Freie Universität Bozen
Libera Università di Bolzano
Free University of Bozen · Bolzano

Deep Analysis for an Interactive Question Answering System

a thesis submitted to the
Free University of Bozen-Bolzano
for the degree of
European Master in Computational Logic
by

Marija Slavkovik
supervised by Dr. Raffaella Bernardi
Faculty of Computer Science,
Knowledge Representation meets Databases Research Center

October 2007

Acknowledgments

This thesis was conducted at the Knowledge Representation meets Databases Research Center and would not have existed without the helpful discussions, insights, support and affections of all the people who work there and who I had the pleasure to meet with on daily bases.

I would like to specially thank Dr. Rosella Gennari for her irreplaceable insights in the area of decidable fragments of first-order logic.

I would like to express my enormous gratitude to Dr. David Toman for his great patience in discussing with me parts of this thesis, as well for his knowledgeable advices and proof reading in the final stages of this thesis.

This thesis would not have been completed without the great support of my supervisor, Dr. Raffaella Bernardi, who had the patience to tolerate even the most unmanageable and the generosity to invest many weekend hours into the work of this thesis. Her affection and faith in my abilities gave me the self confidence to pursue a career in research. The numerous hours spent in discussion with her opened the world of linguistic to me, but they also enriched me with valuable life knowledge.

Abstract

The improvement of current Question Answering (QA) systems relies on finding ways to support the traditional statistic approach to QA with logic reasoning [9]. In this thesis we show one way of supporting an Interactive Question Answering system with logic reasoning.

As a case study we make an overview of BoB, a chatter-bot which interactively answers questions over the library domain [41], [40]. BoB retrieves the most likely answer to a question by searching a tree structure of topic-arranged question patterns and responses. The performance of BoB depends on the intelligent construction of this tree. We suggest an architecture of a Logic Support Unit (LSU) for BoB (Chapter 2). The LSU will support BoB's work by performing verification or refutation of the retrieved answers and extraction of the specific answer(s) from the verified answers.

We focus on determining suitable semantic representations for the question and possible answers, and develop a method to represent the task of answer verification and specific answer extraction in line with the method presented in [7]. For the purpose of building semantic representations for natural language, we use BOXER which builds first-order logic formulas from parsed natural language sentences. To allow for efficient reasoning, we define natural language fragments whose sentences have decidable first-order representations (Chapter 4). The fragments have high coverage by following simple lexicon restriction rules (91% of both the possible answers and the questions from the analysed corpora can be written using only the fragments).

We represent the problem of answer verification and specific answer extraction in terms of Answer Set Programming (Chapter 5). We show how to build logic programs from the first-order representations of the questions and answers in the decidable natural language fragments. We show how to analyze the answer sets of those programs to verify or refute an answer and to extract a specific answer.

As a general conclusion, we find that the ASP framework has many features which can be used for the task of supporting IQA with deep analysis. The most promising of these features we present as Future Work of this thesis (Chapter 6).

Contents

Chapter 1

Introduction

The majority of all human knowledge is solely represented in natural language. This knowledge is accessible for humans, who can understand the natural language texts and answer questions about them, but it is not in the same measure accessible for machines. The the fields of Knowledge Representation and Reasoning (KR&R) and Databases suffered vast developments over the past decade but this did not diminish the demand for developing new and improving the old techniques of processing knowledge resources in natural language. The task of automated retrieval of specific information from a knowledge source in natural language, as a response to a natural language question, is not a simple one even for relatively small knowledge resources.

One approach toward solving this task is to cast it in KR&R terms as the task of answering a query over a Knowledge Base (KB). To this end, the knowledge has to be translated from natural language to a Knowledge Representation (KR) language, such as OWL[1] [37] or DL-lite [20]. The question has to be represented as a query over the corresponding KB.

The translation from a Natural Language (NL) to a KR language is made difficult by the fact that natural language has two main computationally undesirable properties. Natural language is ambiguous (one sentence can have more then one meaning) and syntactically rich (one and the same meaning can be conveyed by many different natural language expressions).

Even if the translation NL to KR were easy, querying large KB's is, for practical purposes, computationally inefficient because query answering over KB (if they are built over an expressive KR language) is of high complexity[2].

It also has to be taken into consideration that the retrieved answers from a KB will be represented in the KR Language of the KB. The users of a question answering system are, in general, people who are not acquainted with formal languages so they will not be able to understand the retrieved answer. (The gap between the non-expert user and the formal languages is an existing problem in KR and Databases and it caused for special attention to be devoted into creating formal languages which closely resemble natural languages[3]).

The task of finding an answer to a question, when both are in natural language, is traditionally being handled by the field of Question Answering (QA).

[1] http://www.w3.org/TR/owl-features/
[2] http://www.cs.man.ac.uk/∼ezolin/dl/
[3] http://attempto.ifi.unizh.ch/site/description/index.html

Classical QA is executed by performing syntactic analysis on the question and using Information Retrieval (IR) techniques to process the text (knowledge source). IR systems are today able to successfully isolate the short text (snippet) which has the highest probability to contain the required information even from large text collections[4]. To do so, IR heavily relies on statistical and probabilistic methods. The obtained result (reply) is the "most likely" answer and the QA System has no way of "knowing" when the retrieved answer certainly answers the posed question. The answer verification requires additional intelligence.

When the domain over which the questions are being answered is restricted (in size and topic) and known to the QA system, the knowledge source can be structured to increase the precision of the system. In this case the result will be more precise but it will still be probabilistic. It is up to the user to read the entire returned snippet and maybe find in it, or deduce from it, the exact information which he/she has asked for.

An efficient QA system should verify that the retrieved text certainly answers the posed question before presenting it to the user. In addition, it should return only the information which was specifically asked for and not a chunk of text. In order to be able to do all this, the QA System can not rely only on syntactical analysis of the question. It needs to employ a semantic analysis of the question and possible answer, and be able to perform logic reasoning over both.

The field of Natural Language Representation and the field of KR&R are both subfields of Artificial Intelligence and in the beginning of AI the methods and goals of both interleaved. But, as each field developed, they set different goals and diverged from the each other—KR&R toward logic and NLP toward linguistic sciences and statistics. As a result, today it is not simple and straightforward to combine the methods of QA with methods of KR. In the field of QA there is inevitable need for these two AI disciplines to reunite. In the last years, the QA recently became aware that the improvement of the performance of the QA Systems depends on supporting the traditional statistic approach with logic reasoning. Consequently, it suggested that special attention should be given to developing such supporting means and methods [9].

Intrigued by this new emerging field of combined efforts of logic and QA, we were motivated to devote the work of this master thesis to it and explore one possibility for supporting the standard linguistic approaches of Question Answering with deep analysis (semantic representations and logic reasoning). As a case study, we considered an ongoing Question Answering research project at the Free University of Bozen-Bolzano—the Bolzano library chatter-bot BoB [41], [40].

BoB is aimed to answer questions over the domain of the library of the Free University of Bozen-Bolzano in a dialog setting (interactively). The knowledge about the library is structured in a tree whose leaves are question patterns and answers corresponding answers. The answers in the tree are arranged according to topics—answers on the same topic share mother nodes. BoB performs a shallow analysis of the user utterance (regular expressions pattern matching), attempts to determine the topic of the question, and searches the tree starting from the identified topic. The underlying assumption is that the dialog between the system and the user is led in a topic, and the questions that the user asks will most likely be in the context of the topic. The position in the tree at the

[4] http://trec.nist.gov/

given moment of the on-going dialog is to be used as an indicator of the focus of the question.

The answers which BoB returns are the most likely answers to the posed question, and BoB has no way of "knowing" if what it returns certainly answers the users question.

In a dialog setting, the returned answer has to be specific to the question asked, hence large snippets of text do not suffice. The shallow analysis of the question is insufficient to determine which information is specifically asked for. To retrieve a specific information BoB relies on the "intelligent" structure of the tree. These trees (so called focus trees) are built to offer a specific answer by predicting the questions. The method of focus trees is efficient but the construction of well balanced ("finely tuned") focus trees requires a lot of time and effort.

To provide BoB with the possibility to verify if a retrieved answer certainly answers a question, as well as to reduce the time needed for a construction of an efficient focus tree, we propose that BoB should be augmented with a Logic Support Unit (LSU). The intended role of the LSU is as follows.

The answers in the focus tree can be kept in form of snippets and BoB should retrieve not only the one most likely answer, but the first few most likely answers. The LSU receives (from BoB) the question and the most likely answers for that question in natural language. The LSU will parse the question and answer(s) and construct a semantic representation for them. Then it will verify or falsify, by means of logic reasoning, each possible answer representation for the question representation. Once an answer is verified, the LSU will retrieve the specific answer from its semantic representation. The LSU will translate the specific answer(s) back to natural language and return it to BoB. The development of the LSU incorporates two core tasks.

The first task is the automatic construction of semantic representations for natural language. The answers which BoB retrieves are known before hand and they can be represented using some formal language. However, these answers are provided by library domain experts unfamiliar with formal languages or semantic representations and even the most intuitive formal language such as ACE [42] requires time and effort to be learned. Wanting to minimize the input of domain expert time we choose to build semantic representations from natural language.

The field of formal semantics of natural language offers the theory (CCG Grammar and Montague Theory) and tools (the CCG Parser and the semantic representation tool BOXER) to efficiently build wide-coverage first order logic representations for natural language sentences and questions.

The second task incorporated in the development of the LSU is to verify a possible answer to a question based on the first-order representation of both.

One way of formal verification of answers over semantic representations like the ones BOXER builds is presented in the work of Gabsdil and Bos [16]. In [16] the corresponding formulas of the question and the possible answer are used to build a list of propositions. The task of answer verification is reduced to testing each of the propositions for satisfiability, namely an answer is defined as proper for a question if at least one of the prepositions from the list is consistent and at least one of them is inconsistent.

The prototype (MIDAS) built for this system uses theorem provers (from

MathWeb[5]) for the satisfiability testings. However, satisfiability of FOL formulas is undecidable and a theorem prover may not terminate with a proof. To cope with this undesirable possibility, the theorem provers are paired with model builders.

The QA System of [16] aims to answer questions over the domain of route services and aims to provide the user with a description of a route on the basis of a starting point. In this setting, the system is the one who initializes the dialog and asks the questions while the user is the one who provides the answers. Consequently, the only reasoning task that the system has to perform is the answer verification.

In our setting we need to extract the specific answer from the verified answer in addition to the answer verification. The approach of verification by testing for satisfiability proposed in [16] does not offer us this possibility.

As the work in [16] shows, answer verification and specific answer extractions by means of logic reasoning requires a decision procedure over first-order logic formulas. First-order logic (FOL) is, in general, undecidable. To cope with the undecidability of FOL, we will take advantage of the possibility to use restricted natural language because the answers are pre-known.

By posing simple restrictions on the lexicon of the language in which the answers are built, we can guarantee that the semantic representations obtained for the possible answers will be decidable FOL formulas. The lexicon restrictions are simple enough (a list of words whose usage in a sentence is restricted or prohibited) to allow for the domain experts to be able to quickly construct the answers without special preparation or training.

Unfortunately, restricting the language of the possible answers is not sufficient and we will also pose simple lexicon restrictions for the questions. The questions are not known before hand. Consequently, the LSU will be able to handle most of the users questions but not all of them (without instructing the user to obey the lexicon restrictions).

Having obtained desirable semantic representations we, developed a formalism which represents both the problem of answer verification and answer extraction. We translate the first-order representations of the question and one possible answer into a list of disjunctive logic rules P_A and P_Q. We state the problem of answer verification and answer extraction in terms of finding an answer set to the program $P_A \cup P_Q$.

A QA system which relies on Answer Set Programming for retrieval of the specific answer is presented in [7]. It is a system which receives on input a list of sentences, a "story" over the travel domain, and a question (all in natural language). The system outputs the answers for the questions asked with respect to the story. The natural language sentences of the story are parsed with the Link Grammar Parser. The derivation of the parser is used to build AnsProlog [6] facts. The question is processed separately by the parser and according to the obtained derivation an AnsProlog rule is build. The processing of the rule is done in a much similar fashion to that in [16]—the type of the rule in [7] corresponds to the domain of the rule in [16]. The answers generated are in the answer sets of the program constructed from the rules of the story and the rules of the question. Because the answers sets are nothing else but a set of facts, the answer obtained is indeed the specific answer to the question posed and not a

[5]http://www.mathweb.org

verification of a snippet. We keep the basic idea of this approach but decide to obtain the logic rules that represent the question and the answer by different means.

Natural language is ambiguous and more then one parser derivation is possible for one sentence. The CCG Parser is a statistic parser which is trained over a large corpus to resolve ambiguities by producing only the most probable derivation. We follow this approach to parsing, as well as take advantage of the wide-coverage feature of the semantic representation constructor BOXER (which builds the first-order representations of questions and answers based on the derivation of the CCG Parser). For this reason, we propose a way of obtaining the logic rules P_A and P_Q by directly translating the first-order representations of the possible answer and the question.

The full development and implementation of the LSU for BoB is a large project which out-scopes the time frame of one master thesis. In this thesis we only focus on the tasks of determining adequate semantic representations and reasoning (verifying an answer and extracting a specific answer). This work is organized as follows.

In Chapter 2 we state the general framework of the IQA system we are supporting with logic reasoning (the features of BoB). We state the weaknesses of BoB which can be improved by logic reasoning and we propose the architecture of a Logic Support Unit which can eliminate these weaknesses.

In Chapter 3 we make a brief overview of the task of building semantic representations of natural language and we present the tools we use for constructing the semantic representations. We present the syntactic properties of the first-order formulas obtained by these tools and discuss the semantics of question answering over logic representations.

The majority of the work of this thesis is concentrated in Chapter 4 where we deal with the problem of determining adequate lexicon restrictions for natural language answers and questions. Here we propose the Restricted Lexicons and we show that the first-order representations of sentences and questions built over the allowed lexicons are decidable (for satisfiability and entailment).

In Chapter 5 we first present how to translate into logic rules the first-order representations of answers and questions built in accordance with the defined lexicon restrictions. Then we show how to represent the problem of verifying an answer and extracting a specific answer in terms of ASP.

Lastly, in Chapter 6 we draw our final conclusions, as well as discuss issues that remain for future work.

Chapter 2

IQA supported with Logic Reasoning

In the past years the field of Question Answering, as well as the general field of Natural Language Processing, has extensively relied on statistics and probability as methods for solving its tasks. This approach has one great advantage – scalability. The processing with statistic/probabilistic based algorithms, which rely only on shallow language understanding, allows for various, in continence and size, natural language resources to be processed efficiently.

The disadvantage of the statistic/probabilistic approach is the lack of precision. While for many NLP problems, a "most likely" solution is a "good enough" solutions, the demands of the Question Answering realm go further then what the statistical methods can provide.

In terms of Question Answering this means that the statistically based system will return a most likely answer. The answers retrieved are unspecific (part of a larger text) which has a high probability of containing the information required, however with no possibility to establish the presence of this information for certain or to return exactly the required information and not more then that.

Recent trends in the QA community [9] state that QA systems should support the integration of deeper modes of language understanding as well as more elaborated reasoning schemes in order to boost the performances of current QA systems as well as the quality and the relevance of the produced answers.

Following these trends, we have made an attempt to make a modest contribution by improving a statistically based QA system with a Logic Support Unit (LSU) which aims to improve the precession of a statistically based QA system. As a case study, we considered the chatter-bot BoB, which interactively (in a dialog setting) answers questions over the domain of the university library of the Free University of Bozen-Bolzano.

In this chapter we present the general framework of the Interactive QA System we are supporting with the LSU by presenting the general characteristics of BoB. We will then point out, given the characteristics of BoB, what are the desiderata which are being posed on the LSU. Lastly we will propose an architecture for a LSU for BoB.

2.1 Framework

BoB is a web based application which aims to answer questions over the domain of the university library. It does so by making a shallow analysis of the user utterance by using regular expressions pattern matching. It returns a predicted answer which is most adequate (probable) according to the matched pattern. The performance of the regular expression is enhanced by using an arrangement of all the possible answers in a focus tree. A focus tree [47] is a tree construction in which the knowledge about a domain is structured according to the topics to which it belongs, sub-topics and so on. The question-answer pairs that can be retrieved in the system are positioned at the leaves of the tree. The underlying idea is the presumption that the dialog between the system and the user is led in a topic. The questions that the user asks are presumed to be in the context of the topic. The position in the tree at the given moment of the on-going dialog is to be used as an indicator of the focus of the question.

Following is an overview of the characteristics of BoB.

2.1.1 BoB

[1] BoB is the chatter-bot which assists users of the university library of Bolzano to find *specific answers* to common questions. Classic Question Answering systems tends to return answers in a form of snippets which *contain* the specific answer to the question asked by the user. BoB returns the specific information which is quired by the user, as well as to handle information given by the user (supports a dialog). BoB operates in a constant stimulus-response loop that maps a user utterance to a corresponding system response. The mapping between the user utterances and the system response is done via the shallow form of natural language understanding – regular expression pattern matching extended with boolean operator combinators. In BoB the system answer and pattern pairs are organized hierarchically as nodes in a in a focus tree. It should be noted that at the present BoB is in the preliminary phase of development and only a part of the focus tree in German and English is operational. Once completed, BoB will be able to handle dialog in English, German and Italian.

