singular noun make determiners to be la and el, respectively. Such a triple form of agreements makes Spanish grammar different than the English one.

3.3.2.3 <The rule 3>

- fun specialistInquiry : Specialist -> Zone -> Description_1

The following are the sentences to be recognized by the linearization rule corresponding to above function rule:

Spanish	English	Hindi
Estoy buscando/busco a un cardiólogo cerca de plaza catalunya	I am looking for a cardiologist near plaza catalunya	क्या आप मेरे लिए एक हृदयरोग विशेषज्ञ की तलाश कर सकते हो क्या प्लाजा कातालुन्या के आस पास
Busco una dentista en gracia	I need a dentist in gracia	मुझे एक दंत विशेषज्ञ की तलाश कर सकते हो क्या ग्रासिया के आस पास
....	....	....

3.3.2.4 <The rule 4>

- fun Specialist_with_Time : Description_1 -> Time -> Description

This rule lengthen the previously described rule, <The rule 3>, with the addition of the Time category. Therefore any sentence recognized by <The rule 3> plus the information about Time will be recognized via this rule. A possible sentence recognized by the concrete English grammar rule following above function rule as base is: "*I need a cardiologist close to gracia at 9AM*".

3.3.2.5 <The rule 5 and 6>

- fun userComment : Description -> Comment

This function rule can recognize any sentence recognized with a syntactical category Description, and construct a new syntactical category Comment that is set to be a start category to appear in parse abstract tree.

3.3.2.6 <**Rest of the rules**>

Rest of the syntactic function rules have only one category on right side, i.e they do not combine more categories to manipulate the semantics, but they still are called rules and are used for constructing the other rules previously described.

3.4 Cultural Events Domain: Knowledge Representation

The second scenario we considered where the user wants to consult the information on the cultural events that take place in the city. The users may ask for specific types of events by giving additional information such as event venue, schedule, and location. We represented this knowledge as domain concepts, their attributes and relations among them in doman ontology. Although our focus was restricted to a few of concepts involved in Figure 3.4.1, this conceptual model can be further extended.

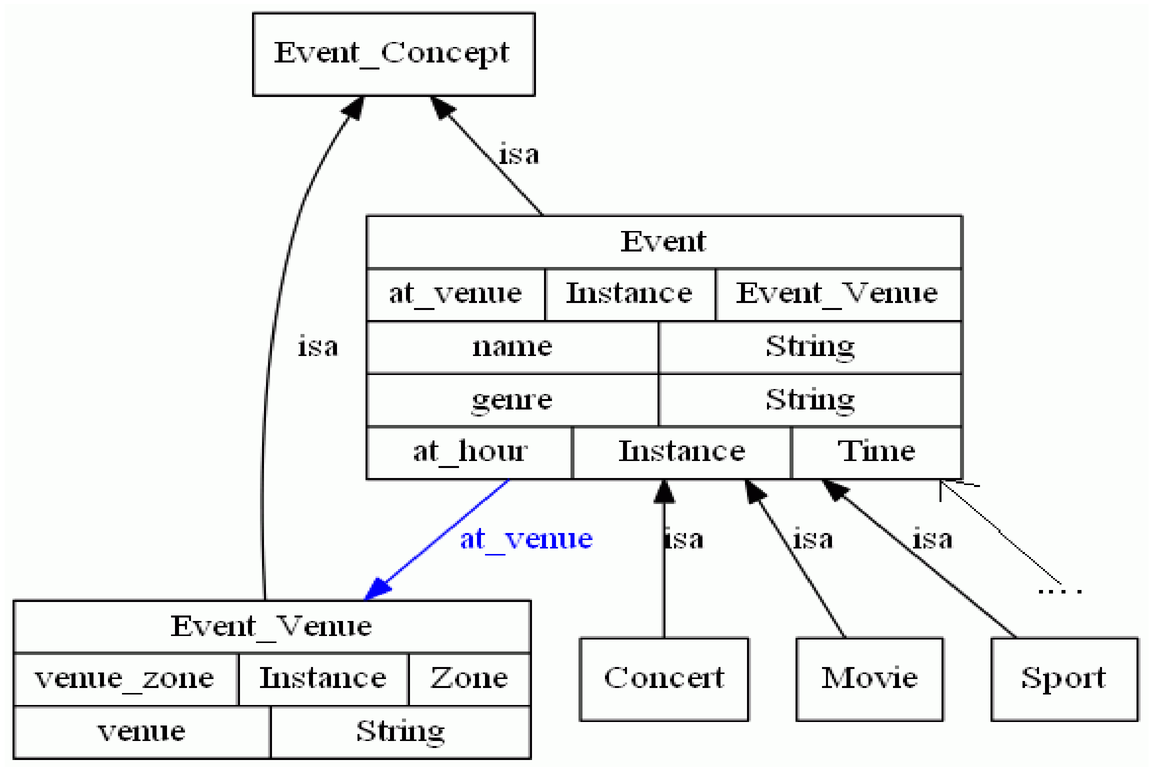


Figure 3.4.1 *Cultural Events Domain Knowledge Representation*

The domain-specific knowledge is described by the concept **Event_Concept** and its subclasses). To represent the general knowledge about time and space we reuse the same general concepts “**Unit_Of_Time**” and “**Space**” and their subclasses from the previously defined knowledge in Section 3.2.

Event_Concept is further subclassified into two concepts **Event** and **Event_Venue**. The concept Event is described by a set of attributes name (to describe the name of the event), *at_hour* (event schedule), *genre* (genre of the event), *at_venue* (venue information about the event).

Event_Venue is described by a set of attributes *venue* (venue name of event), *venue_zone* (venue zone of event). Next, the Event concept can have a particular type subclassified as Movie, Concert, and Sport.

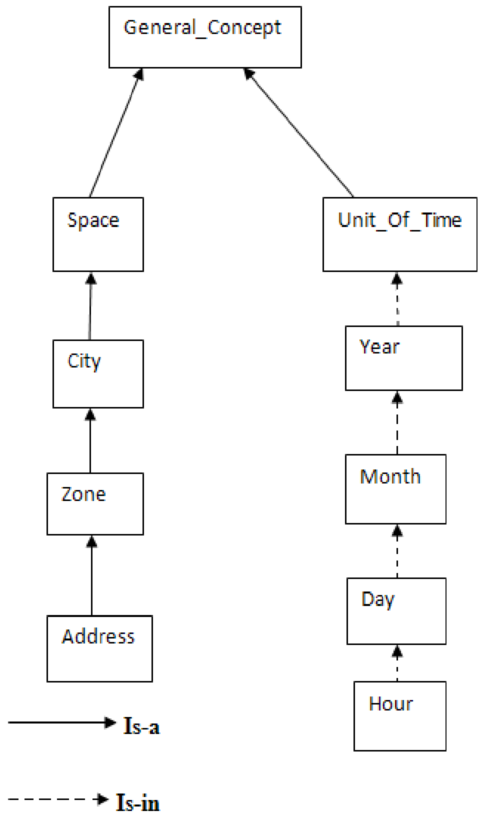


Figure 3.4.2 *domain-independent Time and Zone knowledge representation*

3.5 Cultural Events Domain: Developing the Grammar Rules using Grammatical Framework

A thorough explanation about correspondence between ontology and abstract grammar was already given in introductory part of Section 3.3. We now can directly describe the abstract grammar in cultural events domain.

3.5.1 Abstract Grammar

```
abstract EventDomain = {

flags startcat = Comment;

cat

Comment;
Description;
Event;
Event_Info;
Genre;
Venue;

fun
  userComment : Description -> Comment;
  event_zone_time : Event_Info -> Zone -> Time -> Description;
  event_venue_time : Event_Info -> Venue -> Time -> Description;
  genre_event : Genre -> Event -> Event_Info;
  movie : Event;
  concert : Event;
  sport : Event;
  dramatic : Genre;
  musical : Genre;
  romantic : Genre;
  orchestic : Genre;
  auditori : Venue;
  bcn_centre_hall : Venue;
  centre_public_ground : Venue;
  city_hall : Venue;
  royal_play_ground : Venue;

}
```

Figure 3.5.1.1 *The Domain-Specific Abstract grammar (for Cultural Events Domain)*

We have a list of lexical categories including Event, Genre, Venue, hour and interval_hour (categories inherited from Time grammar), and Zone (category inherited from Zone grammar), and a list of syntactical categories including Event_Info, Description, Time (category inherited from Time grammar) and Comment.

Here we give a thumbnail description about types of categories and their relations with ontology.

Event	lexical category represents the concept Event in ontology and the functions associated with this category express the subclasses of Event (e.g. movie, concert)
Genre	lexical category represents the attribute genre of concept Event in ontology and the functions associated with this category express the values of this attribute (e.g. "musical", "romantic")
Venue	lexical category represents the attribute venue of concept Event_Venue in ontology and the functions associated with this category express the values of this attribute (e.g. city_hall, auditori)
Event_Info	Syntactical category expresses the phrase describing the information related to *Event* and its *Genre*. e.g. "*I want to see a musical concert*"
Description	Syntactical category expresses the phrase describing the information related to functions event_zone_time (*Event_Info,* its Zone and Time) and event_venue_time (Event_Info, its Venue and Time). e.g. "*I want to see a romantic movie that take place at city hall near Barcelona centre*"
Comment	syntactical category expresses the complete phrase (i.e. any possible examples expressed by category: Description)

The functions (syntactic rules) are named to assign a meaning to each rule in right side. For example, musical and romantic are the functions expressing the attribute values of a lexical category Genre. The genre_event is a function name for the rule in its right side. Since right side of this rule contains more than one category and at least one of them is a syntactical category, the function name, genre_event, is just a keyword. The function userComment constructs the syntactical category Comment which is set to be a start category (by flags startcat) to be used in parse tree. The defined syntactic rules are intended to capture the semantics of the user query and to form a parse tree.

3.5.2 Concrete grammar cultural events domain

We have already detailed the GF features facilitating the development of the concrete syntaxes in all three languages in Medical domain. In cultural domain we developed the concrete rules only on smaller range. However, the purpose of using two different domains was to reuse the knowledge and fulfill the aim of constructing the rules in a very organized way what can set a base to adaption to different languages and domains. The implemented concrete grammars for cultural domain are in Appendix.

3.6 Summary of Grammar Development

In this chapter we have worked on the use of domain ontologies for grammar development in several languages. Our work is focused on the generation of grammar rules in three different languages: English, Spanish, and Hindi. The differences in linguistic structure of these three languages are considerable, especially in case of the Hindi language, that it also uses a different alphabet. However GF reduces the cost of development the grammar rules for each language by separating the abstract grammar (conceptual) from the concrete syntax. Because the content of user's interventions is the same for all three languages, they share the same abstract grammar rules. Then, the GF formalism facilitates the generation of concrete syntax rules for each language from abstract rules. Additionally, GF formalism supports richer features which facilitate the conceptual and syntactic agreements between the constituents (categories and words) in concrete rules that are different in all three languages. It also facilitates the representation of different orders of rules constituent orders in different languages.

Chapter 4

Natural language Generation

Most practical communication systems use simple natural language generation (NLG) component of NL systems. When developing NL interfaces usually efforts are focused on NL understanding modules. Main reason is that simple NLG modules using general templates or hand-crafted messages may have good results and also because complex modules using discourse planner and surface generators can be expensive to generate. There have been several relevant works on defining general mechanisms to generate NL sentences for any domain. Several of those works are based on empirical methods while other are based on the use of conceptual knowledge resources, such as ontologies. The work we present in this section belongs to the second group. We have followed a similar approach to that proposed by (Bateman, 1994).

We have used a syntactic-semantic taxonomy of conceptual attributes previously defined in (Gatius, 2001). As we explained in Chapter 2, in Section 2.3 the purpose of this taxonomy is to perform an interface between the conceptual knowledge in the ontology and the linguistic information appearing in the grammars. We adapted the taxonomy to the medical domain to be used in generating the answers in different languages. We have related the attributes of the main concepts: Doctor, as shown in Figure 4.1 (note that for simplicity we started working only with the main domain concepts, we could relate any conceptual attribute in the domain ontology to this taxonomy). Each of the attributes describing the concept Doctor is associated with one or more general classes in the syntactico-semantic taxonomy. Each class in this taxonomy expresses the basic operations of filling and consulting conceptual attributes. For this purpose three new basic attribute classes are defined.

The attribute name is related to a new class, OF_DESIGNATION that describes the doctor name. It is a subclass of the class OF_NAME and it has been created because all doctors are designed with the same title: Doctor/Dr/Specialist. The attribute *visit_at_equipment* describes the action of visiting and the specific equipment where this action takes place. For this reason, it is related to the syntactico-semantic taxonomy by combining the basic class IS with the basic class OF_PLACE. The attribute *visit_at_zone* is related to a combination of two classes: the basic class IS with the basic class OF_TIME.

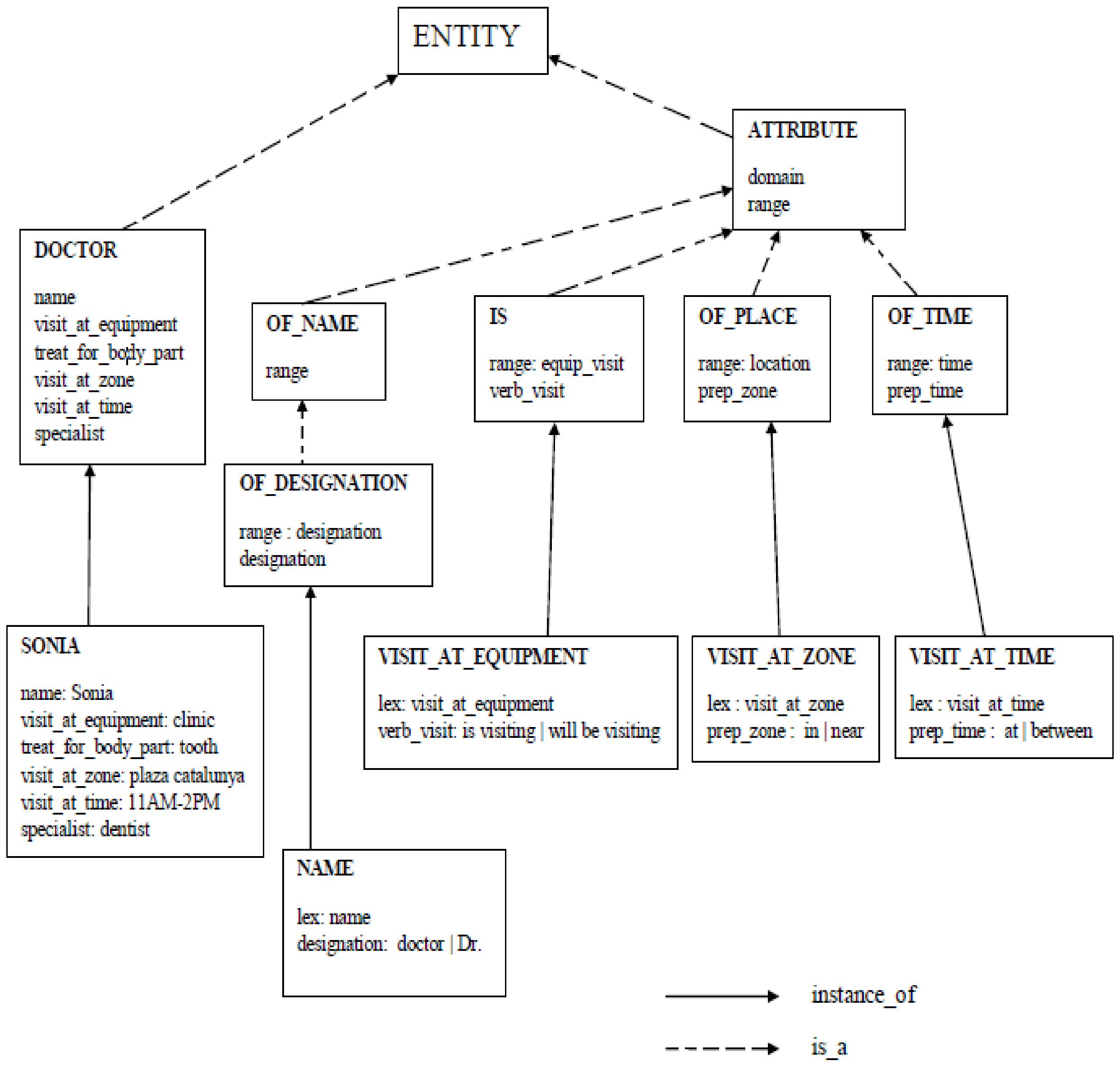


Figure 4.1 *Syntactic-Semantic taxonomy adapted to medical domain*

The attribute classes used would be same for all the languages, or can have some modifications depending on how a particular language uses those attributes. The Figure currently shows the instances of attribute classes to generate messages in English. Adapting it to another language just require the changes in lexical entries. For e.g., the attribute VISIT_AT_ZONE is related to the attribute class OF_PLACE that has an associated lexical entry:

- prep_zone : in | near

Adapting it to Hindi would need to define how the attribute can be expressed in Hindi; the following is the change in associated lexical entry:

- prep_zone : के पास

Using this taxonomy and thus the patterns associated with classes allows the automatic generation of system's messages at system developing time in different languages. Then, we have to select and do minor corrections manually. The cost of this process is less than what would be needed to write manually the sentences from scratch. More complex patterns combining the different attribute classes could also be used.

Chapter 5

Grammar Evaluations and Experimental Examples

In Chapter 3 we described the generation of grammar rules in there different languages: Spanish, Hindi, and English for two different domain scenarios: health domain and cultural events domain. We focused our implementation mostly on health domain, but fulfilled our objective of using a different domain, the cultural events domain, and to reuse grammar components across both domains. In Section 5.1 in this chapter we described some metrics used to measure the reusability of our implemented grammar rules at different level of organization. Section 5.2 provides some of the interesting applications using our developed multilingual grammars.

5.1 Grammar Evaluations

In grammar based NL interfaces, evaluating the results of grammars is quite hard since major goal is to cover as many possible user sentences in different variations. Although building a large set of grammar rules can cover many possible user queries, there still will be the problem of not supporting all possible user interventions (new words, informal expressions, mistakes, etc). These can however be solved to some extent by grammars accepting a few informal words, user mistakes etc., they are still not enough. This is why the grammar based systems often result a poor efficiency.

There can have many other metrics to evaluate the grammar for several purposes. Since our goal has not been to develop a complete grammar for each language and domain but well structured grammars, easy to extend and adapt (to other domains, other types of users, other languages). We measure a degree of reusability of grammar syntaxes at different level of organizations (reusability at modular level, at syntactical level, at lexical level) considering different coverage range of evaluation (across domains and across each language pairs) and different types of grammar (concrete and abstract type).

5.1.1 Reusability at modular level

The developed resources have a total of 14 distinct modules. The distribution is the following:

- Number of domain-specific abstract grammar components is 2 (1 for each domain)
- Number of domain-independent abstract grammar components is 2 (representing Zone and Time grammar)
- Number of domain-specific concrete grammar components is 6 (3 for all three languages in health domain, other three for all three languages in cultural event domain)
- Number of domain-independent concrete grammar components is 4 (3 representing Zone abstract grammar separately in all three languages, 1 representing Time abstract grammar shared by all three language)

Metrics calculated are the following:

	Across domains (health-event)	**Across languages (Eng-Spa)**	**Across languages (Eng-Hin)**	**Across languages (Spa-Hin)**
Abstract grammar reusability	50%	NA	NA	NA
Concrete grammar reusability	40%	20% (health domain) 20% (event domain)	20% (health domain) 20% (event domain)	20% (health domain) 20% (event domain)

Explanation <1> Metric: abstract grammar reusability across domain: 50%

By this metric we calculate the reusability ratio of abstract grammar between health domain and event domain.

We have 2 domain-specific abstract grammars (1 each for health domain and event domain). Both are not reusable across domains. We also have 2 domain-independent (Time and Zone abstract grammars) shared by both domains. Thus we have a total of 4 distinct abstract grammars, where 2 domain-independent abstract grammars are reusable between both domains. Henceforth reusability noted is 50%.

Explanation <2> Metric: concrete grammar reusability across domain: 40%

We have 6 domain-specific concrete grammars: 1 for each language in health domain and 1 for each language in event domain. Apart from this we have 3 domain-independent Zone concrete grammars (one for each language, common in both domains) and 1 domain-independent Time concrete grammar (common in all three languages as well as in both domains). Thus total distinct concrete grammars are 10. Out of them 4 are reusable across both domains (3 Zone grammars and 1 Time grammar). Henceforth reusability noted is 40%.

Explanation <3> Metric: concrete grammar reusability across languages (Eng-Spa): 20% (health domain)

By this metric we calculate the percentage of concrete grammar modules that are reusable between English and Spanish languages.

In health domain, we have total 2 domain-specific concrete grammars one for each language (English and Spanish), and 3 domain-independent concrete grammars (2 Zone concrete grammars one for each language, 1 Time concrete grammar common to both languages). Thus we have a total of 5 distinct grammar modules. Among them total modules reusable in English and Spanish is only 1 (Time grammar), therefore reusability is 20%.

Explanation <rest of metrics>

The same explanation as of Explanation 3 goes for other metrics at modular level (considering different domain and language pairs)

The keyword "NA" implies the measure for that particular grammar type with particular coverage range may not be performed (since the abstract grammars are defined only for domain knowledge, and are the base for generating the concrete grammar in all the languages).

5.1.2 Reusability at syntactical level

On syntactical level we exclude the syntactic function rules that come with only one lexical category at right side, those are called lexical rules.

	Across domains (health-event)	**Across languages (Eng-Spa)**	**Across languages (Eng-Hin)**	**Across languages (Spa-Hin)**
Abstract grammar reusability	25%	NA	NA	NA
Concrete grammar reusability	12%	50% (health domain) 33.3% (event domain)	38.5% (health domain) 33.3% (event domain)	38.5% (health domain) 33.3% (event domain)

Explanation 1> Metric: abstract grammar reusability across domains: 25%

We have a total of 9 syntactic rules in abstract grammars of health domain: 7 in domain specific grammar and 2 in domain-independent Time abstract grammar (Zone grammar does not have any syntactic rules, they have lexical rules instead).

We have a total of 6 syntactic rules in abstract grammars of event domain: 4 in domain-specific grammar and 2 in domain-independent Time grammar (Zone grammar does not have any syntactic rules).

With this distribution, we have a total number of 12 distinct rules (6 distinct from domain-specific abstract grammar of health domain, 3 distinct from domain-specific of event domain, 2 from domain independent Time grammar (common in both domains), and 1 common both in event and health abstract grammar). Reusable grammar modules are 3 out of 12. Henceforth reusability is 25%

Explanation 2> Metric: concrete grammar reusability across domain: 12%

We have a total of 7 syntactic rules in domain-specific concrete grammar of health domain in each language. Among them 3 rules are common in all three languages (let us call it by R1=3), and the 3 rules are distinct in overall language pairs (R2=3), one rule is common in English-Spanish, but not in pair consisting Hindi (R3=1+1=2; the first 1 common to both English and Spanish and the second one for Hindi).

We have a total of 4 syntactic rules in domain-specific concrete grammar of event domain in each language. Among them 1 rule is common in all three languages (R4=1), and rest of the 3 rules are distinct in overall language pairs (R5=3).

Domain-independent concrete Time grammar syntactic rules are 2 (all are common to both domains) (let us say R6=2). The Zone grammar does not contain any syntactic rule.

With this distribution, we have a total number of 25 distinct rules: 2 distinct from R1, 0 distinct from R4, 1 common in R1 and R4 (this rule is shared between domains), 9 distinct from R2*3 (considering the three different languages), 9 distinct from R5*3 (again, considering the three different languages), 2 distinct from R3, 2 distinct from R6 (these rules are common to both domains). Reusable grammar rules across domains are 3 out of 25. Henceforth reusability is 12%

Explanation 3> Metric: concrete grammar reusability across languages (Eng-Spa): 50% (health domain)

We have a total of 7 syntactic rules in domain-specific concrete grammar of health domain in each language (English and Spanish). Among them 4 rules are common in both languages, and rest of the 3 rules are distinct in each language (hence 6 distinct rules considering both languages).