We will first show a more specific description of the focus tree and the search over it, as done by BoB . Then we will point out the weaknesses of BoB which can be overcame by using a logic reasoning provided by a Logic Support Unit.

2.1.2 Structure of BoBs focus tree and search algorithm

The goal of a focus tree is to predict a coherent continuations of discourse. Each node of the tree represents a possible state in the on-going dialog. The nodes can be seen as an abstract representation i.e. elements that are currently being focused on in the conversations. In the nodes of BoB's tree there are two components: information about the last user input (an abstraction representation) and the corresponding canned-text system response. At the moment BoB does not keep any information on the dialog history between two user utterances.

The structure of BoB's focus tree is inherited from the Stella library information system[2] that has been used for several years by the library of the

[1] The majority of this Section is courtesy of Manuel Kirschner, the developer of BoB

[2] http://www.sub.uni-hamburg.de/informationen/projekte/infoass.html

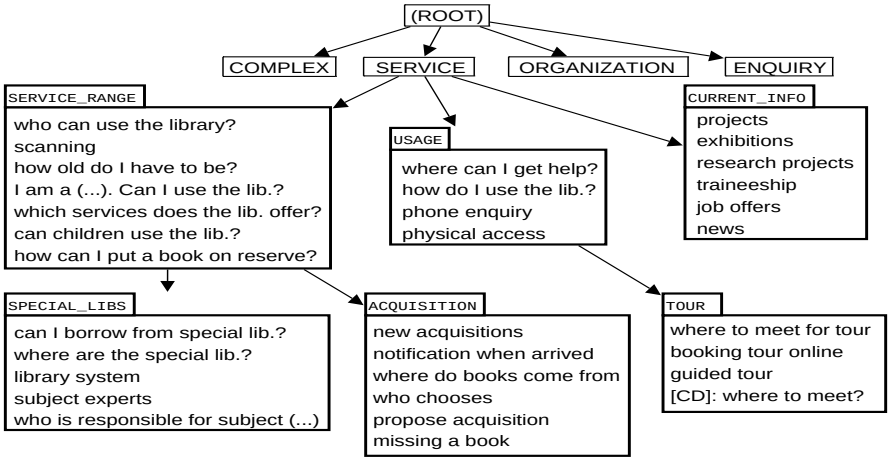

Figure 2.1: A small fragment of the focus tree from BoB: lines of lower-case text denote the leaf nodes that contain a pattern-response pair

University of Hamburg. In their project, a team of librarians has been checking and fine-tuning the focus tree by looking at log files of actual dialogs with Stella users.

The organization of the top nodes in the focus tree are topic nodes that are used to cluster child nodes into topically related subtrees. The topic nodes do not have a regular expression pattern or a system response directly attached to them. All the possible answers that can be retrieved by the system are contained in the leaves. To illustrate how the focus nodes are organized in the focus tree, Fig. 2.1 depicts a small fragment of BoB's focus tree. Names in upper-case letters denote "abstract" topic nodes that are used to cluster child nodes into topically related subtrees; they do not have a regular expression pattern or a system response directly attached to them. It is only the leaf nodes that contain a pattern-response pair.In the figure, every single lower-case text line (inside the large boxes) stands for one leaf node. The names of these nodes should give an intuition about the kinds of user questions that should be matched by the corresponding regular expression pattern.

To provide one example from the BoB focus tree, here are the contents of the leaf node "*how old do I have to be?*" (located under "SERVICE_RANGE" in the focus tree):

Pattern	((from (which|what) age|how old|below .* age|not yet .* old) ((can|must) I)) && (use the library |library use |get .* (library card|account) |sign up)
Response	Everybody above the age of 18 can sign up for a library account.

When the user enters a new query, BoB attempts to find a suitable node in the focus tree in order to return the system response stored there. The search for the adequate topic depends both on the user input and on the previously active focus node. The search begins in the local neighborhood of the previously active node among "related" topics that are likely continuations in a coherent dialog.

A new dialog starts with the root node as current focus node. Then all siblings (sub-topics) of the current node are searched, followed by a search of the descendants a selected sub-topic until a match is found. If a match is not found the algorithm moves one node up in another sub-topic and searches the decedents of the siblings.

2.1.3 Weaknesses of BoB which can be improved by a LSU

A system such as BoB can achieve an overall good efficiency by combining the pattern matching with predicted-topic tree organization of the searched-for answers domain. However, these systems also have certain weaknesses which are inherited from the regular expressions, as well as undesirable constraints which are introduced by the focus tree.

Regular expressions match exact patterns of symbols thus perform a syntactic parsing of the input sentence. In order to correctly identify the topic of the users question (and with that successfully locate the correct answer), it is not sufficient to only look at the constituents of the utterance. Consider for example the question "How can I print the search results?" and the question "How can the printed material of the library be searched?". Both of these questions can be mapped both to the topics "search" and "print". Clearly the first question falls under the topic "print" (the focus of the question is on printing) while the second one falls under the topic "search". Without looking at the semantic roles of the matched lemmas *print* and *search* it is not possible to correctly identify the topic, hence parsing of the question and answers is needed.

The precision of the answer retrieval by BoB depends on the precision of the regular expressions in the leaf nodes. In order for BoB to return a precise answer, the question has to be specifically predicted in the leaf nodes of the focus tree. By using general answers and robust regular expressions the system loses the ability to retrieve the specific information quired by the user. However, by increasing the precision of the regular expressions in the leaves, the entire IQA System losses its robustness. It would be able to perform ideally on the questions it expects, but it would be unable to respond to questions which are not predicted. The case of Stella indicates that building an optimal focus tree is a process that span through several years of work by domain experts.

A too specifically built focus tree introduces another weakness in the BoB system. The required level of specificity varies from question to question and it can happen that the users question is more general then the predicted one. Consequently the full answer to the question can be spread in more then one leaf node. If such a situation occurs the search algorithm will locate both answer leaves but will return only one, although both are needed.

Lastly, even by using the optimal focus tree, this kind of answer retrieval will only return what is a "most likely" answer. For a given question from the user there is no way to verify if what is retrieved certainly answers the question.

In order to cope with these weaknesses we propose that BoB is augmented with a Logic Support Unit. Given a question from a user, BoB will use its search algorithm to traverse the focus tree and retrieve not the most likely, but all the possible answers that match the regular expression. The LSU will take as input from BoB the users question and all the answers which BoB has retrieved. It will build semantic representations of them. By pairing the semantic representation of the question with the semantic representations of each of the possible answers

it will verify or refute one by one each of the possible answer. Once an answer is verified the LSU will extract the specific information quired from its semantic representation. The LSU will translate the specific information back to natural language and return it to BoB. If more then one answer is verified, then the specific information will be extracted from each one of them and returned to BoB. By using the LSU for doing the specific information extraction, BoB's focus tree can be kept robust which will save the long time and effort needed for its "fine tunning".

In the next section we will propose the architecture the LSU which has the necessary capabilities to serve as logic support for BoB. The system can be used as a logic support for any alike Interactive Question Answering system that is able to narrow down the retrieved answers to a small set of most likely answers.

2.2 Logic Support Unit for IQA

In this section we will propose the architecture of a unit which has the necessary capabilities to serve as logic support for BoB. The system is designed inspired by BoB, but it can be used as a logic support for any alike Interactive Question Answering system that is able to narrow down the retrieved answers to a small set of most likely answers.

We will begin by specifying the capabilities which a Logic Support Unit for BoB needs to possess in order to improve the weaknesses of the Interactive Question Answering process presented in the previous section. We will then present a proposal for an architecture of a LSU that satisfies those desiderata.

2.2.1 Desiderata for the Logic Support Unit

The role which the Logic Support Unit for an Interactive Question Answering should play in order to provide with an adequate support for an IQA system implies certain capabilities which the LSU should inevitably possess.

The LSU we are looking for has to have the capability to efficiently (fast and accurate) build wide-coverage **semantic representations** of the question and the answers. The reasons for these demands are the following:

The need for fast processing is directly motivated by the fact that the entire IQA System is a web based application. The LSU, that works in parallel with the IQA System, has to satisfy the speed efficiency demanded from any web application.

The accuracy one of the basic features which the LSU is contributing to the IQA system – without accurate semantic representations the reasoning will not be reliable.

The demand for wide-coverage capabilities of the semantic representation is dictated by the fact that the IQA System we are supporting is accepting questions directly from users. The LSU has to be able to obtain semantic representations from any question posed by the user. Although all the answers that the IQA system can possibly return are known beforehand, those answers are to be provided by domain experts unfamiliar with the semantic representation itself. We want the answers to be created with

the minimum possible guidance, their creation should be simple for the domain experts. Using wide-coverage semantic representation will enable us to reduce the guidelines for creating answers to the minimum. The usage of a wide-coverage semantic representation tool increases the overall portability of the LSU system and makes it easy to be adopted for any domain without changes of the way the semantic representations are being built.

In addition to these demands, the representation chosen has to be such that offers the possibility for efficient reasoning to be implementation over it.

The **reasoning** which the LSU incorporates has to provide a way for a possible answer to be verified for a question given the semantic representations of both. It should also be able to extract the specific information from a once verified answer. Before choosing/creating a reasoning tool a logic formalism as to be found for formally representing the problem of verifying an answer to a question as well as the problem of extracting a specific information from the semantic representation of the answer. Based on the formalism chosen, a powerful and efficient automated reasoning solution should be found which will be able to automate the defined reasoning. This solution should be such that allows for a choice of off-the-shelf available reasoning tool.

The LSU we need has to possess a **background knowledge** about the domain and the world in general represented using the same semantic representation as the question and possible answers. The goal of the LSU is to provide intelligence in the process of retrieving an answer to a question. In order to do so reasoning solely over the possible answers will not suffice. When we intuitively attempt to determine if a given text contains an answer to a question, we need to use not only the information we understand from the text, but also the information about the context of the text (the domain to which the text belongs to). We also need to use general knowledge about the world. Consider for example, the question "*Where can I return the books I borrowed?*" and the possible answer "*The borrowed material can be checked in on the machines or left with the library staff in the library.*" This last sentence answers our question, however, no reasoner will be able to reach this verification without domain knowledge that in the library setting books are material, and to check in or leave material at the desk are synonyms with return.

Lastly, once the reasoner combined with the background knowledge succeeds in verifying an answer and extracting the specific information for a question, this specific information needs to be returned to the IQA and through it to the user. However, this specific answer will be in the shape of semantic representation and we need to return it to the user in natural language. In order to do so, the LSU needs to incorporate a unit which will be able to translate the extracted semantic representation of the requested specific information back into natural language i.e. it needs to contain a **natural language generator**.

2.2.2 Proposed Architecture for the LSU

We will now propose an architecture for a system which has the described capabilities to serve as a Logic Support Unit for an IQA System. The architecture of this Logic Support Unit is shown on Figure 2.2.

The LSU has five components. We will shortly represent each of them before we explain the intended functionality of the entire system.

- a subunit for building semantic ***representations***. We chose to use first-order logic to build the semantic representations of the natural language. To build the first-order representations we will use an off-the-shelf tool (BOXER(combined with the CCG parser).

- a subunit for ***reasoning*** over the built semantic representation (verifying an answer and extracting the specific answer). We have chosen to translate the first-order representation of the question and the answer into logic program rules and use the Answer Set Programming setting to verify that a question is answered, as well as to extract the specific answer from A. For the implementation of the ASP setting we have chosen the state-of-the art ASP solver—the DLV System. The outcome of an ASP solver are *answer sets* which in the Reasoning subunit will be analyzed to determine if the answer processed has been verified and used to extract the specific answer (if such exists).

- a natural language ***answer generator*** which transforms the extracted specific answers semantic representation back to natural language. We assume that this subunit accepts the output from the Reasoning subunit which will be in form of a set of Prolog-like facts. Given these facts and the natural language answer they originate from this unit generates a natural language phrase or a sentence.

- a ***background knowledge*** base which is combined with the semantic representations of the answers for improving the reasoning for verification and written in DLV syntax.

- an *answers repository* which is in the form of a look-up table. Given that all the possible answers which can be retrieved by BoB are known before hand (they are the leaves of BoB's focus tree) we can process them off-line. (The black arrow in the Figure shows the flow of this process). We will then store the built semantic representations in a table. When a possible answer passed on to the LSU, the LSU will search the answers repository for a matching representation. This is more efficient (faster) then building the semantic representation over and over each time a possible answer is retrieved.

The LSU and BoB combined are intended to function in the manner we will next describe. When a users question is received, BoB processes the question by using its regular expressions and the search algorithm over the focus tree to retrieve a list of possible answers to the question. (We considered not more then five possible answers for efficiency reasons.) The question (green block arrow on the figure) is then translated into its first-order representation in the Representation section of the LSU and into DLV rules in the Reasoning section. Let us denote the result with Q. The possible answers, lets assume there are three of them P_1^A, P_2^A and P_3^A (pink block arrow on the figure) are sent to the answer repository for a DLV representation match, which corresponds them, to be retrieved. Assume the corresponding matches are A^1, A^2 and A^3. *For each of the answers* a tuple is formed together with the question and the background

knowledge. In the case we have three answers we will have three tuples $Q \cup A^1 \cup KB$, $Q \cup A^2 \cup KB$ and $Q \cup A^3 \cup KB$. Each of these tuples is sent to the DLV system and the resulting answers sets of each are analyzed for verification and specific information extraction. The results of the analysis (which are in the form of grounded predicate symbols (they are the same as Prolog facts), combined with the answer in natural language to which they belong to , are then sent to the Natural Language Answer Generator (represented with the blue block arrow in the figure) to be translate back into natural language phrase or sentence and returned to BoB (orange arrow on the figure). The answers which failed the verification test will be sent to the natural language generator as a natural language answer unpaired with additional information.

In case none of the possible answers was verified for an answer a final attempt for verification will be made, by sending the set $Q \cup A^1 \cup A^2 \cup A^3 \cup KB$ to the DLV system. This is done to cover the possibility that the answer to the question is spread into more then one possible answers. This tuple is treated like another regular possible answer, namely the reasoning over it and the analysis of its answer sets is done exactly the same as for the other possible answers.

The development of the LSU we here propose contains the four difficult problems of Natural Language Processing: semantic representation, reasoning, generating background knowledge to support the reasoning and natural language generation. The question and answers have to be semantically represented. A reasoning formalism and implementation has to be determined which will meet the demands we pose on the LSU. The reasoning can not be completed without a knowledge base containing the background knowledge of the university library domain. Once an answer is verified and extracted it will be in the form of a semantic representation and before it can be returned to BoB (i.e. the user) the LSU has to translate the (semantically represented) specific answer back to natural language.

The time frame of one master thesis allows us to tackle only one of these four problems. Nevertheless, we have chosen to work on both representation and reasoning because these two problems are co-dependent and for practical purposes one can not be considered without the other. In order to be able to devote our attention to both we have decided to find the most basic solutions for the representations in order to be able to set up the reasoning framework as well. We have also decided (given that this is an application oriented research) to employ as many as possible state-of-the-art available tools. To this end, we use the BOXER of C&C Tools to build semantic representations of natural language (English) and the DLV system to implement the reasoning framework we will propose to be used.

As we mentioned, we chose to use the first-order semantic representation. On the one hand first-order logic is insufficient to cover all the linguistic phenomena of natural language, but on the other it is too expressive for efficient reasoning to be performed over it. In order to cope with the undecidability of the representations we have devoted considerable amount of our attention in this thesis to define *Lexicon Restrictions* on the natural language used. The LSU will function over restricted natural language. These restrictions and the representations which they yield, as well as the achieved results by using the restrictions proposed, will be thoroughly described in Chapter 4.