Domain-independent concrete Time grammar rules are 2 (common in both languages). The Zone grammar does not contain any syntactic rule.

With this distribution, reusable grammar rules are 6 out of 12. Henceforth reusability is 50%

Here we can notice that the most sharing is because of the general conceptual knowledge, especially the Time grammars, where the syntactic rules do not combine any linguistic information but the categories only in upper rules. And the shared rules in domain-specific grammars are because there are not any lexicons/words present, but the rules are just combining categories with appropriate access to categories' values from the lower lexical rules that construct them. For e.g., one syntactic rule that is common both in English and Spanish is:

- **lin** userComment_specialistInquiry d1 = {s = d1.s}

In right side of the rule, the information is common to both languages.

Explanation <rest of metrics>

The same explanation goes for other metrics at syntactical level.

5.1.3 Reusability at lexical level

	Across domains (health-event)	Across languages (Eng-Spa)	Across languages (Eng-Hin)	Across languages (Spa-Hin)
Abstract grammar reusability	26.2%	NA	NA	NA
Concrete grammar reusability	14.7%	20% (health domain) 27.8% (event domain)	13.2% (health domain) 17.9% (event domain)	13.2% (health domain) 17.9% (event domain)

Explanation 1> Metric: abstract grammar reusability across domains: 26.2%

We have a total of 30 lexical rules in abstract grammars of health domain: 19 in domain specific grammar and 7 in domain-independent Time grammar, 4 in domain-independent Zone grammar.

We have a total of 23 lexical rules in abstract grammars of event domain: 12 in domain-specific grammar and 7 in domain-independent Time grammar, 4 in domain-independent Zone grammar.

With this distribution, we have a total number of 42 distinct rules (19 distinct from domain-specific abstract grammar of health domain, 12 distinct from domain-specific of event domain, 11 distinct from domain independent Time plus Zone grammar (common in both domains). Reusable grammar modules across both domains are 11 out of 42. Henceforth reusability is 26.2%

Explanation 2> Metric: concrete grammar reusability across domain: 14.7%

We have a total of 19 lexical rules in domain-specific concrete grammar of health domain in each language. None of them are common to any language. So domain specific distinct rules considering all three languages are 19*3= 57

We have a total of 12 lexical rules in domain-specific concrete grammar of event domain in each language. None of them are common to any language. So domain specific distinct rules considering all three languages are 12*3= 36

Domain-independent concrete Time grammar lexical rules are 7 (all are common to both domains and all three languages). Domain-independent concrete Zone grammar distinct lexical rules are 9 (4 in Zone Hindi concrete grammar, 1 distinct in Zone English grammar, 1 distinct in Zone Spanish grammar, 3 common in both Spanish and English). Each of these 9 distinct lexical rules from Zone grammar is common to both domains

With this distribution, we have a total number of 57+36+7+9= 109 distinct lexical rules. Reusable lexical grammar rules across domains are 16 out of 109. Henceforth reusability is 14.7%

Explanation 3> Metric: concrete grammar reusability across languages (Eng-Spa): 20% (health domain)

We have a total of 19 lexical rules in domain-specific concrete grammar of health domain in each language (English and Spanish). None of them are common to any language. So domain specific distinct lexical rules considering both languages are 19*2= 38

Domain-independent concrete Time grammar lexical rules are 7 (each of them common both languages). Domain-independent concrete Zone grammar distinct lexical rules are 5 (1 distinct in Zone English grammar, 1 distinct in Zone Spanish grammar, 3 common in both Spanish and English Zone grammar).

With this distribution, we have a total of 38+7+5= 50 distinct lexical rules. Reusable lexical grammar rules between English and Spanish are 10 out of 50. Henceforth reusability is 20%

Explanation <rest of metrics>

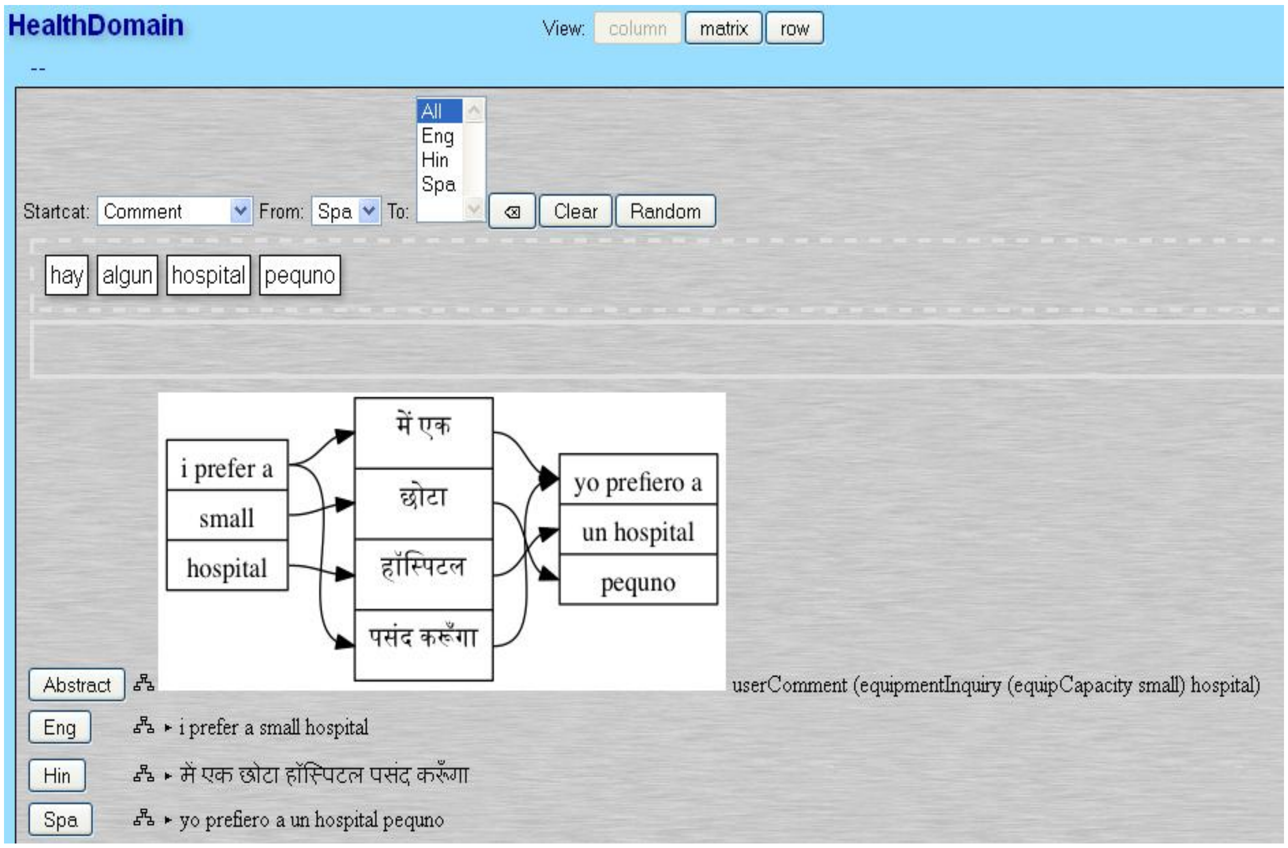
The same explanation goes for other metric at lexical level.

We can see the reusability ratios in a pair consisting of Hindi language for lexical rules are lower in number than Eng-Spa pair. In Eng-Spa pair reusability for lexical rules is because they have same alphabets in some lexicon values (for e.g. "*plaza catalunya*" is same lexicon for Spanish and English). Some of the reusable lexicons in Hindi are also present, but those are just because of the shared Time grammar that is expressed with numbers in Hindi too.

5.2 Generating the parsed abstract tree and translating the query into all the other languages within multilingual developed resources

An interesting functionality of GF consists of showing the parsed tree of a natural language sentence recognized by a specific grammar. In the Figure below the user query "*hay algun hospital pequno*" is in Spanish and the abstract tree was obtained by parsing this Spanish query using the health domain abstract grammar and concrete Spanish grammar. Furthermore, GF can translate the query into other languages by mapping the tree-like syntactic function (i.e. abstract tree) to the strings. This process is called linearization of abstract tree to the corresponding concrete syntax.

It is worth noting that the GF can generate random sentences (those are of course developed in concrete grammar in different variations to express a single query) similar in meaning to the expressed query, for e.g., the above mentioned query has many forms of expressing that are developed in concrete grammar, the GF generates randomly any form similar in meaning, that is, for the same abstract rule (e.g. "*yo prefiero a un hospital pequno*" has same meaning as of the expressed user query).



userComment (equipmentInquiry (equipCapacity small) hospital)

▸ i prefer a small hospital

▸ में एक छोटा हॉस्पिटल पसंद करूँगा

▸ yo prefiero a un hospital pequno

5.3 Guiding the user access the resources

GF environment supports another interesting functionality for assisting the user where building the query, it presents next acceptable options on the screen when writing. When using this functionality the errors when processing user interventions are minimized, resulting in a friendlier communication. Next figures show how this functionality has been used to guide the user when using the grammars we have developed .