The remaining of the thesis is concerned with developing the Reasoning

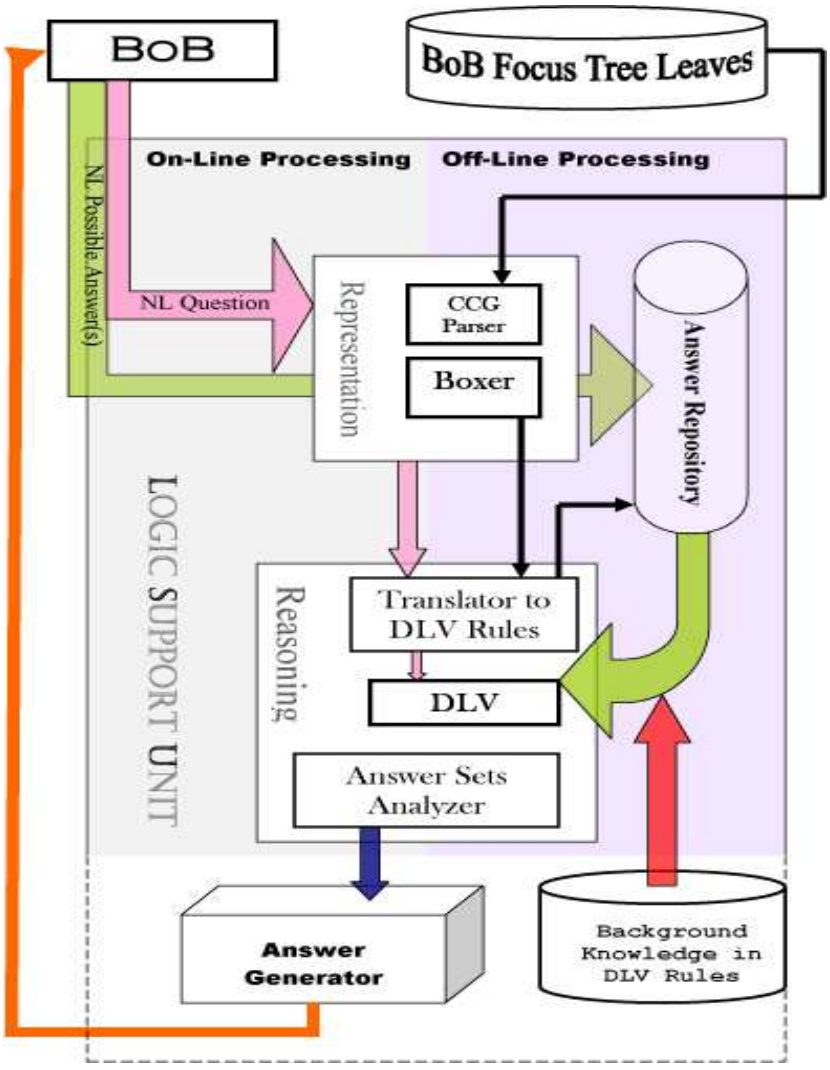

Figure 2.2: The proposed architecture of the Logic Support Unit

Module of the LSU. The work done on this sub-unite is presented in Chapter 5. We assume the existence of a Knowledge Base (KB) written in DLV rule syntax, however the development of such a KB out-ranges the scope of this thesis. We will only briefly consider the properties of the needed Background Knowledge for this LSU in the Section 5.4 of Chapter 5.

The specifics of the Natural Language Generator are outside of the the scope of this thesis.

In the next Chapter (Chapter 3) we present how the first-order representation are constructed, the tool we use for this purpose, as well as the logic formalism for verifying a possible answer (logical entailment) and extracting specific information from a verified answer.

Chapter 3

Semantic Representations of Natural Language

For the LSU to perform efficiently, in particular to offer efficient reasoning, we need to ensure the decidability of the first-order representation for natural language. To do this we determine Lexicon Restrictions for the language used in the sentences of the possible answer and the question over which the reasoning is performed. These Lexicon Restrictions have to be simple to use and with wide coverage to represent most of the natural language the LSU works with.

To discover these Lexicons we studied the mechanism of building the first-order formulas from natural language. In this chapter we present how our first-order representations are built. The goal of this chapter is to serve as a preliminary study for the problems treated in this thesis.

We begin by presenting the underlying theory of formal semantics in natural language, followed by a description of the tool we will use to build the first-order representations. We will also present the syntactic properties of these representations.

The outlined semantic properties of the representations were used as a guide in constructing the Lexicon Restrictions and as a base for developing of the most adequate reasoning formalism for verifying an answer and extracting the specific information from a verified answer.

In the last section (Section 3.3) we present question answering formalisms from related work in QA which can be applied in the case of first-order logic representations. We discuss why these formalisms do not fit our purposes and make an analysis on the semantic properties of the first-order formulas which represent the possible answers. We used the analysis presented in this section as a further guide toward finding an adequate logic reasoning formalism.

3.1 First-Order Representations of Natural Language

First-order logic is well established in Natural Language Processing as the language of choice for performing efficient semantic analysis for many state-of-art systems ([15], [29]). The choice of first-order logics is motivated by its impres-

sive theoretical coverage of linguistic phenomena [39], [26], [15]. The current state of automated deduction offers a range of highly sophisticated inference tools for first-order logic. For any other language with more expressive power (such as second-order or higher-order logic) there simply are no efficient tools available to be used at present.

Our choice is vastly supported by the availability of an off-the-shelf tool which is powerful enough to efficiently build wide-coverage first-order representations for natural language sentences – BOXER combined with the CCG Parser both provided by C&C Tools[1].

In section 3.2 we will in more detail present the tool (BOXER) which we will use, as well as the properties of the first-order representations which are produced by BOXER , but first we will briefly look into the problematic of deriving semantic representations for natural language, focusing more on deriving semantic representations by using Combinatory Categorial Grammar (CCG).

Theory of Formal Semantics

Formal Semantic of natural language was pioneered in the 1960s and early 1970s by Richard Montague, an American mathematician and philosopher who was a student of Alfred Tarski. He stated that there is no essential difference between the semantics of natural languages (as fore example English) and formal languages (as for example predicate logic). In this spirit Montague founded the theory, which after his death was known as Montague Grammar. The Montague grammar is based on formal logic, especially lambda calculus and set theory. Montague Grammar presents the basics of many contemporary formal semantics theories. [2]The central idea of the Montague Theory is that a formal grammar for natural language should be able to be cast in the following form: the syntax is an algebra, the semantics is an algebra, and there is a homomorphism mapping elements of the syntactic algebra onto elements of the semantic algebra. This very general definition leaves a great deal of freedom as to nature of these algebras. The *homomorphism requirement* formalizes one of the most constant features of this theory, over time – the Principle of Compositionality:

> "The meaning of a whole is a function of the meanings of its parts and their mode of syntactic combination."

Relying on the central idea of the Montague Theory, and based on work by Ajdukiewicz, Bar-Hillel proposed a grammar formalism in which every syntactic derivation corresponds to a semantic interpretation. This formalism, Categorial Grammar, has been developed further in many ways. One of its extensions is *Combinatory Categorial Grammar* (CCG) [50].

CCG is a grammatical theory which provides a completely transparent interface between surface syntax and underlying semantics [33]. Each (complete or partial) syntactic derivation corresponds directly to an interpretable structure. This allows CCG to provide an account for the incremental nature of human language processing. The syntactic rules of CCG are based on the categorial

[1]http://svn.ask.it.usyd.edu.au/trac/candc

[2]The brief outline of the origins of Montague grammar, as well as summarization of the basic principles of the classical form of the theory can be found on http://people.umass.edu/partee/docs/MontagueGrammarElsevier.PDF

calculus of Ajdukiewicz [1] and Bar-Hillel [5] as well as on the combinatory logic of Curry and Feys [25].

In CCG, same as in any categorial grammar, words are associated with very specific *categories* which define their syntactic behaviour. The set of syntactic categories is defined recursively building over atomic categories (such as S (sentence), NP(noun phrase), N(noun), PP(Propositional Phrase) etc.). A set of universal rules defines how words and other constituents can be combined according to their categories. In addition to the rules of combinatorial grammar (backward and forward application), CCG also allows for the following rules: Forward and Backward composition, Forward and Backward cross composition, Generalized Forward and Backward composition, Generalized Forward and Backward cross composition, Forward and Backward Type-raising, Forward and Backward Substitution and Forward and Backward Crossing Substitution. For a detailed introduction and elaboration on the rules we direct the reader to [50] or [33].

Combinatory Categorial Grammar (CCG) provides a particularly simple and semantically transparent treatment of extraction (long distance dependency) and coordination. The immediate availability of interpretable predicate-argument structure (or logical forms) even for those constructions is what makes CCG particularly attractive for any application which requires semantic interpretation [33].

One of the main reasons for the natural language to be intuitively regarded as very different from formal languages (such as first-order logic) is its *ambiguity*. Ambiguity arises in natural language analysis when more than one interpretation is possible for a given sentence. The ambiguity may be lexical, structural or semantic. Lexical ambiguity occurs when a lexical entry allows a word more than one possible meaning. Syntactic ambiguity occurs when there are different possible syntactic parsers for a grammatical sentence. Semantic ambiguity refers to the broad category of ambiguity which arises when the meaning of the sentence must be determined with the help of greater knowledge resources. For example, the problem of resolving simple pronominal reference falls under semantic ambiguity [4]. Because of the ambiguity of natural language, it is not sufficient (and often impossible) to return all analysis that a grammar provides for a sentence. Syntactic analysis, or parsing, forms an integral part of any system which required natural language understanding, since it is a prerequisite for semantic interpretation.

To make parsing an efficiently computer processable task, the linguistic community turned toward deriving the *most likely analysis instead of all the analysis* that a grammar provides for a sentence. In recent years, programs that use statistical models to analyze a wide range of natural language sentences with high accuracy have been created (e.g. [23] and [21]).

Today the state-of-the-art statistical parser (CCG Parser) developed for CCG, which offers efficient and robust parsing of real text[3], is available as an off-the-shelf tool trough C& C Tools. The parser uses a grammar derived from CCGBank ([34], [33]) and produces *only one* analysis which is (the most likely interpretation) for the sentence it parses. The parser is highly efficient and it can parse up to 35 newspaper sentences per second. The details on the CCG Parser can be found in [22].

[3]http://svn.ask.it.usyd.edu.au/trac/candc/wiki/Parser

BOXER takes as an input the derivation produced from the CCG Parser and builds first-order representations. We will next present in more detail the process of producing first-order representations by BOXER as well as the properties of the first-order representations obtained. The full information on BOXER (description, usage, availability, examples and demo) can be found on the web page of C&C Tools.

3.2 BOXER

BOXER is a component of C&C Tools developed by Johan Bos. It takes as an input CCG derivations outputted by the C&C tool the C&C parser and uses them to generate semantic representations. BOXER implements a first-order fragment of Discourse Representation Theory, DRT [39] and generates the box-like structures of DRT known as Discourse Representation Structures (DRSs). DRT is a formal semantic theory backed up with a model theory, and it provides large coverage of linguistic phenomena [24]. BOXER implements a first-order fragment of DRT [4]. The DRS's of this DRT are translated to obtain first order logic representations. Before looking at the first-order representations created by BOXER we will first briefly describe these DRS's.

The Underlying Discourse Representation Structures

DRSs are recursive data structures which are defined over a set of first-order variables and a vocabulary which describes predicate symbols and their respective arities. A DRS is comprised of a domain (a set of variables which in the DRT are called *discourse referents*) and a set of *conditions*. Formally:

If $\{x_1, \ldots, x_n\}$ is a finite set of variables and $\{\gamma_1, \ldots, \gamma_1\}$ a finite set of DRS conditions, then the ordered pair $\langle\{x_1, \ldots, x_n\}, \{\gamma_1, \ldots, \gamma_m\}\rangle$ is a DRS.

The DRS conditions are defined by simultaneous recursion using the following clauses:

1. If R is a relation symbol for an n-place predicate and $x_1, \ldots, x_n$ are variables then $R(x_1, \ldots, x_n)$ is a (simple) DRS-condition.
2. If x_1 and x_2 are variables, then $x_1 = x_2$ and $x_1 \neq x_2$ is a (simple) DRS-condition.
3. If B is a DRS then $\neg B$ is a (complex) DRS-condition.
4. If B is a DRS and x is a variable then $x : B$ is a (complex) DRS-condition. $x : B$ means that the discourse referent x is related to the DRS B.
5. If B_1 and B_2 are DRSs, then $B_1 \vee B_2$ and $B_1 \Rightarrow B_2$ are (complex)DRS-conditions.

[4] BOXER implements the (standard) DRT theory as presented in [39] with two exceptions – the Van der Sandt's "presupposition as anaphora" theory and the neo-Davidsonian analysis for events and roles.

The relation symbols are derived from the words in the parsed sentence. Nouns, verbs, adjectives and adverbs introduce one-place relations. Named entities are also represented as one-place relations, for example *Rome* is represented by $rome(x)$.

Verbs are treated as events to which the semantic roles *agent*, *patient* and *theme* are associated. BOXER always represents the verbs both with the unary relation *event* and the unary verb relation over the same variable (for example y). In addition it creates the associated roles as binary relations by making their first argument the variable y shared between the verb and the *event* relation [5]. For example, the verb and associated roles, (as well as the noun and pronoun) for the sentence "*You may use the service*" will be represented by the predicates accordingly: $person(x)$, $service(y)$, $use(z)$, $event(z)$, $agent(z,x)$, $patient(z,y)$. This approach to treating verbs as events is known as the neo-Davidsonian analysis of events and roles.

Verb-roles and prepositions introduce two-place relations. Apart from the unary and binary predicates introduced directly by the words of the language, and the reserved predicates *event*, *agent*, *patient* and *theme*, the full list of other special predicates that appear in the DRSs built by BOXER can be found on the web pages of the C& C Tools. *All the predicates symbols which appear in the DRSs built by* BOXER *are either unary or binary.*

Besides the core DRS language, described above, there are two operators. The first operator is unary (it takes a DRS as an argument) and its purpose is to indicate presupposed and anaphoric information withing a DRS. The second operator is binary (it takes two DRS as arguments) and it is used to indicate a merge between two DRSs. The result of both of these operators is a DRS built over the core DRS language.

Presupposition triggers are the following words: determiners (such as *the*, *both*, *another*, *all*, ...), possessives (such as *my*, *mine*, *hers*, ...), pronouns (such as *I*, *he*, *it*, ...), presuppositional adjectives (such as *other*, *previous*, *new*, ...) and proper names. Presupposition triggers are being treated by BOXER in two ways.

One of the ways of treating presupposition triggers is by accommodating them on an accessible level of DRS. For example, *other country* is represented as a DRS with discourse referents $\{x, y\}$ and DRS conditions $\{\text{country}(x), \text{country}(y), x \neq y\}$. The second way is by resolving presuppositions by binding them to a suitable antecedent. This way possessives and pronouns are represented using one of the following reserved predicates: *person* (for example for *I*, *yours*), *neutrum* (for example for *it*), *female* (for example for *she*, *hers*), *male* (for example for *him*, *he*).

BOXER performs the presupposition and anaphoric resolution on the level of a sentence or across sentences (depending on user selection) over the built DRSs before performing the translation to first-order logic. The precise algorithm for implementation of presupposition resolution can be found in [14]. This approach to handling presuppositions is known as the Van der Sandt's "presupposition as anaphora" theory. The details of this theory are presented in [26]. BOXER builds one DRS for one CCG derivation (one derivation corresponds to one grammatical sentence).

[5] Unlike the (standard) DRT in [39] according to which represents verbs as unary, binary or ternary relations with the semantic roles which are associated with them as arguments of the verb relation

The way BOXER builds the DRS is compositionally by employing the *lambda*-calculus. BOXER creates first-order representations by translating the DRS's. We show the essentials of the compositional semantic representation mechanisms through the viewpoint of the first-order representations because that is the semantic representation that we will work with, as well as for simplicity reasons. The construction mechanism we presented after presenting the translation from a DRS to a first-order logic formulas and the properties of the obtained formulas

From Discourse Representation Structures to first-order formulas

We will present here the translation function f° for a DRT and a DRT-conditions as given in [11]. BOXER recursively, bottom-up, implements the translation rules of f° over a DRS and builds one closed first-order logic formula for each DRS.

The definition of the translation function f° follows.

Assume that D is a DRS with a universe $x_1, \ldots, x_n$ and a set of conditions $\gamma_1, \ldots, \gamma_m$. γ_i can be either a simple or a complex DRS-condition. The translation of D to first-order logic is defined as:

1. $f^\circ(D) = \exists x_1 \cdots \exists x_n (f^\circ(\gamma_1) \wedge \cdots \wedge f^\circ(\gamma_m))$

2. If γ_i is a simple DRS-condition then:
   $f^\circ(\gamma_i) = f^\circ(P(x_1, \ldots, x_n)) = P(x_1, \ldots, x_n)$
3. If γ_i is a complex DRS-condition and D_1 and D_2 are DRSs then:
   (a) $f^\circ(\neg D_1) = \neg(f^\circ(D_1))$
   (b) $f^\circ(D_1 = D_2) = (f^\circ(D_1) = f^\circ(D_2)$
   (c) $f^\circ(D_1 \vee D_2) = (f^\circ(D_1) \vee f^\circ(D_2)$
   (d) If D_1 has the universe $y_1, \ldots, y_k$ and the conditions $\gamma'_1, \ldots, \gamma'_l$, and if D_2 has the universe $z_1, \ldots, z_p$ and the conditions $\gamma''_1, \ldots, \gamma''_q$ then:
   $f^\circ(D_1 \Rightarrow D_2) = \forall y_1, \ldots, \forall y_k (f^\circ(\gamma'_1) \wedge \cdots \wedge f^\circ(\gamma'_l)) \Rightarrow \exists z_1 \cdots \exists z_p (f^\circ(\gamma''_1) \wedge \cdots \wedge f^\circ(C''_q))$
   (e) If x is a discourse referent, then:
   $f^\circ(x : D1) = f^\circ(D')$, where D' is a DRS obtained from the DRS D when x is added to the universe of D and $event(x)$ or $proposition(x)$ (depending on whether the DRS D was representing an event or a proposition) is added to the conditions of D.