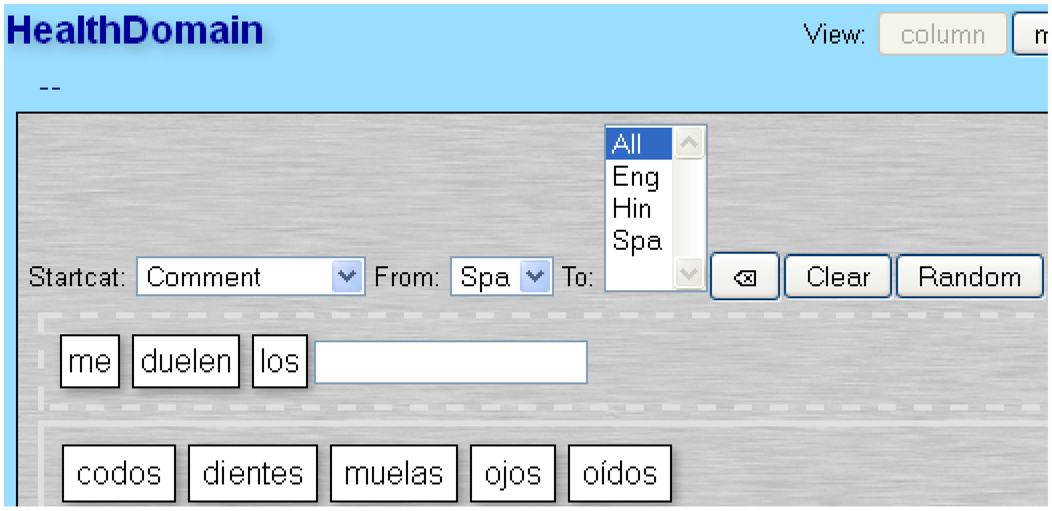


Figure 5.3.1 *demo1: guiding the user in Spanish (health domain)*

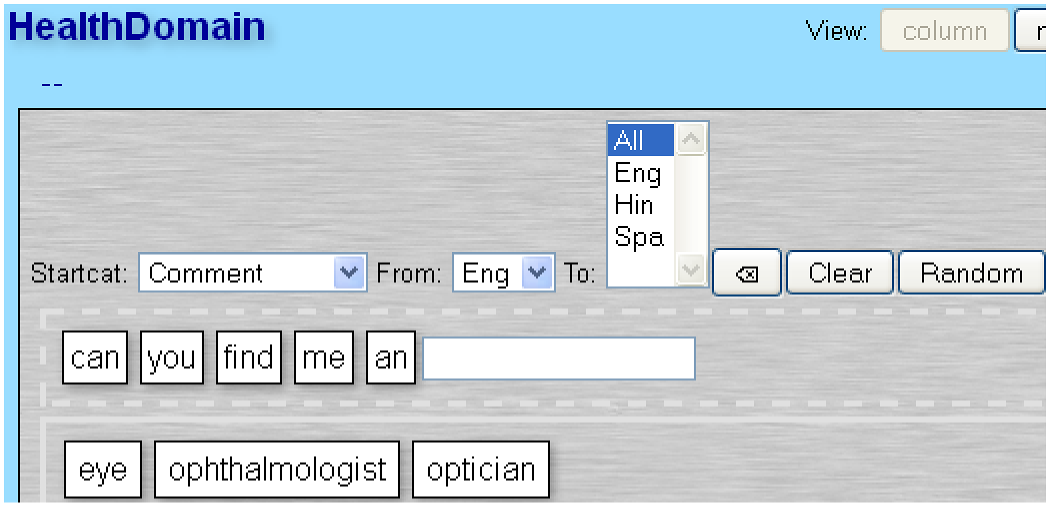


Figure 5.3.2 *demo2: guiding the user in English (health domain)*

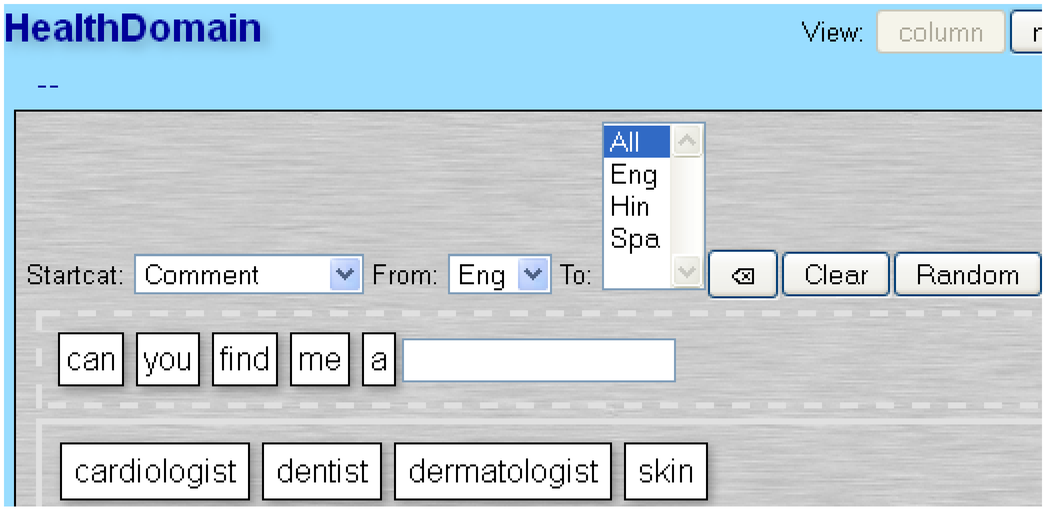


Figure 5.3.3 *demo3: guiding the user in English (health domain)*

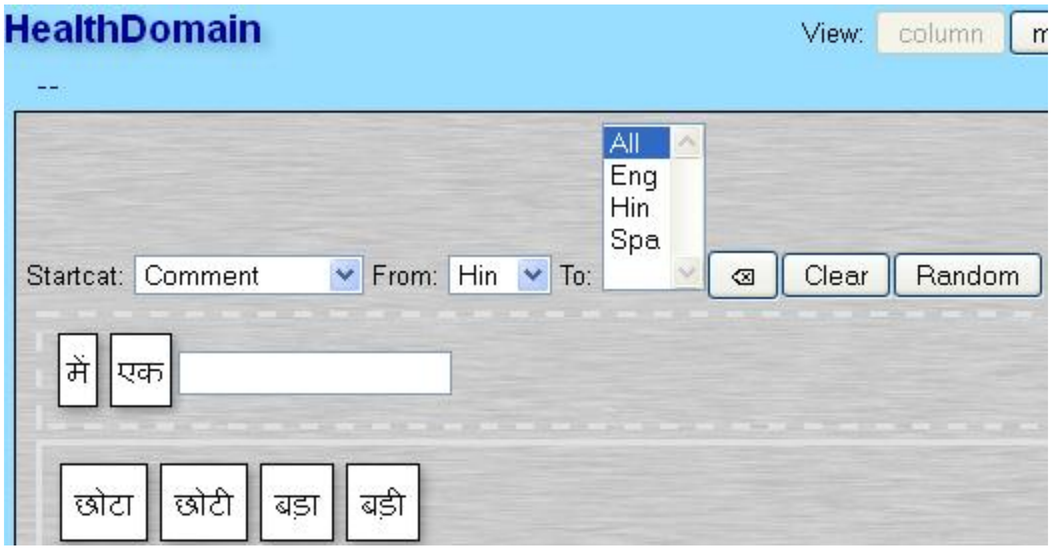
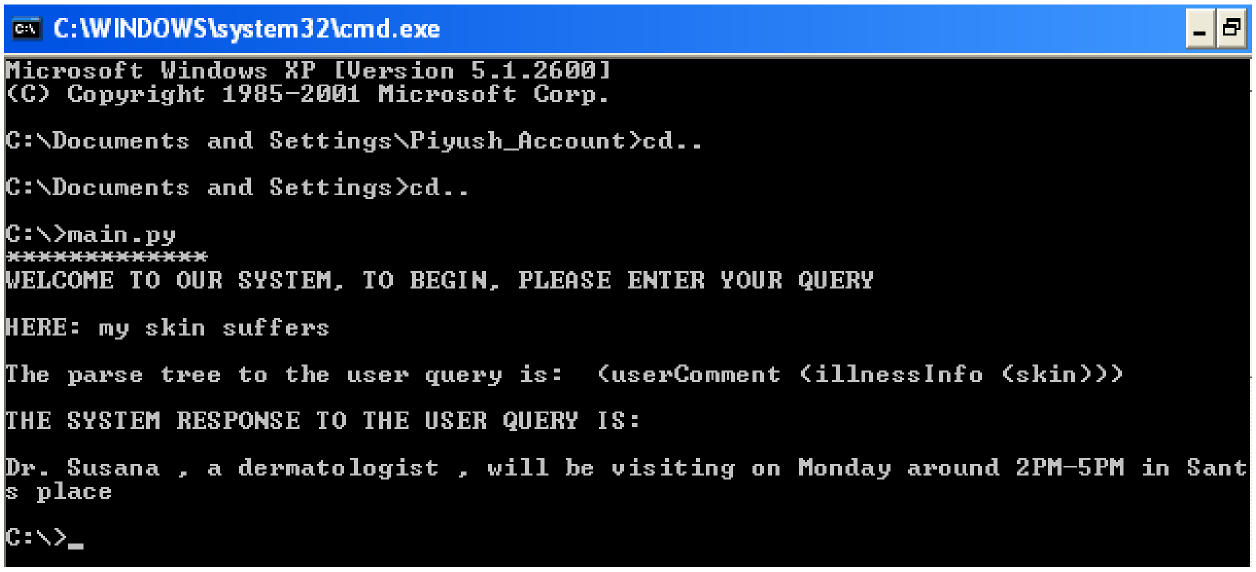


Figure 5.3.4 *demo4: guiding the user in Hindi (health domain)*

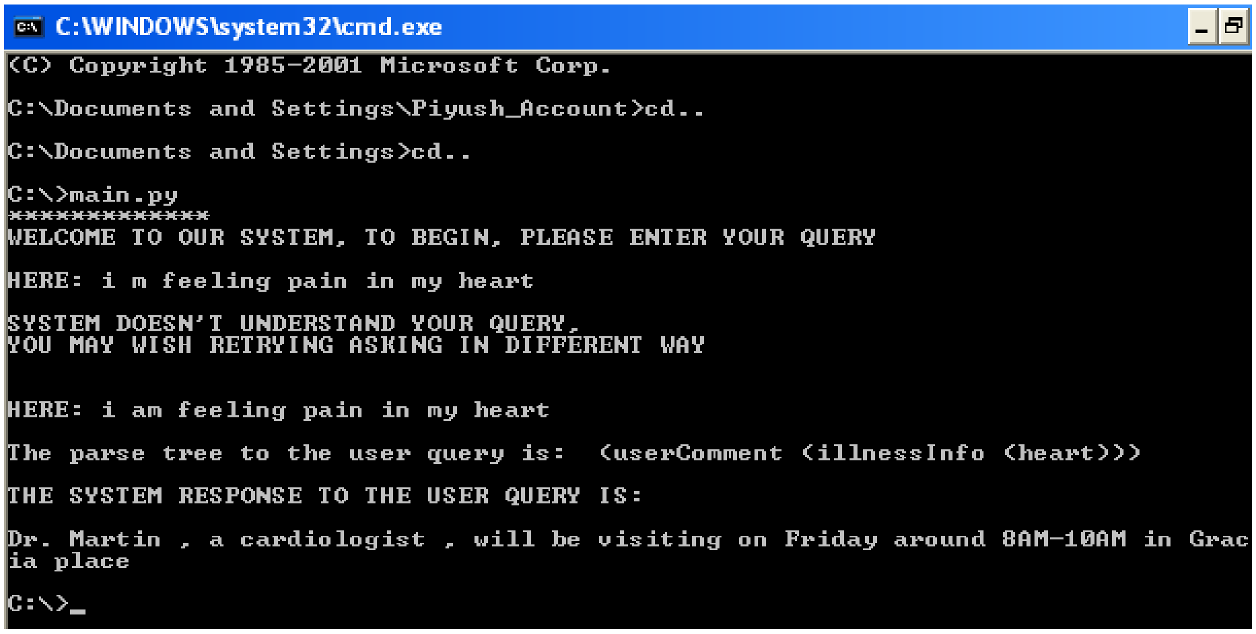
5.4 Demo Experimentations

Along with the NLU component, we tested our system by building a toy prototype and incorporating a small set of databases. The prototype uses the information resulting from the parsed used interventions and generates the answers. However, the answer generated consisted of canned sentences.



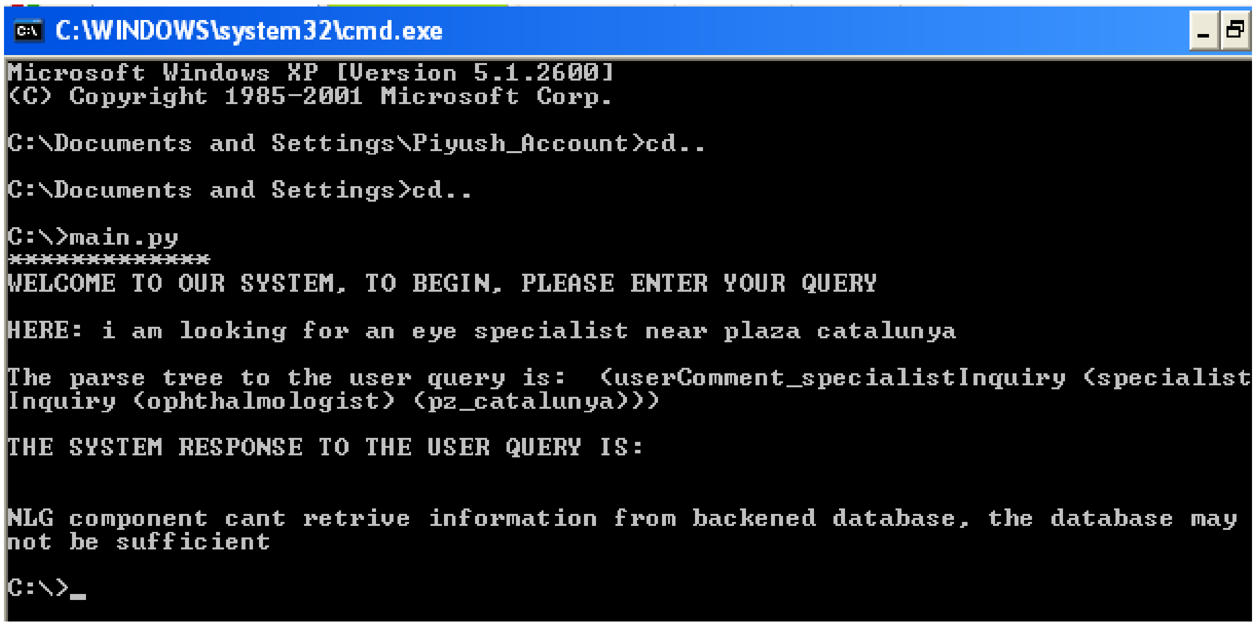
```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Piyush_Account>cd..

C:\Documents and Settings>cd..

C:\>main.py
*************
WELCOME TO OUR SYSTEM, TO BEGIN, PLEASE ENTER YOUR QUERY

HERE: my skin suffers

The parse tree to the user query is:  (userComment (illnessInfo (skin)))

THE SYSTEM RESPONSE TO THE USER QUERY IS:

Dr. Susana , a dermatologist , will be visiting on Monday around 2PM-5PM in Sant
s place

C:\>_
```

Experiment 1: parsing of a sentence "my skin suffers" was successfully done, and then the answer was generated



```
C:\WINDOWS\system32\cmd.exe
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Piyush_Account>cd..

C:\Documents and Settings>cd..

C:\>main.py
**************
WELCOME TO OUR SYSTEM. TO BEGIN. PLEASE ENTER YOUR QUERY

HERE: i m feeling pain in my heart

SYSTEM DOESN'T UNDERSTAND YOUR QUERY.
YOU MAY WISH RETRYING ASKING IN DIFFERENT WAY


HERE: i am feeling pain in my heart

The parse tree to the user query is:  (userComment (illnessInfo (heart)))

THE SYSTEM RESPONSE TO THE USER QUERY IS:

Dr. Martin , a cardiologist , will be visiting on Friday around 8AM-10AM in Grac
ia place

C:\>_
```

Experiment 2: parsing of a sentence "i m feeling pain in my heart" was not successfully done because NLU component doesn't understands this, the system then asks the user to retry the query, second time when the user asks query recognized by NLU, the answer was generated



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Piyush_Account>cd..