□

Properties of the first-order formulas

Here we will outline the properties of the first-order representations which are obtained from BOXER.

We denote with ψ the first-order formulas which are produced by BOXER. According to this translation (step (b)), equality will appear in ψ if and only if there exists the equality complex DRS-condition in the translated DRS. Because

the properties of the formulas with equality differ from those of the formulas without equality, certain notions should be given regarding to the presence and properties of the equality relation conditions in the DRS's.

The equality conditions in a DRS built by BOXER can be deriving from only two sources.

In BOXER the equality relation is used to denote the verb *is* when it stands as an indicator that one item (object, property or event) is the same as another. For example, in the sentence "*The books are the material we used.*", the equality DRS-conditions used will be $x = y$ (where $book(x)$ and $old(y)$). (Note that when *is* is used to assign a property to a noun, as in for example *The sky is blue.*, there will be no equality in the semantic representation.) The verb "is" that is represented with the equality DRS-condition in fact (speaking in the language of databases) is the one which conveys the meaning of an "isa" relation between concepts.

The second source is the underlying DRT. In DRT the equality, as a relation between variables, exists to indicate anaphoric and presupposed information, same as in the previously presented example of the representation of *other country*.

However, we will only translate DRS's in which the presupposition and anaphoric information has already been resolved. Consequently, all the equality in ψ will be derived from the representation of the verb *is*.

Based on this, once the presupposition and anaphoric resolution is performed the resulting DRS's will only contain equality relations which correspond to the meaning of the verb *is*. BOXER represents equality with the binary predicate `eq` due to the Prolog syntax for DRS it employs and we will treat this predicate symbol `eq` as a special representation which indicates the "isa" relation. (It would be fairly simple, although not really necessary, to replace all the `eq` predicates with `isa` predicates.) By preserving the specific semantics of the `isa` predicates by adding additional rules in the knowledge base of the Logic Support Unit (see Section 5.4), we can consider that the DRS's fed for translation to first-order formulas does not contain equality DRS-conditions.

We can now state the following properties of ψ.

- Based on step (1) of the translation function f° we can conclude that the ψ will be an existentially closed conjunctions of sub-formulas, namely they will be of shape:
$$\psi = \exists \mathbf{x}(\varphi_1(\mathbf{x}) \wedge \cdots \wedge \varphi_n(\mathbf{x}))$$

- Because all the simple conditions of the translated DRS are unary and binary and f° does not alter them in any way including their arity, all the predicate symbols in ψ will be with a bound arity: either unary or binary.

- Because we treat all equality DRS-conditions after resolution was performed as special predicates, the signature of ψ will not contain equality.

- Because the language of the translated DRS contains no function symbols and constant symbols and f° does not introduce function symbols and constant symbols the signature of ψ will not contain function symbols and it will not contain constant symbols.

In the next section we will show, using the first-order language, how the semantic representations are compositionally constructed.

Building the Semantic Representations

BOXER uses λ-calculus to compositionally construct the semantic representations. As mentioned, it does so over the DRS language, however, for our purposes, we will present this same mechanism using the language of first-order logic (as given in [34]). The actual λ-DRS approach used by BOXER can be found in [15]. We assume that the reader is familiar with the terminology and the basics of λ-calculus. The unfamiliar user can consult for example [8].

Consider an example lexical entries (categories)(in angle brackets we indicate the syntactic categories):

$$\texttt{a}\,\langle NP/N\rangle\,\texttt{librarian}\,\langle N\rangle : \lambda p.\exists x(librarian(x) \wedge p@x)$$

$$\texttt{works}\,\langle S\backslash NP\rangle : \lambda y.\exists e(work(e) \wedge agent(e,y))$$

where the @ denotes functional application, the variable p marks the missing information provided by the verb phrase, while the variable y marks the missing information provided by the noun phrase.

Combinatory rules project lexical categories such as *a*, *librarian* and *works* onto derived categories such as *a librarian*. In the above example, using (backward) functional application, the two categories yield the following expression:

$$\lambda p.\exists x(librarian(x) \wedge p@x)@\lambda y.\exists e(work(e) \wedge agent(e,y))$$

β-conversion is the process of eliminating all occurrences of functional application by substituting the argument for the λ-bound variables in the functor. β-conversion turns the previous expression into a first-order translation for *A librarian works.*:

$$\exists x(librarian(x) \wedge \exists e(works(e) \wedge agent(e,x))).$$

The output of the CCG parser is a tree representing a CCG derivation, where the leaves are lexical items and the nodes correspond to one of the CCG rules. Mapping the CCG derivation into a semantic representation consists of the following tasks:

1. assigning semantic representations to the lexical items;
2. reformulating the combinatory rules in terms of functional application;
3. dealing with type-raising and type changing rules;
4. applying β-conversion to the resulting tree structure.

We will shortly explain how these tasks are handled.

Lexical items are ordered pairs consisting of the CCG category and a lemmatied wordform. This information is used to assign a λ-expression to the leaf nodes in the tree. For most open-class lexical items BOXER uses the lemma to instantiate the lexical semantics. This is illustrated by the following two examples (intransitive verbs and adjectives):

$$\langle S\backslash NP, \texttt{walk}\rangle = \lambda q\lambda u.q@\lambda x.\exists e(walk(e) \wedge agent(e,x) \wedge u@e)$$

$$\langle N/N, \texttt{big}\rangle = \lambda p\lambda x.(big(x) \wedge p@x)$$

For closed-class lexical items the lexical semantics is spelled out for each lemma individually, as in the following two examples:

$$\langle (S\backslash NP)\backslash (S\backslash NP), \texttt{not}\rangle = \lambda v\lambda q\lambda f.\neg((v@q)@f)$$

$$\langle NP/N, \texttt{all}\rangle = \lambda p\lambda q.\forall x(p@x \rightarrow q@x)$$

The second task deals with the combinatory rules. The CCG formalism that BOXER uses, employs the following rules: forward and backward functional application (FAPP, BAPP), generalized forward composition (FCOMP), backward composition (BCOMP), generalized backward-crossed composition (BCROSS), type-raising (TYPERAISE) and type-changing (TYPECHANGE). (There is also a coordination rule conjoining categories of the same type.)

FAPP(x,y) = (x@y)

BAPP(x,y) = (y@x)

FCOMP(x,y) = $\lambda\pi$.(x@(π@y))

BCOMP(x,y) = λu.(y@(u@x))

BCROSS(x,y) = $\lambda\pi$.(y@(x@π))

The type-raising and type-changing rules are dealt with by looking up the specific rule and replacing it with the resulting semantics. For example, the rule that raises the category NP to S/(S\NP) converts the semantics as follows:

$$TYPERAISE(NP, S/(S\backslash NP), x) = \lambda v\lambda e((v@x)@e)$$

The following type-changing rule applies to the lexical semantics of categories of type N and converts them to NP:

$$TYPECHANGE(N, NP, y) = \lambda p.\exists x(y@x \wedge p@x))$$

Tasks 1-3 are implemented using a recursive algorithm that traverses the derivation and returns a λ-expression. Task 4 reduces the λ-expression to the target representation by applying β-conversion. In order to maintain correctness of this operation, the functor undergoes α-conversion(renaming all bound variables for new occurrences) before substitution takes place. β-conversion is implemented as presented in [11].

Semantic Representations of Questions

The semantic of questions is different then the semantics of "declarative" sentences, hence a bit more attention should be given to the representations of questions we will build using BOXER .

A question is a linguistic expression used to make a request for information. We will distinguish between two types of questions:

1. *WH Questions.* These are questions which are formed out of the following WH-phrases (question words): `who`, `which`, `what`, `where`, `why`, `when` and `how` in the role of a question word.

2. *Boolean Questions.* These are questions that do not contain a question word and whose answer is either `yes` or `no`.

In this thesis we will not consider questions which have more then one WH-phrase as a question word. For example, questions like *"Where and how can I print my search results?"* need to be separated in two questions before being processed. Questions that have one WH-phrase as a question word and a relative pronoun as it is the case in *"Where is the book which I ordered?"* will be handled by our system.

The CCG parser (paired with Boxer) outputs questions as queries of the form[6]:

$$\{x \mid \underbrace{\exists \mathbf{y}(\varphi(\mathbf{y}) \wedge D(x) \wedge \exists \mathbf{z}(\psi(x, \mathbf{y}, \mathbf{z})))}_{\alpha}\} \tag{3.1}$$

where x represents the answer to the question, D is the domain of the question, ψ represents the body of the question, and φ represents the knowledge that is presupposed by the user posing the question. As mentioned, we will consider wh-questions and boolean questions. In the first case, if the wh-phrase is *where*, *who/whom*, *how*, *when*, *what*, or *why*, then D refers to *location*, *person*, *manner*, *unit_of_time*, *thing* and *reason*, respectively. If the wh-phrase is *which*, then the domain is the head noun of the noun phrase. For example, in the question *"Which services are offered by the library?"* the domain is *services*. If the question is a Boolean Question, then the domain of the question is empty. In both cases the body of the question contains a conjunction of conditions.

Apart from the building of the domain predicate, the representation of the question (the formula α) is built exactly as in the case of the "declarative" sentences.

For instance, the question *"Who may use the Interlibrary Loan service?"* is represented as follows:

$$\{x \mid \exists y_1 \exists y_2 (loan(y_1) \wedge interlibrary(y_2) \wedge service(y_2) \wedge nn(y_1, y_2)$$

$$\wedge person(x) \wedge \exists z (use(z) \wedge event(z) \wedge agent(z, x) \wedge patient(z, y_2)))\}$$

(The special predicate $\mathtt{nn}(y_1, y_2)$) is assigned to bind two nouns which participate in the same noun phrase.)

Linguistic phenomena not covered by BOXER semantic representations

We found that BOXER is a very powerful and efficient semantic representation tool, but there are certain semantic phenomena which it does not covers.

BOXER currently does not deal with the resolution of pronouns. Consider for example the first-order representation of the sentence *I took my book.*:

$$\exists x_1 \exists x_2 \exists x_3 \exists x_4 (person(x_1) \wedge event(x_2) \wedge take(x_2) \wedge agent(x_2, x_1)$$

$$\wedge book(x_3) \wedge person(x_4) \wedge of(x_3, x_4) \wedge patient(x_2, x_3)))$$

[6]Boxer actually outputs such open FOL formulas as closed formulas in which the free variable is universally quantified and the second conjunction is represented by an implication.

To resolve the pronouns in this sentence is to indicate that $person(x_1)$ and $person(x_4)$ are the same person, i.e. all x_4 should be replaced by x_1.

Ellipsis(the row of three full stops (... or . . .) or asterisks (***) which indicates an intentional omission)is also not being handled by BOXER . A proper analysis of tense and aspect is not given and also there is no distinguishing between distributive and collective readings of plural noun phrases[7].

The statistical parser we used for CCG and hence BOXER does not handle scope ambiguities. A scope ambiguity is a type of semantic ambiguity which occurs when two quantifiers or similar expressions (as for example a negation and a disjunction) can take scope over each other in different ways in the meaning of a sentence[8]. Consider for example the sentence "*Every user borrows a book.*". The more prominent meaning of this sentence is that for every user, there is a book, and it's possible that each user borrows a different book. But the sentence also has a second possible meaning, which says that there is one particular book which is borrowed by every user. There are two representations in first-order logic corresponding to each meaning:

$$\forall x(user(x) \rightarrow \exists y \exists z(book(y) \wedge borrow(z) \wedge event(z) \wedge agent(z,x) \wedge patient(z,y))$$

$$\exists y(book(y) \wedge \forall x(user(x) \rightarrow \exists z(borrow(z) \wedge event(z) \wedge agent(z,x) \wedge patient(z,y))))$$

BOXER always assigns the first interpretation in these cases.

Modal verbs, such as "may", "should", "could" etc., are not represented by BOXER . Their presence in the sentence (and question) is ignored. The semantic representation built for the sentence "*I should borrow a book.*" is exactly the same as the semantic representation built for the sentence "*I must borrow a book.*":

$$\exists x_0 \exists x_1 \exists x_2(person(x_0) \wedge book(x_1) \wedge borrow(x_2) \wedge event(x_2)$$
$$\wedge agent(x_2, x_0) \wedge patient(x_2, x_1))$$

As mentioned before, lexical semantics for closed-class lexical items is spelled out for each lemma individually. The closed-class does not contain the lemma "*without*" and its proper semantic representation. The lemma without should have a negation in its semantic representation.

We also find that there should be a "spelled out" semantic representation for the lemma *exception*. At present it is being treated the same as any noun phrase, however for the purposes of reasoning it is important that there is a semantic representation of this lemma.

Lastly it should be made clear that BOXER only builds semantic representations over English.

3.3 Reasoning Formalisms for IQA

We have shown how semantic representations can be built using BOXER , as well as the (syntactic) properties of the first-order formulas obtained.

[7] http://www.isi.edu/∽hobbs/metsyn/node9.html

[8] http://www.coli.uni-saarland.de/projects/milca/courses/comsem/xhtml/SEC_CLLS-SCOPE-INTRO.xhtml

The primary goal of the Logic Support Unit is to verify (by means of logic) if a question (represented with the first-order formula Q) is certainly answered by a possible answer (represented with the first-order formula A). To this end, a logical formalism is needed to represent the relation which holds between a question and a text (possible answer) which answers it.

In Chapter 1 we mentioned one formalism which is fitting to the first-order representations for the possible answer and question we obtain from BOXER. We state this formalism as it is presented in [16].

Definition 3.3.1. Let A be a first-order representation of an answer. Let $D(x)$ be the domain of the question, as in the representation 3.1 and $B(x)$ be the body of the question. $B(x)$=$\varphi(\mathbf{y}) \wedge \psi(x, \mathbf{y}, \mathbf{z})$, φ and ψ are the body and the context of the question from representation 3.1. The following propositions are stated:

1. $\forall x[D(x) \Rightarrow B(x)] \wedge A$
2. $\exists x[D(x) \wedge B(x)] \wedge A$
3. $\exists x[D(x) \wedge \neg B(x)] \wedge A$
4. $\neg\exists x[D(x) \wedge B(x)] \wedge A$

This answer verification formalism (given in Definition 3.3.1) requires for satisfiability testing of the above formulas. The satisfiability testing can be implemented with a theorem prover paired with a model builder. Each of the propositions is sent to the theorem prover which attempts to find a proof of inconsistency, and a model builder which attempts to build a model for the proposition.

This formalizing approach offers many choices for implementation (the field of first-order theorem proving offers many efficient tools), however it only formalizes the verification of the answer and not the extraction of the specific answer. To find a way to formalize both we considered some work in semantics of questions.

Work on the semantics of questions ([32], [45]) has argued that the formal logic relation between a question representation Q and a text that answers it A (referred to as *licensing* by [32] and *aboutness* by [45]) is the relation of logical entailment. Using this notion, we state a formalism which can offer both for the possibility to verify an answer and to extract a specific answer for the verified answer.

Definition 3.3.2. Given a first-order representation A of a natural language text T^{NL} and a first-order representation Q of a natural language question Q^{NL} we will say that the text T^{NL} answers the question Q^{NL} if and only if the entailment $A \models Q$ holds. We will call this entailment relation which connects the question with its answer *the question answering entailment*. For Q being a question represented with:

$$Q = \exists \mathbf{y}(\varphi(\mathbf{y}) \wedge D(x) \wedge \exists \mathbf{z}(\psi(x, \mathbf{y}, \mathbf{z}))) \tag{3.2}$$

the variable assignment for x which satisfies Q is a *specific answer* to Q.

The entailment relation states that all the models of the answer must satisfy the question as well. These models of the question will each contain one assignment for the specific answer x. All the x which are specific answers to a question are the union of the assignments for x in the models for A.

In Chapter 4 we propose restrictions for natural language which guarantee that the entailment between the first-order representations of the possible answer and question from this language, $A \models Q$, is decidable. However, deciding the entailment is not enough and we need to build all the models of A to use the formalism for retrieving specific answer. Currently, to the best of our knowledge, there is no tool which we can use to build all these models. To circumvent this lack of adequate tools, in Chapter 5 we re-formalize the problem of verifying an answer and extracting a specific answer in terms of Answer Set Programming (ASP). In the remaining of this chapter we show our observations on the semantic of the first-order formulas which represent the possible answer. We used these observations as a guide in developing the ASP formalisms for answer verification and specific answer extraction.

In the remaining of this section we discuss the model semantic properties of the first-order formulas A which are built as representations to the natural language possible answer.