C:\Documents and Settings>cd..

C:\>main.py
**************
WELCOME TO OUR SYSTEM. TO BEGIN. PLEASE ENTER YOUR QUERY

HERE: i am looking for an eye specialist near plaza catalunya

The parse tree to the user query is:  (userComment_specialistInquiry (specialist
Inquiry (ophthalmologist) (pz_catalunya)))

THE SYSTEM RESPONSE TO THE USER QUERY IS:


NLG component cant retrive information from backened database. the database may
not be sufficient

C:\>_
```

Experiment 3: parsing of a sentence "i am looking for an eye specialist near plaza catalunya" was successfully done because the NLU component understands this, but since we have not used database large enough that are related to specialist information, thus answer couldn't be generated.

Chapter 6

Conclusions

In this thesis we have proposed the use of domain-restricted ontologies to generate efficient and structured grammars in different languages. In particular we have worked with three languages: English, Hindi and Spanish. These three languages differ in many different ways: vocabulary, syntax and even a different alphabet for Hindi. In order to facilitate the generation of linguistic resources for the three languages our proposal is based on a complete separation of conceptual and linguistic knowledge bases, being conceptual knowledge reused across the three languages. Conceptual knowledge represented in ontologies consists of the domain concepts together and their attributes involved in a particular scenario. This clear separation between conceptual and linguistic knowledge also facilitates the generation of the grammars in a new language, because conceptual knowledge is already defined and only the specific syntactic rules expressed each concept have to be defined.

Our work has focused on developing the domain-restricted resources needed in a web NL system to assist the users when searching for information in two particular scenarios: finding a medical specialists and looking for information about cultural events in the city. In order to facilitate the adaptation to new domains as well as the reusability of knowledge across domains we propose a clear separation of domain knowledge and the general conceptual knowledge that can be shared by several domain scenarios.

Grammars have been implemented in GF, a multilingual grammar environment. GF present several advantages comparing other existing language environments: it supports Hindi alphabet, efficient parsing, library of resources in many languages and it includes useful functionalities (presentation of the resulting parse-tree, user's guidance, sentence translation, generation of sentences from grammars, etc). However main advantage the GF presents considering our work has been the representation of grammars in two separated modules: conceptual (abstract grammar) and syntactic (concrete grammar). For each domain, we represented domain specific conceptual knowledge in ontology to the abstract grammar of GF. In the abstract grammar conceptual instances and values of conceptual attributes were represented as lexical rules while syntactic rules represent the combination of concepts appearing in user's interventions. From this abstract grammar concrete grammar was generated. Although it was needed an individual linguistic expert to further develop the concrete grammar in each language, the inherent features of GF have become increasingly appreciable in providing the experts an organization that facilitates the construction of the rules that support particular morphological and syntactic variations.

The main goal of our work has been to find a general method to facilitate the generation of grammars that are easy to adapt to new languages, new domains and even new users (i.e. young people using informal languages including new words and mistakes). Our goal has not been to construct a complete grammar. For this reason, evaluation to study how many sentences can be supported by the grammars has not been done. Instead we have measured the reusability of grammar components at different level of organizations: at modular level, syntactical level, and lexical level.

We have also worked on the adaptation of a general existing syntactic-semantic taxonomy that facilitate the semi-automatic generation of the system answers. In particular, we have studied its adaptation to the scenario we considered in the health domain. Although, due to the time constraint we did not complete our work on the generation of NL system's responses, we considered the work we have done in this line could also be extended without major problems for this and other domains.

Chapter 7

Future work

Our proposal for representing in a separate way conceptual and linguistic knowledge and general and domain-specific knowledge facilitate the generation of grammars for different domain and languages. We have studied this proposal applied to three very different languages: different alphabet, very different vocabulary and syntax. Adapting our proposal to similar languages will not be a difficult task. This can lead us to a possible future direction to adapt the grammars to similar languages similar to Spanish (e.g., Catalan) and to Hindi (there are several Indian languages similar to Hindi).

A room is still open for more future directions. One promising direction would be to adapt the modular grammars to different types of users considering their age, language skills, and cultural sensitivity. We then can have separate concrete grammars for each type of users resulting in more efficient and friendly system. In order to develop the most appropriate grammar for each type of user we could collect a corpus of user interventions. The grammars we have already developed could be used in a simple interface to collect a corpus of interventions of the different types of users.

Using the functionality of GF that generates sentences randomly, those can be collected for deriving multilingual corpora to be used in different applications of NL, what would be useful especially for Hindi, because there are not many existing resources in this language.

The main problem with grammar based system is that the users often make mistakes when typing. GF interface using our grammars could overcome this problem by forcing the users access the next acceptable words recognized by grammars. However, more interesting approaches can be used instead, for e.g., to integrate a statistical based spelling correction model, before the users express their query. This will ensure our grammar based NLU component reads a grammatically correct user input.

Appendix

Appendix A Domain-Specific English concrete grammar for health domain

```
concrete HealthDomainEng of HealthDomain =  TimeEng, ZoneEng ** {

lincat
   Body_Part = {s : agreement_param => Str; b : boolean};
   Capacity = {s : Str};
   Comment = {s : Str};
   Description = {s : Str};
   Description_1 = {s : Str};
   EquipCapacity = {s : Str};
   Equipment = {s : agreement_param => Str; b : boolean};
   Specialist = {s : Str; b : boolean};

lin
   userComment d = {s = d.s};
   userComment_specialistInquiry d1 = {s = d1.s};
   Specialist_with_Time d1 t = {s = d1.s ++ checkTime_format t.b ++ t.s};
   equipmentInquiry eqc eq = {s = variants {give_equip_info_start ++ "a" ++ eqc.s ++ eq.s ! Sg;
           "is there" ++ var_equip ++ variants {[]; eqc.s} ++ eq.s ! Sg;
           "are there" ++ var_equip ++ variants {[]; eqc.s} ++ eq.s ! Pl}};
   illnessInfo bp = {s = variants {ill_Info ++ bp.s ! variants {Sg; Pl};
           "my"++ variants {
                      bp.s ! Pl  ++ (verb_N bp.b).s ! variants {PresP; PresCont};
                      bp.s ! Sg ++ (verb_N bp.b).s ! variants {PresS; PresCont_S}}}
     ++ moreInfoIll};
   specialistInquiry sp z = {s = giveSpecialist_info_start
            ++ det_a_an sp.b ++ sp.s ++ variants {"in"; "near"; "close to"; "around"}
                       ++ z.s ++ variants {[]; "please"}};
   cardiologist  = {s = "cardiologist"; b = T};
   dentist  = {s = "dentist"; b = T};
   dermatologist  = {s = variants {"dermatologist"; "skin specialist"}; b = T};
   ophthalmologist  = {s = variants {"ophthalmologist"; "eye specialist"; "optician"}; b = F};
```

```
    clinic  = check_agreement_Equip "clinic" T;
    hospital  = check_agreement_Equip "hospital" T;
    pharmacy  = check_agreement_Equip "pharmacy" F;
    equipCapacity capacity = {s = capacity.s};
    small  = {s = "small"};
    big  = {s = "big"};
    ear  = check_agreement_B_P "ear" T;
    elbow  = check_agreement_B_P "elbow" F;
    head  = check_agreement_B_P "head" F;
    eye  = check_agreement_B_P "eye" T;
    leg  = check_agreement_B_P "leg" T;
    face  = check_agreement_B_P "face" F;
    heart  = check_agreement_B_P "heart" F;
    skin  = check_agreement_B_P "skin" F;
    stomach  = check_agreement_B_P "stomach" F;
    teeth  = check_agreement_B_P "tooth" T;

param
    agreement_param = Sg | Pl;
    boolean = T | F;
    agreement_param_N = PresS | PresP | PresCont | PresCont_S;

oper
    giveSpecialist_info_start  = variants {"i am looking for"; "i would like to see"; "i need";
                                                        "can you find me"};
    give_equip_info_start  = variants {"i prefer"; "i would like"};
    ill_Info  = "i am feeling" ++ variants {ill_strength; []} ++ "pain in my";
    ill_strength  = variants {"severe"; "harsh"};
    moreInfoIll  = variants {[]; variants {","; []} ++ "can i get a doctor" ++ variants {
                                      variants {",";  []} ++ "please"; []};"which"++ variants {
                         "specialist"; "doctor"} ++ variants {"do i"; "should i"} ++ variants {"need";
                                                              "look for"}};
    check_agreement_B_P : Str -> boolean -> {s : agreement_param => Str; b : boolean}  = \x,y
                                            -> {s = table {
                                                        Sg => x;
                                                        Pl => case y of {
                                                                   T => case x of {
                                                                            "tooth" => "teeth";
                                                                                 _ => x + "s"
                                                                          };
                                                                    F => x
                                                                          }
                                                          };
                                                      b = y};
    check_agreement_N : Str -> boolean -> {s : agreement_param_N => Str; b : boolean}  = \x,y
                                          -> {s = table {
```

```
              PresP => case y of {
                          T => x;
                          F => x + "s"
                        };
               PresS => x + "s";
               PresCont_S => "is" ++ x + "ing";
               PresCont => case y of {
                             T => "are" ++ x + "ing";
                             F => "is" ++ x + "ing"
                            }
             };
          b = y};
   check_agreement_Equip : Str -> boolean -> {s : agreement_param => Str; b : boolean}  = \x,y
                 -> {s = table {
                          Sg => x;
                          Pl => case y of {
                                      F => "pharmacies";
                                      T => x + "s"
                                            }
                              };
                            b = y};
   hurt_N  = check_agreement_N "hurt";
   suffer_N  = check_agreement_N "suffer";
   verb_N  = variants {hurt_N; suffer_N};
   var_equip  = variants {[]; "any"; "some"};
   det_a_an : boolean -> Str  = \x -> case x of {
        T => "a";
        F => "an"
       };
   checkTime_format : boolean -> Str  = \x -> case x of {
        T => variants {"at"; "around"};
        F => variants {"around"; variants {[]; "in"} ++ "between"}
       };