Models of the representation A

We have observed that to determine if the question answering entailment holds, it is not necessary to consider all the models of A. Instead of for A, it is sufficient to check if the question answering entailment holds for a formula A^S which is equi-satisfiable to A and is obtained from A by Skolemzation. The reason for this is that the first-order representations admit models which are redundant to be considered from a natural language point of you. We will show why we consider A^S to be better a better choice for representing the possible answer by looking into the properties of the models of A.

Let φ be a first-order representation of natural language text. A model for a φ would be a structure which consists of a finite set of individuals (the domain of the model) and an assignment function $\mathcal{I}$ which maps each variable from the representation to an individual of the domain.

In formal semantics of natural language one place predicates (also referred to as properties) are seen as sets of individuals represented trough constant symbols. The interpretation for a unary predicate symbol P will be the set of individuals that have the property P.

The interpretation of binary predicate symbols is seen as sets of ordered pairs of individuals (relations). For a binary symbol Q the interpretations are all the ordered pairs of individuals which stand in a relation Q.

The model can be represented as a set of grounded properties and relations, for example $\langle P(a), P(b), \ldots, Q(a,b), Q(c,d), \ldots \rangle$.

The assignment function $\mathcal{I}$ in the case of φ must be injective because the formulas outputted by BOXER are resolved for presupposition and anaphora. Consequently, each variable denotes a unique individual and two variables can not be assigned the same individual from the domain. Consider the Example 3.3.1 and its corresponding first-order representation φ.

Example 3.3.1. "*The person runs.*"

$$\varphi = \exists \texttt{x} \exists \texttt{y}\ (\texttt{person(x)} \wedge \texttt{run(y)} \wedge \texttt{event(y)} \wedge \texttt{agent(y,x)}).$$

In the case when $\mathcal{I}$ is not injective, the formula φ can admit a model in which the variable x is assigned to a and also the variable y is assigned to a. In this model, such an assignment would indicate that a is a person, but also that a is the event run. However, this does not make sense and it can not be the intended intuitive meaning of the sentence "The person runs."

To express that φ should only be satisfied if and only if $\mathcal{I}$ is injective, the formula $\forall x \forall y\ x \neq y$ should be conjuncted to φ, however BOXER does not do this and hence φ will admit models in which the assignment function $\mathcal{I}$ is not injective. We do not want to take into consideration these models for the question answering entailment.

The domain of a model for φ consists of a nonempty set of constant symbols (denoting the individuals). These constant symbols will not be rigid designators. This conclusion derives from the following properties of the BOXER derived formulas.

Recall that the formulas produced by BOXER do not have constant symbols in their signature. All the verbs, nouns, adjectives and adverbs are represented with unary predicate symbols and all the variables must appear bound by at least one of these unary predicates. In these representations personal nouns are "lifted" to unary predicate symbols as well and they are not represented with constant symbols. Consider the Example 3.3.2 and its corresponding first-order representation.

Example 3.3.2. "*The person Marija runs.*"

$$\exists \texttt{x} \exists \texttt{y}\ (\texttt{person(x)} \wedge \texttt{marija(x)} \wedge \texttt{run(y)} \wedge \texttt{event(y)} \wedge \texttt{agent(y,x)}).$$

The personal name *Marija* does not get represented with a constant, but with a predicate symbol which shares a variable with the noun it belongs to($person$). Consequently, the individual represented with the variable x will be, interpreted with a symbolic name constant and x will not be *the person marija*, but an individual with the property person and the property *marija*. The set $\{\langle marija(a), person(a), run(b), event(b), agent(b,a))\rangle\}$ and $\{\langle marija(c), person(c), run(d), event(d), agent(d,c))\rangle\}$ will in fact be equivalent.

The constant symbols in the domain are symbolic names for the individuals which have the properties represented with the unary predicates. The individual is identified by the unary predicates it belongs to (it grounds) and not by the symbolic name it has been assigned in the model.

Recall that φ is an existentially closed formula. In first-order logic semantics, for a formula $\exists x P(x)$ to have a model (be satisfied), there has to be at least one (but possibly more then one) interpretation for x for which $P(x)$ evaluates to true. This interpretation of the existential quantifier is too general for the intended semantics of natural language (representations).

In BOXER representations the existential quantifiers are introduced trough the verbs and their associated semantic roles, as well as trough the determiners of the noun phrase. The representation of a sentence will always be existentially closed because there is always a verb in the sentence. However, the meaning of

these existential quantifiers is to indicate the presence of exactly one individual x. Consider the Example 3.3.3and its corresponding first-order representation.

Example 3.3.3. "*I borrowed the books.*"

$\exists$x$\exists$y$\exists$z (person(x)$\wedge$book(z)$\wedge$borrow(y)$\wedge$event(y)$\wedge$agent(y,x)$\wedge$patient(y,z))).

The intended meaning of the sentence "*I borrowed the books.*" is that there is one event *borrow* which is executed by exactly one person (indicated by "I") over exactly one object "the book". Which exactly are these individuals, we do not know. For this the representation of this sentence to be true in a model[9] the should be one interpretation for each variable bound by these existential quantifiers. More then one interpretations are superfluous and not in accordance of the intended semantic of natural language[10].

Obtaining A^S

To determine if the question answering entailment holds between Q and A, we do not want to take into consideration the models allowed for the representation A which do not make sense as an intended meaning for the natural language which A represents. For this reason, instead of for the models of A we want to look for specific answers in the Herbrand models of the equi-satisfiable formula A^S, obtained by Skolemizing A. The Herbrand models of A^S will be a subset of the models of A and will only allow models which make sense to be considered as intended semantics of the natural language which is represented with A. (For the unfamiliar reader, the basic definitions of Herbrand Interpretations we give at the end of this section)

The Skolemization procedure we will use is the following [43]:

1. Replace in A all sub-formulas of form $\exists \mathbf{x}\psi(\mathbf{x})$ and which are *not in the scope of any universal quantifier* with the sub-formula $\psi(\mathbf{c})$. The constants $\mathbf{c}$ (also referred to as Skolem constants) must not appear in the signature of A. Let us name the resulting formula A^{e-}.

2. If $\exists \mathbf{y}\phi(\mathbf{y})$ is a sub-formula in A^{e-} which is under the scope of the quantifier expression $\forall\ z_1 \cdots \forall\ z_n$, replace it with the sub-formula $\phi(f(z_1, \ldots, z_n))$. Replace all such sub-formulas in A^{e-}. All the function symbols f (also referred to as Skolem functions) used in the replacements must not appear in the signature of A^{e-}.

The result of this procedure is A^S. A^S is equi-satisfiable to A. For proof see for example [43]. It is evident that all the Herbrand Models of A^S will be a subset of the models of A—each Herbrand Model of A^S can be transformed into a model of A by assigning to the variables in A the corresponding Skolem constants or Skolem terms which replaced them during Skolemization.

In the first step of the Skolemization we replace some existentially quantified variables with Skolem constants. By doing this we are choosing one possible

[9] the truth conditions are in accordance to those proposed by Tarski. For more details on truth conditions in natural language semantics we refer the reader to [39].

[10] Because BOXER does not represent plurals, the sentence "The students borrow books" will also have the same representation as in Example 3.3.3. In this case the intended semantics of the sentence is that there is one group of *students* which performs the one event of *borrow* over the group of *books*. However, this information will be lost.

variable assignment out of many possible variable assignments. However, one variable assignment that satisfies the formula is sufficient to be considered. Recall that which constant symbol represents the individual from the domain is not important because there were no prior constant symbols in the formula and the constant symbols are not rigid designators. By "naming" the individuals with the Skolem constants we do not reduce the number of possible models "meaningful" models for A^S and hence for A (because A^S is a subset of A).

The first step of the Skolemization also ensures that the existentially quantified variables are only interpreted with one individual, by assigning the symbolic name to the individual. An interpretation in which a is assigned to y in $\exists y$ *run(y)* and also to x in $\exists x$ *person(x)* (from Example 3.3.1) will never be a Herbrand Model for A^S.

In the case of the second step of the Skolemization, the remaining existentially quantified variables are replaced with Skolme function symbols. The latter are introduced to preserve the connection between the universal and existential quantifier. We will give the intuition behind this introduction of function symbols as well as the properties of the Skolem functions in natural language representations trough an example.

Example 3.3.4. *Every student borrows a book.*

$$\forall x \rightarrow \exists y \exists z(\texttt{student}(x) \wedge \texttt{book}(z) \wedge \texttt{borrow}(y) \wedge \texttt{event}(y) \wedge \texttt{agent}(y, x) \wedge \texttt{patient}(y, z)).$$

$$\forall x \rightarrow (\texttt{student}(x) \wedge \texttt{book}(f^1(x)) \wedge \texttt{borrow}(f^2(x)) \wedge \texttt{event}(f^2(x)) \wedge \texttt{agent}(f^2(x), x) \wedge \texttt{patient}(f^2(x), f^1(x))).$$

In this example, the intuitive meaning of `book(`z`)` is the book which is being borrowed particularly by `person(`x`)`. f^1 can intuitively be interpreted as "which belongs to", because the conveyed meaning of the term `book(`$f^1(x)$`)` is "book which belongs to the individual x".

In this case (when the existential quantifier is under the scope of the universal quantifier) we can not simply assign a symbolic name to z because the existence of an assignment for the variable z in a model depends on the existence of an assignment for the universally quantified variable x. We do not know if there is an individual with the property book in the domain of the model. An interpretation for the Skolemized formula 3.3.4 will exist even if there are no individuals with the property *person* in the domain – if there is no person there need not be a book. By replacing the existentially quantified variable x with a constant symbol instead of with a function, we would force the presence of an individual with the property *book* in the domain even when there are no individuals with the property person in it.

In first-order logic semantics, the Skolem functions can have arbitrary interpretations. This means that we can chose to evaluate f^1 (from Example 3.3.4) by $f^1(x) = 3$. However, the Skolem functions $f^1(x)$ has to be evaluated to an individual which has the property *book*. Any other evaluation will not satisfy the formula. This means that the *domain of the Skolem functions is bound* to the domain of the predicate symbol under which it appears – the domain of f^1 can only be a subset of the set $\{z \mid \texttt{book}(z) = \text{true}\}$.

A first-order formula A (without equality) has a model if and only if it has a Herbrand Model (the Herbrand Theorem). We find The Herbrand Interpretations of A^S are closest to the interpretations of predicates symbols in formal semantics ; one Herbrand Model differs from another Herbrand model (of the same formula) if it has a different set of grounded predicate symbols. In Herbrand interpretations, the constant symbols and the function symbols are not interpreted. This is convenient when finding models for natural language representations because, all the function terms in A^S are grounded Skolem terms and those are nothing else but yet another symbolic name for the individuals of the domain. We do not need to interpret the function symbols. For example, in $\texttt{book}(f1(a))$, the term $f1(d)$ is an evaluation for x , namely a symbolic name for the individual to which x is evaluated to.

For the unfamiliar reader we present here the definitions of Herbrand Interpretation.

Definition 3.3.3. A **Herbrand Universe** for a given formula φ is the set of all constant symbols $\{c_1, c_2 \ldots\}$ which appear in φ and the set of all terms which can be built from these constant symbols using all the function symbols $\{f_1, f_2 \ldots\}$ from φ. If the formula contains no constant symbols one constant symbol is added to the universe. The Herbrand Base is finite if the signature of φ does not contain function symbols.

Corollary 3.3.1. *The Herbrand Universe of φ is finite if the signature of φ does not contain function symbols.*

Definition 3.3.4. A **Herbrand Base** for a given formula φ is the set of all possible grounding of the predicates symbols of φ using the constants from the Herbrand Universe.

Corollary 3.3.2. *The Herbrand Base of φ is finite if the Herbrand Universe of φ is finite.*

Definition 3.3.5. A **Herbrand Interpretation** for a given formula φ is the subset from the Herbrand Base. A **Herbrand Model** for φ is the Herbrand Interpretation for which the formula φ is satisfied.

Chapter 4

Fragments of Natural Language

Although first-order logic is not expressive enough, from a natural language processing perspective, to cover all the linguistic phenomena (like for example, tense representation, distributive and plurals), from the computational perspective it is "too expressive". The high expressiveness of first-oder logic implies undecidability of reasoning over it. More precisely, deciding satisfiability and entailment, which we will use as a formalism to model our reasoning task (verify a possible answer for a question), are undecidable.

When dealing with the task of providing logic support for an *interactive* question answering system it is of essence that the supporting system is able to efficiently *decide* if the answer recovered by statistical means *certainly* answers the question posed by the user. If the logic support unit is unable to guarantee decidability over this certainty, then the whole logic support unit loses its purpose because the most *probable* answer is already being retrieved by statistical means. Consequently, it is essential that we ensure termination (decidability) in any implementation of a first-order reasoning (inference) system i.e. find a way to work only with decidable semantic representations. Because the semantic representation depends on the syntax of the sentence, to ensure decidability of the first-order formulas, we have decided to pose restrictions over the natural language used to build the answers and restrictions over the natural language allowed for posing a question. These restrictions will reduce of the coverage boundaries of the Logic Support Unit.

We know before hand all the possible answers which can be retrieved by the IQA System and we can write them in a restricted language. However, we want to allow for easy and natural creation of the possible answers by the domain experts and for this purpose the posed restrictions should be as loose as possible, as well as easy understandable. In the case of the language of the questions, we want to allow the users of the IQA system to pose their questions without difficulty, and possibly without complicated (if any) instructions. Hence, it is essential to also keep the restriction rules minimal and simple.

Decidable fragments of natural language (fragments which yield decidable first-order logic formulas) have been presented in the work of Ian Pratt and Alan Third [49].

Pratt and Third build their semantic representations according to Montague theory using a context-free grammar which they define. They ensure that their fragments yield decidable semantic representations by defining lexicons which contain only certain kinds of verbs (transitive and ditransitive) and controlling the presence of pronouns (relative and reflexive). With these restrictions they control the number of variables in the obtained semantic representations.

The approach of Pratt and Third is interesting because the restricted natural language which they define is still wide enough to potentially satisfy our needs.

Our (BOXER derived) semantic representations and Pratt and Third's semantic representation share the underlying theory (Montague Theory), but BOXER represents verbs as events without making a difference between intransitive, transitive and ditransitive verbs. With the BOXER approach (the neo-Davidsonian analysis of events) of representing verbs, the type of verb is represented by the number (and type) of associated semantic roles. The introduction of semantic roles introduces the need for more variables in the semantic representation than in the semantic representations of the same verbs in [49], hence the complexity results of Pratt and Third do not hold for BOXER semantic representations over their natural language restrictions.

The circumvention of BOXER altogether and the usage of the Pratt and Third approach to derive semantic representations is unjustified. Pratt and Third do not possess an implementation for a parser that would build semantic representations according to the grammar they use. Their results are primarily theoretical.

To obtain semantic representations which are decidable, we will try to control, not the number of variables in the first-order formulas, but the presence and number of the universal quantifier and the negation in the formulas. This approach derives from observations made about the semantic representations as well as from the principles according to which the representations are compositionally constructed.

We have observed that the semantic representations for natural language which we derive (using BOXER) are not general first-order logic formulas, but rather belong to a fragment of first-order logic. We have also observed that this fragment closely resembles certain decidable fragments of first order logic, but it is undecidable itself. By taking advantage of the compositional mechanism in which BOXER builds the semantic representations we have found a way to reduce the semantic representations of the natural language to decidable fragments of first-order logic by imposing as minimal as possible restrictions to the lexicon of the natural language. We constrict the presence and/or the number of certain words in a sentence. The rules which define our restrictions are minimal and simple enough to allow for building answers and posing questions in a natural way.

The goal of this chapter is to present the restrictions of natural language we impose. The structure of this chapter is as follows. First we will present the decidable fragments of fist-order logic to which we want to restrict our semantic representations. Then we will define and single out the words whose usage we intend to restrict. These are the words which introduce the logic operators implication, disjunction and negation in the semantic representations. In the remaining of the chapter we will define the lexicons allowed to be used in a natural language sentence (the constituents of the answers retrieved by the statistic answer retrieval system) and a lexicon lexicon of allowed natural

language questions.

For each of the "answers" lexicons we are going to describe the first-order logic formulas (as obtained by BOXER) for a sentence built according to their rules. We will show that these formulas are decidable for satisfiability. By using an analysis of the FAQ sheet of our library domain we will show the percentage of coverage of the described lexicons.

For the "questions" lexicon we will also describe the first-order logic formulas (as obtained by BOXER) for questions built according to its rules. We will show that the entailment relation between answers and questions of the defined lexicons is decidable. We are going to use an analysis of question corporas (presented in [10]) to show the coverage capabilities of our lexicon.

4.1 Decidable Fragments of First-Order Logic

By observing the mechanism and the results of the process of building semantic representations by BOXER, we have established that the first-order formulas obtained resemble two well known fragments of first-order logic. We are going to restrict the natural language in such a way for the semantic representations to belong to one of these decidable fragments.

Before proceeding with the motivations for the natural restrictions and the definitions of the restrictions we will briefly introduce these decidable fragments of first-order logic. We will define both of the fragments and then state those properties (of the formulas which belong to them) which we will use later.

The Bernays-Schönfinkel-Ramsey (BSR) class

The Bernays-Schönfinkel-Ramsey class of decidable formulas of first-order logic is one of the first discovered decidable fragments(1929). It was named after the logicians Paul Bernays and Moses Schönfinkel. In some of the literature (as in for example [13]) the name of the logician Frank Plumpton Ramsey is added to the name of the class, because he proved that this class of formulas has the finite model property if every satisfiable formula in the class has a finite model (Ramsey theorem). For a detailed analysis of this decidable class we refer the reader to [13].

Definition 4.1.1. The Bernays-Schönfinkel-Ramsey (BSR) class of formulas is the class of first-order logic formulas which do not contain function symbols in their signature and that, when written in prenex normal form, have an $\exists^*\forall^*$ quantifier prefix.

The formulas of the Bernays-Schönfinkel-Ramsey class are decidable for satisfiability in $\sum_2^p$ time. The proof for the complexity of satisfiability for this can be found in various literature. See for example page 293 of [13].

The Guarded Fragment (GF)and the Loosely Guarded Fragment (LGF)

The guarded fragments (without equality) were introduced by Andréka, Németi and van Benthem [2] in 1998. The complexity results for satisfiability of the Guarded and the Loosely Guarded Fragment (with equality) were given by Erich Grädel [31]. The proof that the Loosely Guarded fragment has the finite model

property is presented in [36]. We state here the definitions of the Guarded Fragment and the Loosely Guarded Fragment as they are given in [31].

Definition 4.1.2. The Guarded Fragment GF of first-order logic without functions symbols is defined by induction as follows:

1. Every atomic formula belongs to the GF.
2. GF is closed under the propositional connectives $\neg$, $\wedge$, $\vee$, $\Rightarrow$ and $\Leftrightarrow$.
3. If $\mathbf{x}$,$\mathbf{y}$ are tuples of variables, $\alpha(\mathbf{x},\mathbf{y})$ is atomic and $\psi(\mathbf{x},\mathbf{y})$ is a formula in GF such that $free(\psi) \subseteq free(\alpha)$, then the formulas

$$\exists \mathbf{y}(\alpha(\mathbf{x},\mathbf{y})) \wedge (\psi(\mathbf{x},\mathbf{y}))$$

$$\forall \mathbf{y}(\alpha(\mathbf{x},\mathbf{y})) \Rightarrow (\psi(\mathbf{x},\mathbf{y}))$$

belong to the GF.

Here, $free(\psi)$ means the set of free variables of ψ.

Definition 4.1.3. The Loosely Guarded Fragment (LGF) is defined similarly to GF in 4.1.2, but the quantifier-rule is relaxed as follows:

1. Every atomic formula belongs to the LGF.
2. LGF is closed under propositional connectives $\neg$, $\wedge$, $\vee$, $\Rightarrow$ and $\Leftrightarrow$.
3. If $\psi(\mathbf{x},\mathbf{y})$ is in LGF, and $\alpha(\mathbf{x},\mathbf{y}) = \alpha_1 \wedge \cdots \wedge \alpha_m$ is a conjunction of atoms, then

$$\exists \mathbf{y}(\alpha_1 \wedge \cdots \wedge \alpha_m) \wedge (\psi(\mathbf{x},\mathbf{y}))$$

$$\forall \mathbf{y}(\alpha_1 \wedge \cdots \wedge \alpha_m) \Rightarrow (\psi(\mathbf{x},\mathbf{y}))$$

belong to LGF, provided that $free(\psi) \subseteq free(\alpha) = \{\mathbf{x},\mathbf{y}\}$ and for every quantified variable y_i and every other variable $z \in \{\mathbf{x},\mathbf{y}\}$ there is at least one atom α_j that contains both y_i and z.

We are going to use the results on the complexity for satisfiability of the GF and the LGF, when the formulas have a bound on the arity of the predicate symbols. The formulas of the Guarded and the Loosely Guarded Fragment are decidable for satisfiability in EXP time if their predicate symbols have a bound arity. The proof for the complexity of satisfiability for the GF and the LGF is presented in [31].

4.2 Logic Operator Introducing Words

BOXER builds the semantic representations compositionally starting from the leaves of the CCG derivation tree. The leaves of the tree contain lexical items (ordered pairs consisting of the CCG category and a lemmatised wordform). Each of these lexical items is assigned a λ-expression. The semantic representation for the sentence is obtained by applying the combinatory rules of CCG (reformulated in terms of functional application) over each of these λ-expression and applying β-conversion to the result. The function application rules and the

β-conversion rule can not introduce logic operators ($\exists, \forall, \neg, \wedge, \vee, \rightarrow$). Consequently all the logic operators in the final first-order representation will be propagated from the λ-expressions of the lexical items.

We have already noted that in the implementation of BOXER semantic representation representation for the closed-class lexical items is spelled out for each item individually (note that in BOXER an item can also be a phrase as for example "*instead of*" and not only a lemma). By studying the implementation of BOXER we have observed that all the lexical items which introduce the logic symbols $\forall, \neg, \vee$ and $\rightarrow$ are members of the closed-class of lexical items. Consequently, the logic symbols $\forall, \neg, \vee$ and $\rightarrow$ will be present in the semantic representation (the first-order formula) of a sentence if and only there is a lexical item present in the CCG derivation of the sentence (a word or a phrase in the NL sentence) which is a member of the closed-class lexical items who has (one or more of) these logic symbols in its assigned λ-expression. In addition we have observed that the logic operator (quantifier) $\forall$ only appears together with the logic operator $\rightarrow$.

We will exploit these observations to determine natural language restrictions which will ensure that the semantic representations over such restricted natural language sentences belong, or can be reduced, to one of the decidable fragments of first-order logic we already defined.

In order to define the natural language restrictions we intend to pose, we will first present the closed-class lexical items which introduce the logic operators $\neg, \vee$ and $\rightarrow$. These are the words and phrases whose presence and frequency *in a sentence* we will restrict.

Negation and Implication Introduces

The words which introduce the logic operators negation and implication are considered together because it can happen that both logic operators are being introduced by one lexical item. There are a few groups (according to their assigned λ-expressions) of words which introduce negation and/or implication:

1. *Simple Negation Introducer*

   Simple negation introducer is one of the lexical items: another, instead of, neither, nobody, none, noone, no-one, not, nothing, other, previous.

   One of the simple negation introducer words introduces exactly one negation in the semantic representation of the sentence in which it appears. The reason for this is that the λ-expression assigned to each of the simple negation introducer is:

   $$\lambda V \lambda Q \lambda F. \neg((V@Q)@F) \tag{4.1}$$

   The expression $(V@Q)@F$ is said to be in the scope of the negation. The scope of the negation indicates which expressions will be negated. In natural language, when the scope of negation intersects with he scope of another logical operator, there can be more then one semantic representation which applies. This is a case of scope semantic ambiguities which we mentioned earlier. BOXER does not handle scope ambiguities and it produces only one semantic representation.

For example, in the sentence *Mary did not see or meet the student* the scope of the negation intersects with the scope of the disjunction. There are two possible interpretations for this sentence. According to the first, the phrase "see or meet" is in the scope of the negation. According to the second, only the verb "see" is negated. BOXER will build the second interpretation.

In the remaining of this chapter, to denote the presence of *one Simple Negation Introducer per sentence*, we will add the letter $\boldsymbol{N}$ to the name of the lexicon which allows simple negation introducing words.

2. *Implication Introducer*

   Implication introducer is one of the lexical items: all, any, anybody, anyone, anything, anywhere, each, either, every, everybody, everyone, everything, everywhere, few, if. Implication introducer are also those lexical items (noun s) which have been assigned the category (NP\NP)/N[num], namely the phrases which express that a number property holds for each member of a group. For for example in the phrase "seven cents a share", the noun "cents" will be assigned the category (NP\NP)/N[num].

   One of the implication introducer words introduce exactly one logic operator implication in the semantic representation of the sentence in which it appears. The reason for this is that the λ-expression assigned to each of the implication introducer word is:

   $$\lambda P \lambda Q. \forall x (P@x \rightarrow Q@x) \tag{4.2}$$

   The expression $(P@x \rightarrow Q@x)$ is said to fall under the scope of the universal quantifier, and the variably x is said to be universally bound.

   As it was the case with negation, here also the universal quantifier can give rise to scope ambiguity if it intersects with another quantifier or similar expression. This example was considered at the end of Section 3.2.

   In the remaining of this chapter, to denote the presence of *one Implication Introducer in a sentence*, we will add the letter $\boldsymbol{I}$ to the name of the lexicon which allows simple negation introducing words.

3. *Special Negation Introducer*

   Special negation introducer is one of the lexical items: no, neither, nowhere.

   One of the special negation introducer words introduces exactly one negation and one implication in the semantic representation of the sentence in which it appears. The reason for this is that the λ-expression assigned to each of the members of the special negation introducer group is:

   $$\lambda P \lambda V \lambda Q \lambda F. \forall x (P@x \rightarrow (\neg((V@Q)@F)@x) \tag{4.3}$$

   In the case of special negation introducer lexical items, the scope of the negation falls under the scope of the universal quantification.

   In the remaining of this chapter, to denote the presence of *one Special Negation Introducer in a sentence*, we will add the letter N^S to the name of the lexicon which allows special negation introducing lexical item.

4. *Superlatives*

   Superlatives are adjectives or adverbs which indicate that something has a feature to a greater degree than anything it is being compared to in a given context.

   Each superlative introduces both implication and negation in the semantic representation of the sentence it appears in. Because of the complexity of the first-order formulas obtained from natural language sentences which contain superlatives as well as the fact that no superlative appear in the FAQ sheet for the library domain which we use for answer retrieval in our IQA system, we will exempt the superlatives from all the lexicons we will define and work with in this thesis.

Disjunction introducer

Lastly we are going to look into the introduction of the logic operator disjunction in the semantic representations of natural language sentences.

Disjunction is a form of coordination in the sentence. Disjunction introducer we will call the the lexical items: *or*, all the range constructions (e.g., "10 to 20").

The λ-expression assigned to each of the members of the disjunction introducer group is:

$$\lambda P \lambda Q.P \vee Q \tag{4.4}$$

P and Q are in the scope of the disjunction. Disjunctions can also give rise to scope ambiguities. Considered the example we gave in the case of simple negation introduces.

In the remaining of this chapter, to denote the presence of arbitrarily many Disjunction Introducer items in a sentence we will add the letter $\boldsymbol{D}$ to the name of the lexicon which allows these items.

The logical operators groups of lexical items we have introduced will serve us as a base to define the restrictions of the natural language which will guarantee us decidable semantic interpretations. We will establish the restrictions by defining lexicons of allowed and forbidden lexical items (words/phrases) over which we will build the comprising sentences of the possible answer or the question. Because the semantic representation of a question differs from the semantic representation of a declarative sentence, we are going to define lexicons separately for the possible answers and the questions. We will begin by defining the lexicons for the possible answers. It is enough to consider the Lexicons for individual sentences, because a text of more then one sentence is semantically represented as a conjunction of the semantic representations of the individual sentences.

4.3 Lexicon Restrictions for Declarative Sentences

We will now look into lexicon restrictions for controlling the natural language of the possible answers which can be retrieved by the IQA system over the library domain.

We consider a possible answer to be a short text (up to five sentences) of *declarative* sentences in English. Declarative we will call every sentence which is not a question.

In this section we will start by defining a basic lexicon which contains none of the words whose semantic representations introduce the logical operators negation, implication, disjunction and consequently the universal quantification. We will show the properties of the representations of sentences built over this basic lexicon and we will then extend this lexicon to include a controlled number of words/phrases previously defined as Negation Introducer, Implication Introducer and Special Negation Introducer, as well as Disjunction Introducer.

For each of the defined lexicons we will show the general schema of first-order representations and their equv-satisfiable formulas obtained by Skolemization [1]for a sentence built over the lexicon in question. We will show the decidable first-order logic fragment to which the schema belongs to together with the complexity results for deciding satisfiability for that fragment, as well as the. For each lexicon, once we define all of them, we will also show how many of the sentences from the answers of our FAQ sheet can be built using only words allowed by the lexicon in question.

4.3.1 The EC Lexicon

The first lexicon we will define is the most basic one which will be contained in all the subsequent lexicons defined in this chapter. This basic lexicon contains all the words of natural language except the words (and phrases) previously defined as Simple Negation Introducer, Implication Introducer, Special Negation Introducer, Disjunction Introducer and Superlatives. (Recall that BOXER does not represent plurals and tense.) The semantic representations of the sentences build over this lexicon are built out of the following language: constants, variables, unary and binary predicate symbols, *existential quantifiers* and *conjunctions*. Because the only logic operators that appear in these representations are the **E**xistential quantifier and the **C**onjunction we will name this lexicon the EC Lexicon.

Definition 4.3.1. EC Lexicon is the lexicon which contains all the words of natural language except the words (and phrases) previously defined as Simple Negation Introducer, Implication Introducer, Special Negation Introducer, Disjunction Introducer and Superlatives.

We will now derive the general schema which describes all the first-order representations of sentences built from the EC lexicon.

The semantic representations built by BOXER consist of predicate symbols which can only be unary or binary. Also recall that BOXER produced first-order formulas do not contain constant symbols. Because we have removed all the possible sources of universal quantification, implication, disjunction and negation the semantic representation of all the words from the EC Lexicon will be built from the following language:

> unary predicates, binary predicates, variables, existential quantifiers and conjunctions

[1]recall that we in Section 3.3 we stated that instead of with the direct representation of possible answers A we will work with the eqiv-satisfiable formula A^S

Consequently, all the first-order representations of sentences built from the EC Lexicon will fit the schema presented in 4.5. With P we will denote the unary predicates and with Q the binary predicates. The formula 4.5 will not contain any free variables.

$$\exists x_1 \cdots \exists x_n \bigwedge_1^m P(x_i) \wedge \bigwedge_1^s Q(x_j, x_k) \quad i, j, k \in \{1, \ldots n\} \tag{4.5}$$

Because the formula 4.5 contains only the logic symbols $\{\exists, \wedge\}$ which is not a complete set of boolean operators, the formulas 4.5 can be reduced to propositional logic and as such will always be satisfiable. This is stated in the corollary 4.3.1.

Corollary 4.3.1. *The first-order representations of sentences of the EC Lexicon built by* BOXER *are always satisfiable.*

If the formula 4.5 is Skolemized as described in Section 3.3, we will obtain the Formula 4.6. Note that 4.6 will have exactly one Herbrand Model.

$$\bigwedge_1^m P(a_i) \wedge \bigwedge_1^s Q(a_j, a_k) \quad i, j, k \in \{1, \ldots n\} \tag{4.6}$$

Consider the following example (Example 4.3.1) of a sentence built using only words of the EC Lexicon and its corresponding first-order representation.

Example 4.3.1. *The current opening hours of the University Library can be found at the website.*

The first-order representation:

$$\begin{aligned}\exists x_0\ \exists x_1\ \exists\ x_2 \exists\ x_3\ \exists x4\ (&\texttt{university\_library}(x_0) \wedge \texttt{opening}(x_1) \wedge \texttt{current}(x_2) \\ &\wedge \texttt{hour}(x_2)\ \wedge\ \texttt{nn}(x_1, x_2)\ \wedge\ \texttt{of}(x_2, x_0)\ \wedge\ \texttt{website}(x_3) \\ &\wedge \texttt{find}(x_4)\ \wedge\ \texttt{patient}(x_4, x_2)\ \wedge\ \texttt{event}(x_4)\ \wedge\ \texttt{at}(x_4, x_3))\end{aligned}$$

The Skolemized equ-satisfiable formula:

$$\begin{aligned}&\texttt{university\_library}(a_0)\ \wedge\ \texttt{opening}(a_1)\ \wedge\ \texttt{current}(a_2) \\ &\wedge \texttt{hour}(a_2)\ \wedge\ \texttt{nn}(a_1, a_2)\ \wedge\ \texttt{of}(a_2, a_0)\ \wedge\ \texttt{website}(a_3) \\ &\wedge \texttt{find}(a_4)\ \wedge\ \texttt{patient}(a_4, a_2)\ \wedge\ \texttt{event}(a_4)\ \wedge\ \texttt{at}(a_4, a_3)\end{aligned}$$

We proceed by extending the EC Lexicon with Disjunction Introducer's and observe how this extension influences the first-order representations of natural language sentences built by BOXER.

4.3.2 The ECD Lexicon

Extending the EC lexicon with disjunction.

Apart from the words allowed in the EC lexicon we are going to additionally allow for arbitrarily many lexical items (words/phrases) which we previously described as Disjunction Introducer's. Consequently, apart from the constants, variables, unary and binary predicate symbols, existential quantifiers and conjunctions, the semantic representations over the new lexicon will also contain *disjunctions*. To denote that the new lexicon is obtained by extending the EC Lexicon with **D**isjunction we will name it the ECD Lexicon.