}
```

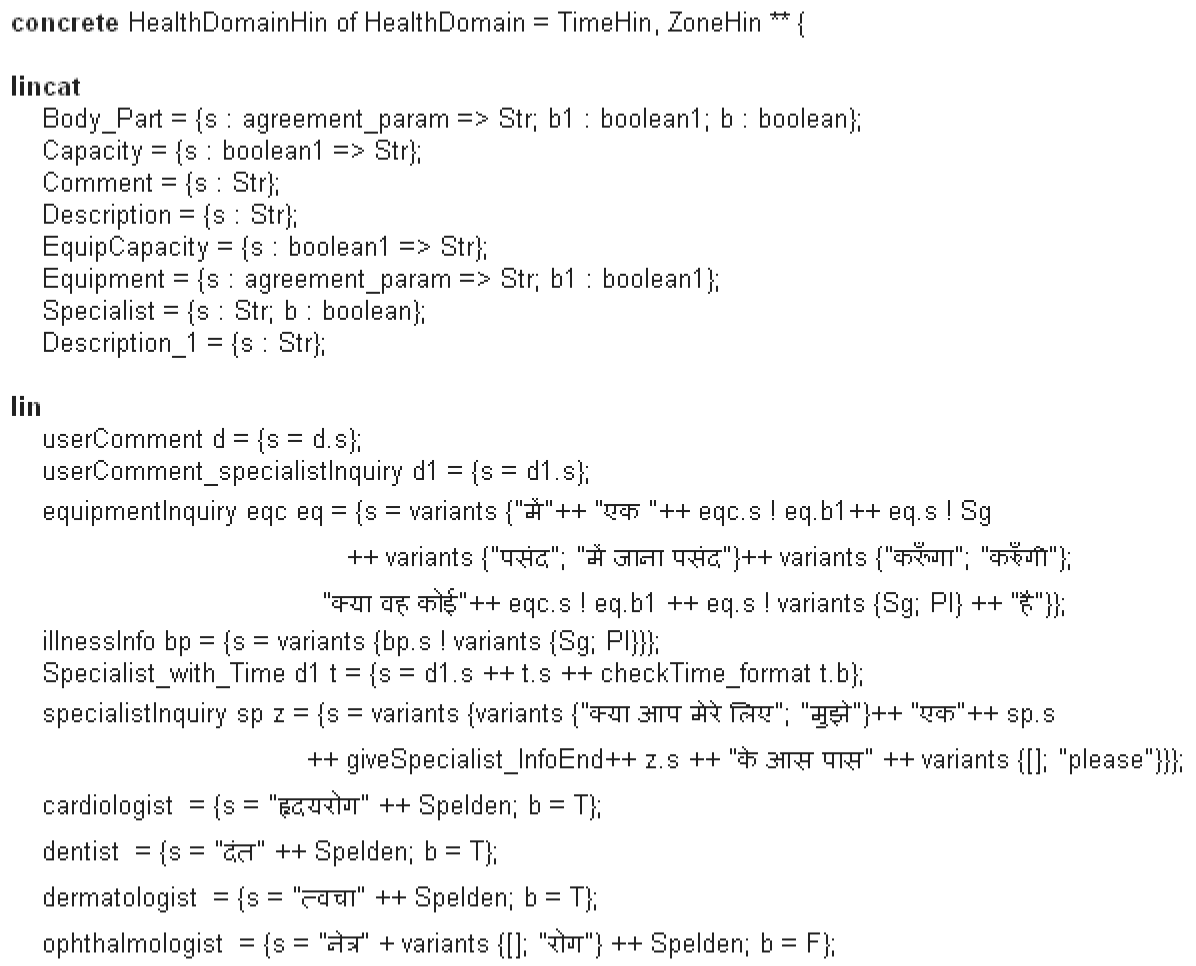
Appendix B Domain-Specific Spanish Concrete Grammar Health Domain

```
concrete HealthDomainSpa of HealthDomain = TimeSpa, ZoneSpa ** {

lincat
   Body_Part = {s : agreement_param => Str; b1 : boolean1; b : boolean};
   Capacity = {s : boolean1 => Str};
   Comment = {s : Str};
   Description = {s : Str};
   EquipCapacity = {s : boolean1 => Str};
```

```
    Equipment = {s : agreement_param => Str; b1 : boolean1};
    Specialist = {s : Str; b : boolean};
    Description_1 = {s : Str};

lin
    userComment d = {s = d.s};
    Specialist_with_Time d1 t = {s = d1.s ++ checkTime_format t.b ++ t.s};
    userComment_specialistInquiry d1 = {s = d1.s};
    equipmentInquiry eqc eq = {s = variants {give_equip_info_start
                              ++ variants {"a"; []} ++ eq.s ! Sg1 ++ eqc.s ! eq.b1;
                                 "hay" ++ eq.s ! variants {Sg; Pl} ++ eqc.s ! eq.b1}};
    illnessInfo bp = {s = variants {"me"++ variants {"duelen" ++ bp.s ! Pl;
                                                        "duele" ++ bp.s ! Sg}}};
    specialistInquiry sp z = {s = giveSpecialist_info_start ++ det_a_an sp.b ++ sp.s
              ++ variants {"en" ++ variants {[]; "el área de"};"cerca" ++ variants {"de"; "del"}}
                 ++ z.s ++ variants {[]; "por favor"}};
    cardiologist  = {s = "cardiólogo"; b = T};
    dentist  = {s = "dentista"; b = F};
    dermatologist  = {s = variants {"dermatólogo"; "médico de cuero"}; b = T};
    ophthalmologist  = {s = variants {"oftalmólogo"; "óptico"}; b = T};
    clinic  = check_agreement_Equip "clínica" FEM;
    hospital  = check_agreement_Equip "hospital" MASC;
    pharmacy  = check_agreement_Equip "farmacia" FEM;
    equipCapacity capacity = {s = capacity.s};
    small  = {s = table {
                        FEM => "pequna" + variants {[]; "s"};
                        MASC => "pequno" + variants {[]; "s"}
                           }};
    big  = {s = table {
                       FEM => "grande" + variants {[]; "s"};
                      MASC => "grande" + variants {[]; "s"}
                      }};
    ear  = check_agreement_B_P "oído" MASC T;
    elbow  = check_agreement_B_P "codo" MASC T;
    head  = check_agreement_B_P "cabeza" FEM F;
    eye  = check_agreement_B_P "ojo" MASC T;
    face  = check_agreement_B_P "cara" FEM F;
    heart  = check_agreement_B_P "corazón" MASC F;
    skin  = check_agreement_B_P "piel" FEM F;
    leg  = check_agreement_B_P " pierna" FEM T;
    teeth  = check_agreement_B_P variants {"muela"; "diente"} MASC T;
    stomach  = check_agreement_B_P "estómago" MASC F;

param
    agreement_param = Sg | Pl | Sg1;
    boolean1 = FEM | MASC;
```

```
   boolean = T | F;

oper
   giveSpecialist_info_start  = variants {"estoy buscando"; "busco";
           variants {"yo"; []} ++ "necesito";
           "puedes" ++ variants {"buscarme"; "buscar"}; "me pueden buscarme"};
   give_equip_info_start  = variants {variants {"yo"; []} ++ "prefiero";
           variants {"me"; []} ++ "gustaría"};
   check_agreement_B_P : Str -> boolean1 -> boolean -> {s : agreement_param => Str;
                          b1 : boolean1; b : boolean}  = \x,y,z ->
                              {s = table {
                                      Sg => case y of {
                                                 FEM => "la" ++ x;
                                                MASC => "el" ++ x
                                                              };
                                     Pl => case z of {
                                                   T => case y of {
                                                                  FEM => "las" ++ x + "s";
                                                                 MASC => "los" ++ x + "s"
                                                                        };
                                              F => case y of {
                                                         FEM => "la" ++ x;
                                                         MASC => "el" ++ x
                                                                     }
                                                              };
                               Sg1 => ""
                                               };
                                b1 = y; b = z};
   check_agreement_Equip : Str -> boolean1 -> {s : agreement_param => Str; b1 : boolean1}  =
                       \x,y -> {s = table {
                                     Sg1 => case y of {
                                                 FEM => "una" ++ x;
                                                MASC => "un" ++ x
                                                              };
                                      Pl => case y of {
                                                  FEM => "algunas" ++ x + "s";
                                                  MASC => "algunos" ++ x + "es"
                                                              };
                                    Sg => case y of {
                                                   FEM => "alguna" ++ x;
                                                   MASC => "algun" ++ x
                                                            }
                                         };
                                    b1 = y};
   det_a_an : boolean -> Str  = \x ->
                          case x of {
                                T => variants {"a"; []} ++ "un";
                                F => variants {"a"; []} ++ variants {"un"; "una"}
                                  };
   checkTime_format : boolean -> Str  = \x ->
                                 case x of {
                                       T => "a" ++ variants {[]; "las"};
                                       F => "entre"
                                        };

                                 }
```

Appendix C Domain-Specific Hindi concrete grammar Health Domain



```
concrete HealthDomainHin of HealthDomain = TimeHin, ZoneHin ** {

lincat
   Body_Part = {s : agreement_param => Str; b1 : boolean1; b : boolean};
   Capacity = {s : boolean1 => Str};
   Comment = {s : Str};
   Description = {s : Str};
   EquipCapacity = {s : boolean1 => Str};
   Equipment = {s : agreement_param => Str; b1 : boolean1};
   Specialist = {s : Str; b : boolean};
   Description_1 = {s : Str};

lin
   userComment d = {s = d.s};
   userComment_specialistInquiry d1 = {s = d1.s};
   equipmentInquiry eqc eq = {s = variants {"मैं"++ "एक "++ eqc.s ! eq.b1++ eq.s ! Sg
                                       ++ variants {"पसंद"; "मैं जाना पसंद"}++ variants {"करूँगा"; "करूँगी"};
                                     "क्या वह कोई"++ eqc.s ! eq.b1 ++ eq.s ! variants {Sg; Pl} ++ "है"}};
   illnessInfo bp = {s = variants {bp.s ! variants {Sg; Pl}}};
   Specialist_with_Time d1 t = {s = d1.s ++ t.s ++ checkTime_format t.b};
   specialistInquiry sp z = {s = variants {variants {"क्या आप मेरे लिए"; "मुझे"}++ "एक"++ sp.s
                                   ++ giveSpecialist_InfoEnd++ z.s ++ "के आस पास" ++ variants {[]; "please"}}};
   cardiologist  = {s = "हृदयरोग" ++ Spelden; b = T};
   dentist  = {s = "दंत" ++ Spelden; b = T};
   dermatologist  = {s = "त्वचा" ++ Spelden; b = T};
   ophthalmologist  = {s = "नेत्र" + variants {[]; "रोग"} ++ Spelden; b = F};
```

```
    clinic  = {s = table {
                        Sg => variants {"क्लीनिक"; "क्लिनिक"};
                        Pl => variants {"क्लीनिक"; "क्लिनिक"}
                            };
                       b1 = MASC};
    hospital  = {s = table {
                        Sg => "हॉस्पिटल";
                        Pl => "हॉस्पिटल्स"
                             };
                      b1 = MASC};
    pharmacy  = {s = table {
                        Sg => "फार्मसी";
                        Pl => "फार्मसिया"
                               };
                      b1 = FEM};
    equipCapacity capacity = {s = capacity.s};
    small  = {s = table {
                       FEM => "छोटी";
                       MASC => "छोटा"
                        }};
    big  = {s = table {
                     FEM => "बड़ी";
                     MASC => "बड़ा"
                       }};
    ear  = check_agreement_B_P "कान" MASC T;
    elbow  = check_agreement_B_P "कोहनी" FEM F;
    head  = check_agreement_B_P ("सिर"|variants {"मस्तिष्क"; "मस्तक"}) MASC F;
   eye  = check_agreement_B_P "आँख" FEM F;
   face  = check_agreement_B_P "चेहरा" MASC F;
   heart  = check_agreement_B_P ("ह्रदय"| "दिल") MASC F;
   skin  = check_agreement_B_P "त्वचा" FEM F;
   leg  = check_agreement_B_P "पैर" MASC T;
   teeth  = check_agreement_B_P "दांत" MASC T;
   stomach  = check_agreement_B_P "पेट" MASC F;

param
   agreement_param = Sg | Pl;
   boolean = T | F;
   boolean1 = FEM | MASC;
```

```
oper
    giveSpecialist_InfoEnd  = variants {"की" ++ variants {"तलाश"; "खोज"} ++ "कर सकते हो क्या"};
    check_agreement_B_P : Str -> boolean1 -> boolean -> {s : agreement_param => Str;
                              b1 : boolean1; b : boolean}  = \x,y,z -> {s = table {
                  Sg => case y of {
                          FEM => "मेरी" ++ x ++ "दुख" ++ "रही" ++ "है";
                          MASC => "मेरा" ++ x ++ "दुख" ++ "रहा" ++ "है "
                        };
                  Pl => case z of {
                          T => case y of {
                                 FEM => "मेरे" ++ x ++ "दुख" ++ "रहे" ++ "है";
                                 MASC => "मेरे" ++ x ++ "दुख" ++ "रहे" ++ "है "
                               };
                          F => case y of {
                                 FEM => case x of {
                                          "आँख" => "मेरी" ++ "आँखे" ++ "दुख" ++ "रही" ++ "है";
                                          _ => "मेरी" ++ x ++ "दुख" ++ "रही" ++ "है"
                                        };
                                 MASC => "मेरा" ++ x ++ "दुख" ++ "रहा" ++ "है "
                               }
                        }
                  };
              b1 = y; b = z};
    Spelden  = variants {"विशेषज्ञ"; "चिकित्सक"};
    checkTime_format : boolean -> Str  = \x -> case x of {
           T => "के आस पास";
           F => variants {"के बिच में"; "के आस पास"}
         };

}
```

Appendix D Domain-Specific English Concrete Grammar for Cultural Events Domain

```
concrete EventDomainEng of EventDomain = ZoneEng, TimeEng ** {

lincat

   Comment = {s : Str};
   Description = {s : Str};
   Event = {s : Str};
   Genre = {s : Str; b : boolean};
   Venue = {s : Str};
   Event_Info = {s : Str};

lin
   userComment d = {s = d.s};
   event_zone_time e z t = {s = e.s++ variants {"around"; "near"; "close to"}
                                 ++ z.s ++ checkTime_format t.b1 ++ t.s};
   event_venue_time e v t = {s = e.s ++ variants {"that takes place"} ++ variants {
                                "at" ++ variants {[]; "venue"}; "in"}
                                   ++ v.s ++ checkTime_format t.b1 ++ t.s};
   genre_event g e = {s = give_info ++ det_a_an g.b ++ g.s ++ e.s};
   royal_play_ground  = {s = "royal play ground"};
   city_hall  = {s = "city hall"};
   centre_public_ground  = {s = "centre public ground"};
   bcn_centre_hall  = {s = "barcelona centre hall"};
   auditori  = {s = "auditori"};
   romantic  = {s = "romantic"; b = T};
   musical  = {s = "musical"; b = T};
   orchestic  = {s = "orchestic"; b = F};
   dramatic  = {s = "dramatic"; b = T};
   sport  = {s = "sport"};
   concert  = {s = "concert"};
   movie  = {s = variants {"film"; "movie"}};

param
   boolean = T | F;

oper

   give_info  = variants {"i am looking for"; "can you find me";
                                  "i want to see"};
   det_a_an : boolean -> Str  = \x -> case x of {
                                              T => "a";
                                              F => "an"
                                             };
   checkTime_format : bool -> Str  = \x -> case x of {
                          T1 => variants {"at"; "around"};
```

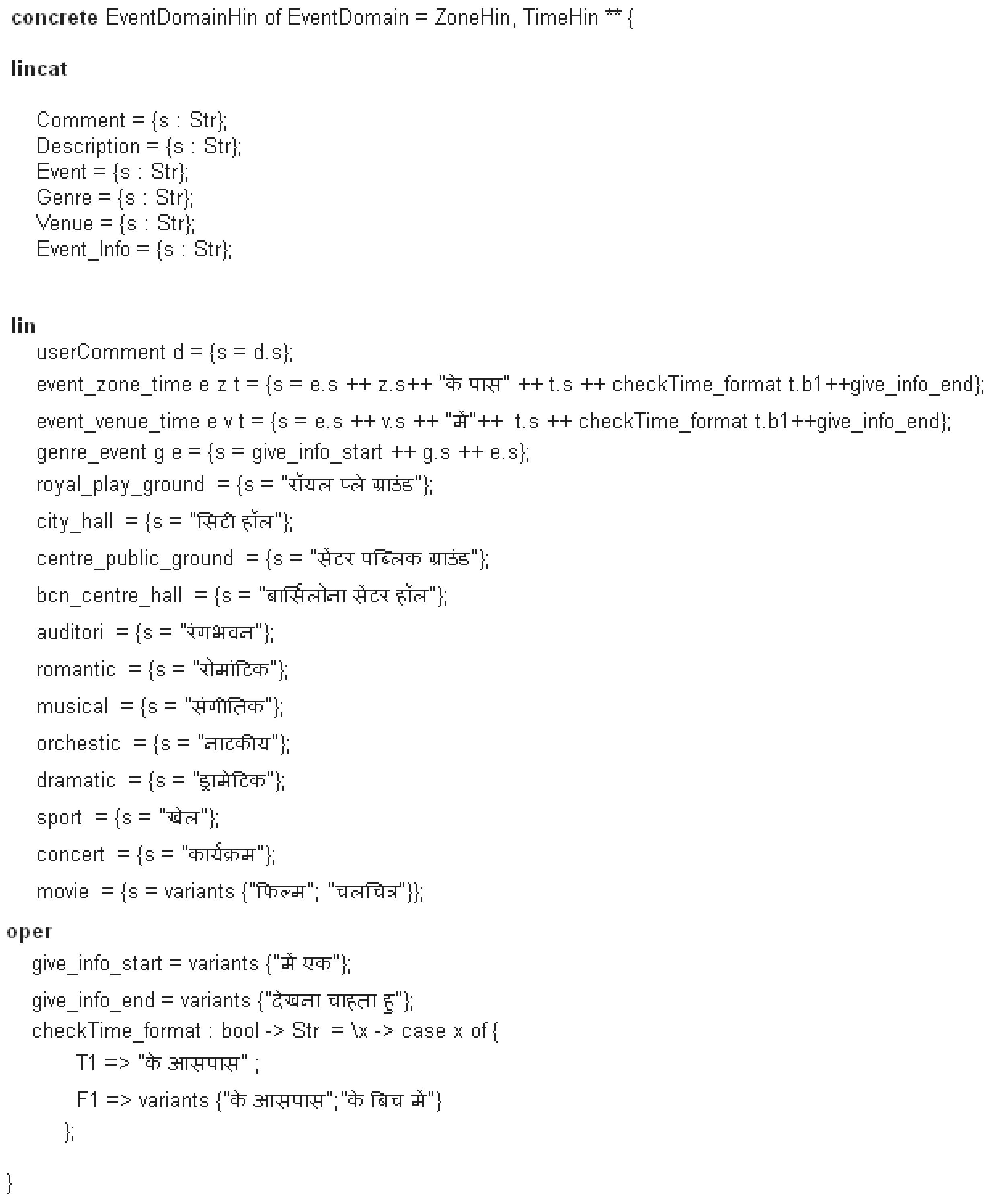
```
                    F1 => variants {"around"; variants {[]; "in"} ++ "between"}
                            };

                       }
```

Appendix E   Domain-Specific Spanish Concrete Grammar for Cultural Events Domain

```
concrete EventDomainSpa of EventDomain = ZoneSpa, TimeSpa ** {

lincat

  Comment = {s : Str};
  Description = {s : Str};
  Event = {s : Str; b : boolean};
  Genre = {s : boolean => Str};
  Venue = {s : Str};
  Event_Info = {s : Str};

lin
  userComment d = {s = d.s};
  event_zone_time e z t = {s = e.s++ variants {"en" ++ variants {[]; "el área de"};
                  "cerca" ++ variants {"de"; "del"}} ++ z.s ++ checkTime_format t.b1 ++ t.s};
  event_venue_time e v t = {s = e.s++ "en"++ variants {[];"venue"}
                          ++ v.s ++ checkTime_format t.b1 ++ t.s};
  genre_event g e = {s = give_info ++ det_a_an e.b++ e.s ++ g.s ! e.b};
  royal_play_ground  = {s = "royal play ground"};
  city_hall  = {s = "city hall"};
  centre_public_ground  = {s = "centre public ground"};
  bcn_centre_hall  = {s = "barcelona centre hall"};
  auditori  = {s = "auditori"};
  romantic  = {s = table {
                   FEM => "romántic" + "a";
                   MASC => "romántic" + "o"
                    }};
  musical  = {s = table {
                  FEM => "músic" + "a";
                  MASC => "músic" + "o"
                   }};
  orchestic  = {s = table {
                FEM => "orchestic" + "a";
               MASC => "orchestic" + "o"
                   }};
  dramatic  = {s = table {
                  FEM => "dramatic" + "a";
                  MASC => "dramatic" + "o"
                   }};
  sport  = {s = "deporte"; b = MASC};
```

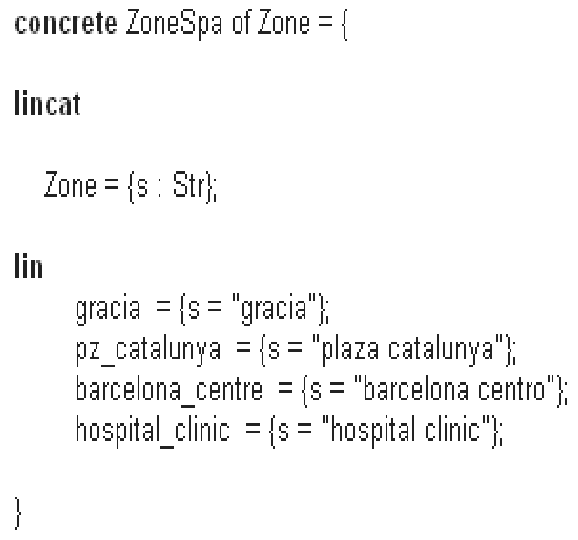
```
   concert  = {s = "concierto"; b = MASC};
   movie  = {s = "película"; b = FEM};
   interval_hourInfo ih = {s = ih.s; b1 = ih.b1};
   hourInfo h = {s = h.s; b1 = h.b1};
   t_11AM_2PM  = {s = "11AM-2PM"; b1 = F1};
   t_8AM_11AM  = {s = "8AM-11AM"; b1 = F1};
   t_2PM_5PM  = {s = "2PM-5PM"; b1 = F1};
   t_6PM_9PM  = {s = "6PM-9PM"; b1 = F1};
   t_11AM  = {s = "11AM"; b1 = T1};
   t_12PM  = {s = "12PM"; b1 = T1};
   t_9AM  = {s = "9AM"; b1 = T1};

param
   boolean = FEM | MASC;

oper
   give_info = variants {"quiero ver"; "estoy buscando";"busco"};
   det_a_an : boolean -> Str  = \x -> case x of {
                                        MASC => "un";
                                        FEM => "una"
                                            };
   checkTime_format : bool -> Str  = \x -> case x of {
                                               T1 => "a" ++ variants {[]; "las"};
                                               F1 => "entre"
                                                };

                                              }
```

F Domain-Specific Hindi Concrete Grammar for Cultural Events Domain

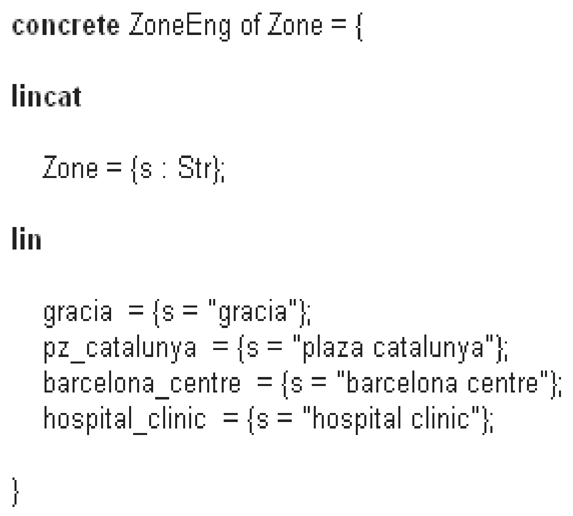