Definition 4.3.2. ECD Lexicon is the lexicon which contains all the words of natural language which are contained in the EC Lexicon and the lexical items previously defined as Disjunction Introducer.

Compared to the EC Lexicon, which had no sources that introduce universal quantification, implication, disjunction and negation in the semantic representation, here in the ECD Lexicon we have arbitrarily many sources of disjunction (recall that we consider the lexicon of a single sentence). All the first-order representations of sentences built from the ECD Lexicon fit, or can be reduced[2] to, the schema presented in 4.7 . Again, with P we will denote the unary predicates and with Q the binary predicates. The formula 4.7 will not contain any free variables.

$$\exists x_1 \cdots \exists x_n \bigwedge_1^m (\bigvee_1^r P(x_i) \vee \bigvee_1^s Q(x_j, x_k)) \quad i,j,k \in \{1, \dots n\} \tag{4.7}$$

The formula 4.7 contains only the logic symbols $\{\exists, \wedge, \vee\}$ which is still not a complete set of boolean operators. Consequently the formulas 4.7, as the formulas 4.5, can be reduced to propositional logic and as such they will be always satisfiable. This is stated in the Corollary 4.3.2.

Corollary 4.3.2. *The first-order representations of sentences of the ECD Lexicon built by* BOXER *are always satisfiable.*

If the formula 4.7 is Skolemized as described in Section 3.3, we will obtain the Formula 4.8.

$$\bigvee_1^r P(a_i) \vee \bigvee_1^s Q(a_j, a_k) \quad i,j,k \in \{1, \dots n\} \tag{4.8}$$

We will consider an example, a sentence built only with words from the ECD Lexicon.

Example 4.3.2. *A user can access the services of the University Library with a Student Card or a Campus Card.*

First-Order representation:

[2]The reduction in question is the standard reduction of an arbitrary logic formula to a Conjunctive Normal Form (CNF)

$$\exists x_0\ \exists x_1\ \exists x_2\ \exists x_3\ \exists x_4\ \exists x_5\ (\texttt{university\_library}(x_0) \wedge \texttt{service}(x_1) \wedge \texttt{of}(x_1, x_0)$$
$$\wedge\ \texttt{user}(x_2)\ \wedge\ \texttt{access}(x_3)\ \wedge\ \texttt{agent}(x_3, x_2) \wedge\ \texttt{patient}(x_3, x_1)$$
$$\wedge\ ((\texttt{event}(x_3)\ \wedge\ \texttt{campus\_card}(x_4)\ \wedge\ \texttt{with}(x_3, x_4))$$
$$\vee(\texttt{event}(x_3)\ \wedge\ \texttt{student\_card}(x_5)\ \wedge\ \texttt{with}(x_3, x_5))))$$

The Skolemized equ-satisfiable formula:

$$\texttt{university\_library}(a_0) \wedge \texttt{service}(a_1) \wedge \texttt{of}(a_1, a_0) \wedge \texttt{user}(a_2)$$
$$\wedge\texttt{access}(a_3) \wedge \texttt{agent}(a_3, a_2) \wedge \texttt{patient}(a_3, a_1) \wedge ((\texttt{event}(a_3)$$
$$\wedge\texttt{campus\_card}(a_4)\wedge\texttt{with}(a_3, a_4))\vee(\texttt{event}(a_3)\wedge\texttt{student\_card}(a_5)\wedge\texttt{with}(a_3, a_5)))$$

Next we will extend the EC Lexicon with one Simple Negation Introducer or one Special Negation Introducer and observe how this extension influences the first-order representations of natural language sentences built by BOXER.

4.3.3 The ECN Lexicon and ECN^S Lexicon

Extending the EC lexicon with negation.

We will define extensions of the EC Lexicon by negation introducing lexical items. We will start from the EC Lexicon and first extend it with *only one* member (lexical item) of the Simple Negation Introducer group showing the general schema and its properties for the representations of sentences constructed over this lexicon. Next we will repeat the same for the Special Negation Introducer.

The ECN Lexicon.

We will extend the *EC* lexicon with *only one* member of choice (per sentence) from the the Simple Negation Introducer group. This means that we will allow for sentences which contain one Simple Negation Introducer, as it is the case in the sentence *John did **not** eat the cupcakes*, but we will forbid more then one Simple Negation Introducer in a sentence, as it is the case in the sentence *John did **not** eat cupcakes **instead of** the muffins*.

Apart from the constants, variables, unary and binary predicate symbols, existential quantifiers and conjunctions, the semantic representations over this new lexicon will also contain *negations*. To denote that the new lexicon is obtained by extending the EC Lexicon with a negation derived from the semantic representation of a Simple **N**egation Introducer we will name it the ECN Lexicon.

Definition 4.3.3. ECN Lexicon is the lexicon which contains all the words of natural language which are contained in the EC Lexicon *and one lexical item previously defined as Simple Negation Introducer.*

The presence of not more then one Simple Negation Introducer in the syntax of the natural language sentence guarantees that there will be only one negation in the semantic representation of that sentence. All the first-order representations of sentences built from the ECN Lexicon will fit the schema presented in

4.9. Again with P we will denote the unary predicates and with Q the binary predicates and the formula 4.9 will not contain any free variables.

$$\exists \mathbf{x} \exists \mathbf{y} \bigwedge_{1}^{m} P(\mathbf{x}) \wedge \bigwedge_{1}^{s} Q(\mathbf{x}, \mathbf{y}) \wedge \neg (\exists \mathbf{z} \bigwedge_{1}^{m'} P'(\mathbf{y}) \wedge \bigwedge_{1}^{s'} Q(\mathbf{x}, \mathbf{y}, \mathbf{z})) \tag{4.9}$$

If we unfold the conjunctions in 4.9, write the formula in prenex normal form (push all the quantifiers in front of the formula) and push the negation to appear only in front of atoms, the schema 4.9 will become the schema 4.10, where P_i will denote a unary or a binary positive atom. (For simplicity we will omit writing the variables within the predicates.) It is important to emphasize that the scope of the universal quantifiers in 4.10 *does not extend over any existential quantifiers.*

$$\exists \mathbf{x} \forall \mathbf{y} (P_1 \wedge \cdots \wedge P_n \wedge (\neg P_{n+1} \vee \cdots \vee \neg P_{n'})) \tag{4.10}$$

It is evident that the formula 4.10 belongs to the Bernays-Schönfinkel-Ramsey class. Consequently, the representations of sentences from the ECN Lexicon will inherit all the properties of the Bernays-Schönfinkel-Ramsey class including the decidability for satisfiability of this class.

Corollary 4.3.3. *For the first-order representations of sentences of the ECN Lexicon built by* BOXER*, satisfiability is decidable in* $\sum_2^p$ *time over the size of the formula.*

Note that a conjunction of closed formulas which belong to the BSR class will also belong to the BSR class.

The Formula 4.11 represents the Formula 4.9 after Skolemization.

$$\bigwedge_{1}^{m} P(\mathbf{a}) \wedge \bigwedge_{1}^{s} Q(\mathbf{a}, \mathbf{b}) \wedge (\forall \mathbf{z} \neg (\bigwedge_{1}^{m'} P'(\mathbf{b}) \wedge \bigwedge_{1}^{s'} Q(\mathbf{a}, \mathbf{b}, \mathbf{z})) \tag{4.11}$$

Note that Formula 4.11 will not contain function symbols because the existential quantifiers never are in the scope of the universal quantifiers. All the variables of 4.11 will be universally quantified.

We will consider an example, a sentence built only with words from the ECN Lexicon.

Example 4.3.3. *The access to databases and electronic journals is not restricted to computers within the library.*

First-Order representation:

$$\exists x_0\ \exists x_1\ \exists x_2\ \exists x_3\ (\texttt{access}(x_0)\ \wedge\ \texttt{database}(x_1)\ \wedge\ \texttt{to}(x_0, x_1) \wedge\ \texttt{electronic}(x_2)$$
$$\wedge\ \texttt{journal}(x_2)\ \wedge\ \texttt{to}(x_0, x_2)\ \wedge\ \texttt{library}(x_3) \wedge\ \neg(\exists x_4\ \exists x_5\ (\texttt{restrict}(x_4)$$
$$\wedge\ \texttt{computer}(x_5)\ \wedge\ \texttt{to}(x_4, x_5)\ \wedge\ \texttt{event}(x_4)\ \wedge\ \texttt{within}(x_4, x_3))$$

The Skolemized equ-satisfiable formula:

$$\texttt{access}(a_0) \wedge \texttt{database}(a_1) \wedge \texttt{to}(a_0, a_1) \wedge \texttt{electronic}(a_2)$$
$$\wedge \texttt{journal}(a_2) \wedge \texttt{to}(a_0, a_2) \wedge \texttt{library}(a_3) \wedge (\forall x_4 \, \forall x_5 \, \neg(\texttt{restrict}(x_4)$$
$$\wedge \texttt{computer}(x_5) \wedge \texttt{to}(x_4, x_5) \wedge \texttt{event}(x_4) \wedge \texttt{within}(x_4, a_3)))$$

Next we will extend the EC Lexicon with one Special Negation Introducer instead of extending it with the one Special Negation Introducer which we just observed and show that in the case of extension with a Special Negation Introducer the first-order representations still belong to the Bernays-Schönfinkel-Ramsey class of formulas.

The ECN^S Lexicon.

We will extend the EC Lexicon with *only one* lexical item (per sentence) from the Special Negation Introducer group. As it was the case with the ECN Lexicon, here also we will allow for sentences which contain only one Special Negation Introducer.

For example, the sentence *John ate* ***no*** *cupcakes* will be allowed but the sentence ***Nowhere no*** *cupcakes can be found* will not be allowed.

Apart from the constants, variables, unary and binary predicate symbols, existential quantifiers and conjunctions, the semantic representations over the new lexicon will also contain *negations*. To denote that the new lexicon is obtained by extending the EC Lexicon with a negation derived from the semantic representation of a **S**pecial **N**egation Introducer we will name it the ECN^S Lexicon.

Definition 4.3.4. ECN^S **Lexicon** is the lexicon which contains all the words of natural language which are contained in the EC Lexicon *and one word (or phrase) previously defined as Special Negation Introducer.*

In the most general case, all the first-order representations of sentences built from the ECN^S Lexicon will be a conjunction of arbitrarily many formulas of shape as presented in 4.5 to which *only one* formula of shape $\forall \mathbf{y} \, (\varphi(\mathbf{x}, \mathbf{y}) \rightarrow \neg\exists \mathbf{z} \, (\phi(\mathbf{x}, \mathbf{y}, \mathbf{z})))$ is conjuncted (which derives from the semantic representation of the unique special negation introducer).

More precisely, all the first-order representations of sentences built from the ECN^S Lexicon will fit the schema presented in 4.12. Here also with P we will denote the unary predicates and with Q the binary predicates and the formula 4.9 will not contain any free variables.

$$\exists \mathbf{x}(\bigwedge_1^m P(\mathbf{x}) \wedge \bigwedge_1^s Q(\mathbf{x}) \wedge \forall \mathbf{y}(\bigwedge_1^m P(\mathbf{y}) \wedge \bigwedge_1^s Q(\mathbf{x}, \mathbf{y}) \rightarrow \neg(\exists \mathbf{z} \bigwedge_1^{m''} P(\mathbf{z}) \wedge \bigwedge_1^{s''} Q(\mathbf{x}, \mathbf{y}, \mathbf{z})))) \tag{4.12}$$

When we unfold the conjunctions in 4.12 , write the formula in prenex normal form and push the negation to appear only in front of atoms, the schema 4.12 will become the schema 4.13. The P_i again denote unary and binary positive atoms. The scope of the universal quantifier in 4.13 does not extend over any existential quantifiers.

$$\exists \mathbf{x}\ \forall \mathbf{y}\ \forall \mathbf{z}(P_1 \wedge \cdots \wedge P_n \wedge ((P_{n+1} \wedge \cdots \wedge \neg P_{n'}) \rightarrow (\neg P_{m'+1} \vee \cdots \vee \neg P_{n''})) \quad (4.13)$$

It is now clear that the formula 4.13 will belong to the Bernays-Schönfinkel-Ramsey class.

As it was the case with the representations of sentences from the ECN Lexicon, the representations of sentences from the ECN^S Lexicon will also be decidable for satisfiability.

Corollary 4.3.4. *For the first-order representations of sentences of the ECN^S Lexicon built by* BOXER*, satisfiability is decidable in $\sum_2^p$ time over the size of the formula.*

The Formula 4.14 represents the Formula 4.12 after Skolemization.

$$\bigwedge_1^m P(\mathbf{a}) \wedge \bigwedge_1^s Q(\mathbf{a}) \wedge \forall \mathbf{y}(\neg(\bigwedge_1^m P(\mathbf{y}) \wedge \bigwedge_1^s Q(\mathbf{a},\mathbf{y}) \rightarrow \forall \mathbf{z} \bigwedge_1^{m''} P(\mathbf{z}) \wedge \bigwedge_1^{s''} Q(\mathbf{a},\mathbf{y},\mathbf{z}))) \quad (4.14)$$

The Formula 4.14 will not contain function symbols because the existential quantifiers, again, are never in the scope of the universal quantifiers. All the variables of 4.14 will be universally quantified.

We will consider an example, a sentence built only with words from the ECN^S Lexicon.

Example 4.3.4. *John ate no cupcakes.*

First-Order representation:

$$\exists x_0\ (\texttt{john}(x_0)\ \wedge\ \forall x_1\ (\texttt{cucpake}(x_1)\ \rightarrow\ \neg\exists x_2\ (\texttt{eat}(x_2)$$
$$\wedge\ \texttt{event}(x_2)\ \wedge\ \texttt{agent}(x_2, x_0)\ \wedge\ \texttt{patient}(x_2, x_0))))$$

The Skolemized equ-satisfiable formula:

$$\texttt{john}(a_0)\ \wedge\ \forall x_1\ \forall x_2(\texttt{cucpake}(x_1) \rightarrow$$
$$\neg(\texttt{eat}(x_2) \wedge \texttt{event}(x_2) \wedge \texttt{agent}(x_2, a_0) \wedge \texttt{patient}(x_2, x_0)))$$

Next we will now go back to he EC Lexicon we started from and extend it with one Implication Introducer.

4.3.4 The ECI Lexicon

We are now interested in extending the EC lexicon with an Implication Introducer. Similarly as we did with the Simple Negation Introducer and the Special Negation Introducer, the new lexicon will allow for sentences which contain only one Implication Introducer.

For example, the sentence Everyone *eats cupcakes* can be constructed from the EC Lexicon extended with one Implication Introducer, but the sentence ***Everyone*** *eats cupcakes* ***all*** *the time.* can not be constructed.

Now, apart from the constants, variables, unary and binary predicate symbols, existential quantifiers and conjunctions, the semantic representations over the new lexicon will contain *implication and universal quantification* but no negation. To denote that the new lexicon is obtained by extending the EC Lexicon with an implication derived from the semantic representation of an **I**mplication Introducer we will name it the ECI Lexicon.

Definition 4.3.5. ECI Lexicon is the lexicon which contains all the words of natural language which are contained in the EC Lexicon *and one word (or phrase) previously defined as Implication Introducer.*

As a result of the semantic interpretation of the Implication Introducer one sub-formula of the form $\forall \mathbf{y}\ (\varphi(\mathbf{x},\mathbf{y}) \rightarrow \exists \mathbf{z}\ (\phi(\mathbf{x},\mathbf{y},\mathbf{z})))$ will appear in the first-order interpretation of the sentences built from the ECI Lexicon. This sub-formula will be the representation of the natural language phrase which falls in the scope of the universal quantifier. The remaining of the representation corresponding to the remaining of the sentence can only consist of existentially quantified conjunctions of positive atoms because there are no sources of disjunctions and negations in the ECI Lexicon and no other sources of universal quantifiers and implications.

The first-order representations of sentences built from the ECI Lexicon will fit the schema presented in 4.15. P denotes the unary predicates and Q denotes the binary predicates. The entire formula contains no free variables.

$$\exists \mathbf{x}(\bigwedge_{0}^{m} P(\mathbf{x}) \wedge \bigwedge_{0}^{s} Q(\mathbf{x}) \wedge \forall \mathbf{y}(\bigwedge_{0}^{m'} P(\mathbf{y}) \wedge \bigwedge_{0}^{s'} Q(\mathbf{x},\mathbf{y}) \rightarrow (\exists \mathbf{z} \bigwedge_{0}^{m''} P(\mathbf{z}) \wedge \bigwedge_{0}^{s''} Q(\mathbf{x},\mathbf{y},\mathbf{z})))) \tag{4.15}$$

To make the formula 4.15 easier to read we will represent with ϕ, φ and η the quantifier free conjunctions of positive unary and binary atoms. The result is presented in the formula 4.16. The formula 4.16 is closed.

$$\exists \mathbf{x}\ (\phi(\mathbf{x})\ \wedge\ \forall\ \mathbf{y}\ (\varphi(\mathbf{x}_i,\mathbf{y}) \rightarrow \exists\ \mathbf{z}(\eta(\mathbf{x}_j,\mathbf{y},\mathbf{z})))),\ \mathbf{x}_i \subseteq \mathbf{x},\ \mathbf{x}_j \subseteq \mathbf{x} \tag{4.16}$$

The formula 4.16 resembles the formulas from the Loosely Guarded Fragment. The condition that will not be satisfied by 4.16 is the condition $free(\psi) \subseteq free(\alpha) = \{\mathbf{x},\mathbf{y}\}$ because the variables $\mathbf{x}_i$ in φ are not necessarily the same as, or a subset of, the variables $\mathbf{x}_j$ in η.