```
concrete EventDomainHin of EventDomain = ZoneHin, TimeHin ** {

lincat

    Comment = {s : Str};
    Description = {s : Str};
    Event = {s : Str};
    Genre = {s : Str};
    Venue = {s : Str};
    Event_Info = {s : Str};


lin
    userComment d = {s = d.s};
    event_zone_time e z t = {s = e.s ++ z.s++ "के पास" ++ t.s ++ checkTime_format t.b1++give_info_end};
    event_venue_time e v t = {s = e.s ++ v.s ++ "में"++  t.s ++ checkTime_format t.b1++give_info_end};
    genre_event g e = {s = give_info_start ++ g.s ++ e.s};
    royal_play_ground  = {s = "रॉयल प्ले ग्राउंड"};
    city_hall  = {s = "सिटी हॉल"};
    centre_public_ground  = {s = "सैंटर पब्लिक ग्राउंड"};
    bcn_centre_hall  = {s = "बार्सिलोना सैंटर हॉल"};
    auditori  = {s = "रंगभवन"};
    romantic  = {s = "रोमांटिक"};
    musical  = {s = "संगीतिक"};
    orchestic  = {s = "नाटकीय"};
    dramatic  = {s = "ड्रामेटिक"};
    sport  = {s = "खेल"};
    concert  = {s = "कार्यक्रम"};
    movie  = {s = variants {"फिल्म"; "चलचित्र"}};
oper
   give_info_start = variants {"मैं एक"};
   give_info_end = variants {"देखना चाहता हु"};
   checkTime_format : bool -> Str  = \x -> case x of {
          T1 => "के आसपास" ;
          F1 => variants {"के आसपास";"के बिच में"}
        };

}
```

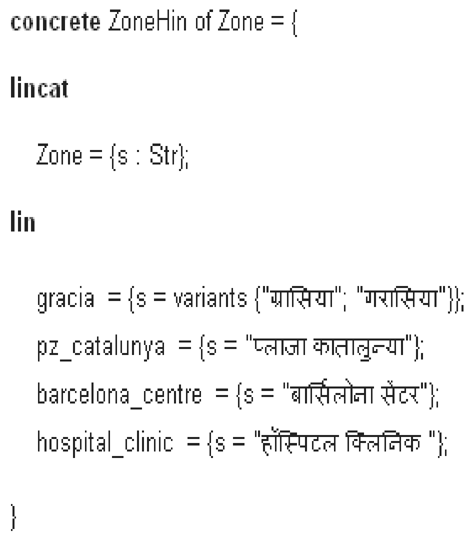
Appendix G domain-independent Zone concrete grammars



```
concrete ZoneSpa of Zone = {

lincat

    Zone = {s : Str};

lin
        gracia  = {s = "gracia"};
        pz_catalunya  = {s = "plaza catalunya"};
        barcelona_centre  = {s = "barcelona centro"};
        hospital_clinic  = {s = "hospital clinic"};

}
```

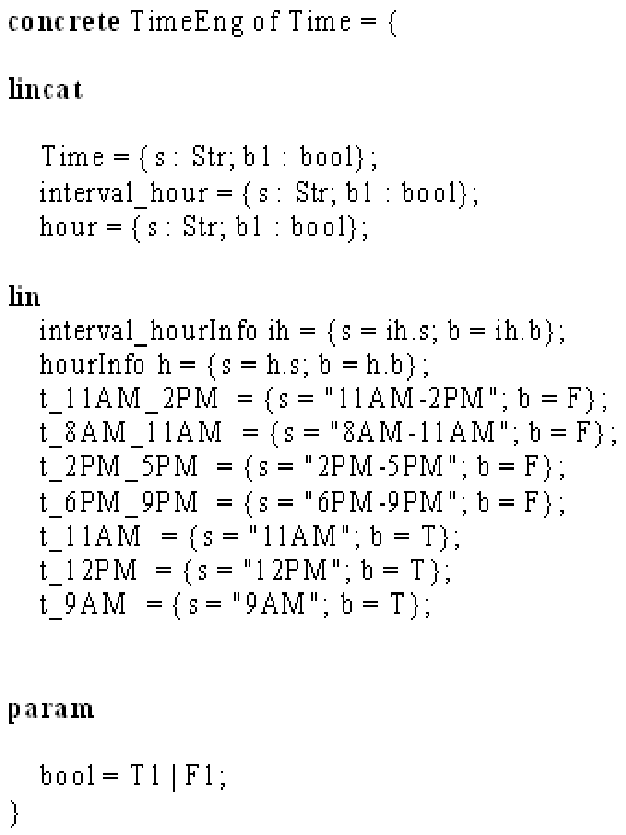


```
concrete ZoneEng of Zone = {

lincat

    Zone = {s : Str};

lin

    gracia  = {s = "gracia"};
    pz_catalunya  = {s = "plaza catalunya"};
    barcelona_centre  = {s = "barcelona centre"};
    hospital_clinic  = {s = "hospital clinic"};

}
```



```
concrete ZoneHin of Zone = {

lincat

   Zone = {s : Str};

lin

   gracia  = {s = variants {"ग्रासिया"; "गरासिया"}};

   pz_catalunya  = {s = "प्लाजा कातालुन्या"};

   barcelona_centre  = {s = "बार्सिलोना सैंटर"};

   hospital_clinic  = {s = "हॉस्पिटल क्लिनिक "};

}
```

Appendix H domain-independent Time concrete grammars



```
concrete TimeEng of Time = {

lincat

   Time = {s : Str; b1 : bool};
   interval_hour = {s : Str; b1 : bool};
   hour = {s : Str; b1 : bool};

lin
   interval_hourInfo ih = {s = ih.s; b = ih.b};
   hourInfo h = {s = h.s; b = h.b};
   t_11AM_2PM  = {s = "11AM-2PM"; b = F};
   t_8AM_11AM  = {s = "8AM-11AM"; b = F};
   t_2PM_5PM  = {s = "2PM-5PM"; b = F};
   t_6PM_9PM  = {s = "6PM-9PM"; b = F};
   t_11AM  = {s = "11AM"; b = T};
   t_12PM  = {s = "12PM"; b = T};
   t_9AM  = {s = "9AM"; b = T};


param

   bool = T1 | F1;
}
```

Note: TimeSpa and TimeHindi are exactly the same as TimeEng because of the same lexicon (number alphabets) used.

References

Chong Wang, Miao Xiong, Qi Zhou and Yong Yu (2007), "*PANTO: A Portable Natural Language Interface to Ontologies*", Proceedings 4th European Semantic Web Conference, ESWC, 473-487.

Oberle, D., Ankolekar, A., Hitzler, P., Cimiano, P., Sintek, M., Kiesel, M., Mougouie, B., Vembu, S., Baumann, S., Romanelli, M., Buitelaar, P., Engel, R., Sonntag, D., Reithinger, N., Loos, B., Porzel, R., Zorn, H.P., Micelli, V., Schmidt, C., Weiten, M., Burkhardt, F., Zhou, J. (2006): "*Dolce ergo sumo: On foundational and domain models in swinto (smartweb integrated ontology)",* Technical report, AIFB, Karlsruhe (July 2006).

Victor W. Zue and James R. Glass, (2000), "*Conversational Interfaces: Advances and Challenges*", Published in Proceedings of the IEEE, VOL. 88, NO. 8, 1166-1180, August, 2000.

J. Butzberger, H. Murveit, and M. Weintraub, (1992), "*Spontaneous Speech Effects in Large Vocabulary Speech Recognition Applications*", Proc. ARPA Workshop on Speech and Natural Language, 339-344, 1992.

L. Hetherington and V. Zue, (1991), "*New words: Implications for Continuous Speech Recognition,*" Proc. Eurospeech, 475-931, 1991.

J. Dowding, J. Gawron, D. Appelt, J. Bear, L. Cherny, R.Moore, and D.Moran, (1993), "*Gemini: A Natural Language System for Spoken Language Understanding*", Proc. ARPA Workshop on Human Language Technology, 21-24, 1993.

W. Ward, (1990), "*The CMU Air Travel Information Service: Understanding Spontaneous Speech*", Proc. ARPA Workshop on Speech and Natural Language, 127-129, 1990.

J. Glass, J. Polifroni, and S. Sene (1994), "*Multilingual Language Generation across Multiple Domains*", Proc. ICSLP, 983-976, 1994.

A. Oh (2000), "*Stochastic Natural Language Generation for Spoken Dialog Systems*", M.S. Thesis, CMU, May 2000.

D. Klatt (1987), “*Review of text-to-speech conversion for English*", J. Acoust. Soc. Am., 82(3), 737-793, 1987.

Y. Sagisaka, N. Kaiki, N. Iwahashi, and K. Mimura (1992), “*ATR v-Talk Speech Synthesis System*", Proc. ICSLP, 483-486, 1992.

Daniel Sonntag , Ralf Engel , Gerd Herzog , Alexander Pfalzgraf , Norbert Pfleger , Massimo Romanelli , Norbert Reithinger, (2007), “*SmartWeb Handheld — Multimodal Interaction with Ontological Knowledge Bases and semantic web services*”, Proceeding ICMI'06/IJCAI'07 Proceedings of the ICMI 2006 and IJCAI 2007 international conference on Artificial intelligence for human computing Pages 272-295.

Gangemi, A., Guarino, N., Masolo, C., Oltramari, A., Schcneider, L.: “*Sweetening Ontologies with DOLCE*”, In 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW02). Volume 2473 of Lecture Notes in Computer Science., Sig¨unza, Spain (Oct. 1–4 2002) 166 ff.

Niles, I., Pease, A.: “*Towards a Standard Upper Ontology*”, In Welty, C., Smith, *B., eds*.: Proceedings of the 2nd International Conference on Formal Ontology in Information Systems (FOIS-2001), Ogunquit, Maine (October 17–19 2001).

Reithinger, N., Sonntag, D.: “*An integration framework for a mobile multimodal dialogue system accessing the semantic web”,* In: Proc. of Interspeech’05, Lisbon, Portugal (2005).

Philipp Cimiano, Peter Haase, J¨org Heizmann (2007), “*Porting natural language interfaces between domains: an experimental user study with the ORAKEL system*”, IUI '07 Proceedings of the 12th international conference on Intelligent user interfaces, 180-189.

C. Masolo, S. Borgo, A. Gangemi, N. Guarino, and A. Oltramari (2003), “*Ontology library (final)*”, WonderWeb deliverable D18.

Didier Guzzoni, Charles Baur, and Adam Cheyer (2006), “*Active: A unified platform for building intelligent web interaction assistants*”, Proceeding WI-IATW '06 Proceedings of the 2006 IEEE/WIC/ACM international conference on Web Intelligence and Intelligent Agent Technology, 417-420.

Fellbaum, C., *Wordnet: An Electronic Lexical Database.* Cambridge: MIT Press (1998).

Cohen, W.W., Ravikumar, P., Fienberg, S.E.: *A Comparison of String Distance Metrics for Name-Matching Tasks*. In: IIWeb. (2003) 73-78.

S.W. Hamerich, V. Schubert, V. Schless, R. de C´ordoba, J.M. Pardo, L.F. dHaro, B. Kladis, O. Kocsis, and S. Igel. *Semi-automatic generation of dialogue applications in the gemini project*. In SIGdial Workshop on Discourse and Dialogue, pages 31–34, 2004.

S.W. Hamerich, Y.F. Wang, V. Schubert, V. Schless, and S. Igel., *Xml-based dialogue descriptions in the gemini project*, In Berliner XML-Tage, 2003.

M. González, "*DIGUI: a Flexible Dialogue System for Guiding the User Interaction to Access Web Services",* PhD thesis, Universitat Politecnica de Catalunya, 2001/

M. Gatius. "*Using an ontology for guiding NL interaction with knowledge based systems*", PhD thesis, Universitat Politecnica de Catalunya, 2001.

J.A. Bateman, B. Magnini and F. Rinaldi, (1994), *The Generalized {Italian, German, English} Upper Model*, Workshop Proceedings, ECAI-94 Workshop on Implemented Ontologies, Amsterdam.

Martin-Löf, P. (1982), "*Constructive mathematics and computer programming*", In Cohen, Los, Pfeiffer, and Podewski, editors, *Logic, Methodology and Philosophy of Science VI*, pages 153–175. North-Holland, Amsterdam.

A. Ranta (2004), "*Grammatical Framework : A Type Theoretical Grammar Formalism",* The Journal of Functional Programming 14(2), 145-189.

A. Ranta (2011), "*Grammatical Framework: Programming with Multilingual Grammars*", CSLI Publications, Stanford.