We will, however, show that 4.16 is decidable for satisfiability in the same complexity class as the formulas of the LGF because it can be transformed (in polynomial time over its size) to an equi-satisfiable formula which belongs to the LGF.

Theorem 4.3.1. *The first-order representations of sentences of the ECI Lexicon built by* BOXER *are decidable for satisfiability in* EXPTIME *time over the size of the formula.*

Proof. Let us name *free existential quantifiers* those existential quantifiers in a formula which do not fall under the scope of any of the universal quantifiers of the formula.

We will transform the formula 4.16 in an equi-satisfiable formula 4.17. In 4.17, **a** denote constant symbols.

$$\phi(\mathbf{a}) \ \wedge \ \forall \ \mathbf{y} \ (\varphi(\mathbf{a}, \mathbf{y}) \rightarrow \exists \ \mathbf{z}(\eta(\mathbf{a}, \mathbf{y}, \mathbf{z}))) \tag{4.17}$$

The formula 4.17 is obtained from 4.16 by replacing all the variables which are bound by free existential quantifiers with fresh constant symbols ($\mathbf{a}$) that do not appear in the signature of 4.16. This transformation is called *Skolemization* and it can be done in polynomial time over the size of 4.16. Proof of the sat-preservation property of this Skolemization can be found for example as proof of Lemma 2.2.2 in [43].

The formula 4.17 satisfies the conditions for membership in the LGF: $free(\eta) \subseteq free(\varphi) = \{\mathbf{y}\}$ and for every quantified variable y_i there will be at least one atom φ_j that contains y_i. The latter is a direct consequence of the semantic interpretation of the Implication Introducer.

As a member of the LGF, 4.17 is decidable for satisfiability in double exponential time over its size. Because all the atoms in 4.17 will have arity of with an upper bound two (because all the predicate symbols derived froms BOXER are at most binary in arity) the complexity of decidability will in fact be EXPTIME, a result which was proved in Gr99. □

The Formula 4.18 represents the Formula 4.15 after Skolemization.

$$\bigwedge_{0}^{m} P(\mathbf{a}) \wedge \bigwedge_{0}^{s} Q(\mathbf{a}) \wedge \forall \mathbf{y} (\bigwedge_{0}^{m'} P(\mathbf{y}) \wedge \bigwedge_{0}^{s'} Q(\mathbf{a}, \mathbf{y}) \rightarrow (\bigwedge_{0}^{m''} P(f(\mathbf{y})) \wedge \bigwedge_{0}^{s''} Q(\mathbf{a}, \mathbf{y}, f(\mathbf{y})))) \tag{4.18}$$

The Formula 4.14 will contain function symbols. All the variables of 4.14 will be universally quantified.

We will consider an example, a sentence built only with words from the ECI Lexicon.

Example 4.3.5. *All users who possess a Student Card may use the Interlibrary Loan service.*

First-Order representation:

$$\exists x_0 \ \exists x_1 \ (\texttt{loan}(x_0) \ \wedge \ \texttt{interlibrary}(x_1) \ \wedge \ \texttt{service}(x_1) \ \wedge \ \texttt{nn}(x_0, x_1)$$
$$\wedge \ \forall x_2 \ (\texttt{user}(x_2) \rightarrow \exists x_3 \ \exists x_4 \ \exists x_5 \ \exists x_6 \ (\texttt{student}(x_3) \wedge \texttt{possess}(x_5) \wedge \texttt{event}(x_5)$$
$$\wedge \ \texttt{card}(x_4) \ \wedge \ \texttt{nn}(x_3, x_4) \ \wedge \ \texttt{agent}(x_5, x_2) \ \wedge \ \texttt{patient}(x_5, x_4)$$
$$\wedge \ \texttt{use}(x_6) \ \wedge \ \texttt{event}(x_6) \ \wedge \ \texttt{agent}(x_6, x_2) \ \wedge \ \texttt{patient}(x_6, x_1))))$$

The Skolemized equ-satisfiable formula:

$$\texttt{loan}(a_0) \wedge \texttt{interlibrary}(a_1) \wedge \texttt{service}(a_1) \wedge \texttt{nn}(a_0, a_1)$$
$$\wedge \forall x_2 \, (\texttt{user}(x_2) \rightarrow (\texttt{student}(f^1(x_2)) \wedge \texttt{possess}(f^2(x_2)) \wedge \texttt{event}(f^2(x_2))$$
$$\wedge \texttt{card}(f^3(x_2)) \wedge \texttt{nn}(f^1(x_2), f^3(x_2)) \wedge \texttt{agent}(f^2(x_2), x_2) \wedge \texttt{patient}(f^2(x_2), f^3(x_2))$$
$$\wedge \texttt{use}(f^4(x_2)) \wedge \texttt{event}(f^4(x_2)) \wedge \texttt{agent}(f^4(x_2), x_2) \wedge \texttt{patient}(f^4(x_2), a_1))))$$

We have shown the first-order representations for lexicons which contain one Negation Introducer. We will show that the decidability property and the complexity class will be the same for representations of a lexicon which additionally is extended by Disjunction Introducer's.

4.3.5 The ECND Lexicon and ECN^SD Lexicon

Extending ECN Lexicon and ECN^S Lexicon with disjunction

If we look again at the definition of the Bernays-Schönfinkel-Ramsey (BSR) class 4.1.1 we will notice that the condition for membership to the BSR is that no existential quantifiers should appear in the scope of the universal quantifiers in the (prenex form) formula. If this "order" of quantifiers is maintained the remaining of the logic operators in the formula do no influence the membership to the BSR class.

The number of universal quantifiers in the first-order representations over the ECN Lexicon and ECN^S Lexicon can only be changed if additional Simple Negation Introducer, Special Negation Introducer, Implication Introducer or Superlatives are added to the lexicon. The number of universal quantifiers will not change if the ECN Lexicon or ECN^S Lexicon are extended with Disjunction Introducer's because a Disjunction Introducer can not introduce a universal quantification.

It is thus possible to extend the ECND Lexicon and the ECN^S Lexicon with arbitrarily many Disjunction Introducer's.

Definition 4.3.6. ECND Lexicon is the lexicon which contains all the words of natural language which are contained in the ECN Lexicon and arbitrarily many words/phrases previously defined as Disjunction Introducer.

Definition 4.3.7. ECN^SD **Lexicon** is the lexicon which contains all the words of natural language which are contained in the ECN^S Lexicon and arbitrarily many words/phrases previously defined as Disjunction Introducer.

All the first-order representations of sentences built from the ECND Lexicon (in prenex normal form and with all the negations appearing in front of atoms only and in Conjunctive Normal Form (CNF)) will fit the schema presented in 4.19. All the first-order representations of sentences built from the ECN^SD Lexicon (also prenex normal in Negated Normal Form (NNF) and in CNF) will also fit the schema presented in 4.19. The P_i denote unary and binary *literals*. The scope of the universal quantifier in 4.13 does not extend over any existential quantifiers.

$$\exists \mathbf{x} \forall \mathbf{y} (\bigvee P_1 \wedge \cdots \wedge \bigvee P_{n''}) \tag{4.19}$$

The properties of 4.19 will be exactly the same as those of 4.10 and 4.13.

Corollary 4.3.5. *For the first-order representations of sentences of the $ECND$ Lexicon and the ECN^SD Lexicon built by* BOXER*, satisfiability is decidable in $\sum_2^p$ time over the size of the formula.*

With this we complete the presentation of the restricted lexicons and their properties. Before proceed by presenting the coverage power of these lexicons, we will present the relationships which hold between each of these lexicons.

Relationships between the Lexicons for Answers

Each of the lexicons was defined on the level of a sentence, while our answers are considered to be texts of more sentences, hence it can be expected that different sentences in one answer belong to different lexicons. For this reason we want to see what is the relationship between each of the lexicons. If the lexicons are considered to be sets whose elements are the words/phrases of natural language which they contain (allow) we can express the relations which hold between the Lexicons in terms of set inclusion and set intersection.

Figure 4.1 shows the set relations between each of the Lexicons defined. (The gray fields denote an empty intersection.) From the level of a text we can consider that the text belongs to a given lexicon if all of its sentences belong to that lexicon. Consequently a text which, for example, has one sentence in the EC Lexicon another in the ECN lexicon and the third in the ECND lexicon will be considered to belong to the ECND lexicon because the ECND lexicon subsumes the EC and the ECN lexicon. A text which has one sentence in the ECN lexicon and another in the ECI lexicon will be considered to not belong to any of the lexicons because the sets ECN and ECI do not stand in a subsume relation.

Lastly it should be noted that although the lexicons which contain the simple negation and the special negation do not stand in a subsume relation, it will not be a problem to allow mixing of sentences in one text between these lexicons because their representations will belong to the same fragment of decidable formulas.

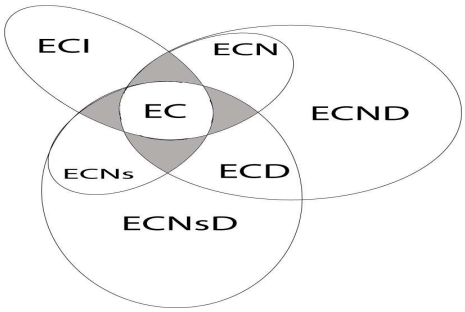

Figure 4.1: Set relations between the Restricted Lexicons for declarative sentences

4.3.6 Analysis of the Library FAQ Sheets

To get an estimate on which percentage of possible answers can be written using the restricted Lexicons for declarative sentences we defined, we made an analysis of the answers which can be found in the FAQ sheets of the university library of the Free University of Bozen-Bolzano which is the domain of our IQA System. The FAQ Sheet consists of 31 question and answer pairs written in natural language (English) by the staff of the Library of the University of Bolzano. The answers in total consist of 68 sentences.

The FAQ Sheets are small in size corpus however, the domain of our IQA system is exactly the material presented in the FAQ Sheets. In addition, the library staff was not given any instructions for writing the FAQ so the material is unrestricted and unstructured to better fit our purposes.

The results of our analysis are summed up in Table 4.1. It is interesting to see that 29% of the anwers belong solely in the EC Lexicon, which is rather surprising condsidering it is the most simple of all lexicons. Remarkable 86% of the sentences are expressed solely using the EC Lexicon. None of the answers or senteces belonged to the Lexicons which contain Special Negation Introducers. No Superlatives were found in the FAQ Sheets as well.

The one "mixed" answer had one sentence in the ECI lexicon and another in the ECD Lexicon. These kinds of problems can be reduced by making the answers as short as possible, namely by reducing the count of sentences in them.

In this particular case however, the Disjunction Introducer is inadequately used and it should be replaced with the coordinator *and*. The question answer pair was:

Example 4.3.6.

> Q: Is it possible to equally access the services at Bozen/Bolzano and Brixen/Bressanone?
>
> A: It is possible to equally access the services at Bozen/Bolzano and Brixen/Bressanone. ***All*** services are integrated at both sites. Therefore, it is possible to order material from one site ***or*** return material to both sites, regardless of the borrowing location.

□

Out of the remaining five uncovered answers, three we were able to easily rewrite to fit the Lexicons. We will show the cases in question.

Table 4.1: Lexicons coverage of the Bolzano library FAQ Sheet

	Answers Tot: 31	**Sentences** Tot: 68
EC Lexicon	9	43
ECD Lexicon	7	9
ECN Lexicon	3	4
ECND Lexicon	2	2
ECI Lexicon	4	5
Not Covered	6	5

Example 4.3.7.

> Q: How can the services of the library be accessed ?
> A: `Every` student, ***all*** teaching and administrative staff of the Free University Bozen/Bolzano can access the services of the University Library with their Student or Campus Card. People who are ***not*** affiliated with the Free University Bozen/Bolzano may use the University Library as external users. In order to receive a Library Card one has to present an identity card and pay the annual fee of 10,00 Euro.

The problematic first sentence can be split into two sentences which belong to the ECI Lexicon:

> S1: Every student of the Free University Bozen/Bolzano can access the services of the University Library with their Student Card.
> S2: All teaching and administrative staff of the Free University Bozen/Bolzano can access the services of the University Library with their Student or Campus Card.

The problem now is that the rewritten answer has sentences from the ECI and ECN lexicon. The ECN sentence and the last sentence will be rewritten by:

> S1: External Users can access the services of the University Library with their Library Card.
> S2: In order to receive a Library Card one has to present an identity card and pay the annual fee of 10,00 Euro.

The definition of External Users can be moved to a new question and answer pair. The new answer will belong to the ECN Lexicon.

□

The second case is the question answer pair in which the answer contains both an Implication and a Disjunction Introducer.

Example 4.3.8.

> Q: Who may use the Interlibrary Loan service ?
> A: ***All*** users who possess a Student Card, Campus Card ***or*** Library Card may use the Interlibrary Loan service .

The answer can be rewritten into three sentences of the ECI Lexicon.

> S1: ***All*** users who possess a Student Card may use the Interlibrary Loan service.
> S2: ***All*** users who possess a Campus Card may use the Interlibrary Loan service.
> S3: ***All*** users who possess a Library Card may use the Interlibrary Loan service.

□

The last answer we were able to rewrite contains more then one Implication Introducer in addition to a Disjunction Introducer.

Example 4.3.9.

> Q: How long are reservations valid ?
> A: The reservations are valid for 7 days with the exception of textbooks from Bozen/Bolzano marked with the number 15. These items have to be collected until 7 p.m. on the day of the reservation ***if*** the reservation has been made before 2 p.m. ***or*** until 7 p.m. on the following day ***if*** the reservation has been made after 2 p.m.

The second (and problematic) sentence of the answer can be rewritten in two sentences which both belong to the ECI lexicon:

> S1: The reservations have to be collected until 7 p.m. on the day of the reservation ***if*** the reservation has been made before 2 p.m.
> S2: The reservations have to be collected until 7 p.m. after the day of the reservation ***if*** the reservation has been made after 2 p.m.

□

In general, the sentences which do not belong to the defined Lexicons because they contain several Implication Introducer's are eligible for rewriting by splinting into several individual sentences by repeating the same information in each sentence for each quantified expression. The sentences which do not belong to the defined Lexicons because they contain a combination of the Implication Introducer and Disjunction Introducer have to be considered on individual bases.

The remaining two uncovered answers from the FAQ Sheets are sentences which contain both a Negation Introducer and an Implication Introducer. We will show one of them.

Example 4.3.10.

> Q: How can the loan periods be extended ? A: In order to extend loan periods one has to access the personal library account via the online catalogue. After entering the Library Number and the personal library password , a list of all material borrowed appears which also shows the respective loan periods. *To the right of the loan period there is an EXTEND button -* ***if*** *the loan period is extendable and the item has* ***not*** *been reserved in the meantime by another user.*

□

These kinds of answers express rules for negated concepts: *If not A then B.* or *If A then not B* and in general can not be rewritten to fit the Lexicons.

The lexicons we have defined here are by no means the most broad lexicons which yield decidable first-order representations. On the contrary, they are the most basic lexicons which can be considered. Future work should be done on further extending these basic lexicons, the most eminent of which would be to discover extensions of the ECI Lexicon with Negation Introducer's so that the sentences of form *If not A then B.* or *If A then not B* can be expressed.

4.4 Lexicon Restrictions for Natural Language Questions

We are going to work with only with the most basic lexicon for questions. This basic lexicon will be general enough to cover the majority of the questions we expect from the users but it is still simple enough to produce first-order representations which can be handled by our Logic Support Unit. The goal of the lexicon restrictions for questions is to obtain representations simple enough to guarantee that the entailment between them and the representations of sentences (built over the defined lexicons for declarative sentences) is decidable.

In Section 3.2 we introduced the semantic representation of questions and presented the general schema of first-order representation for questions 3.1. Here we will define the most basic lexicon for questions and then we will show decidability for entailment between representations of this lexicon and representations of the lexicons: EC, ECD, ECN, ECN^S, ECND, ECN^SD and ECI. Lastly we will present the analysis of several corpora of questions to show the coverage abilities of the QECD lexicon.

4.4.1 The QECD Lexicon

The lexicon for questions we will define contains all the words of natural language except the words (and phrases) previously defined as Simple Negation Introducer, Implication Introducer, Special Negation Introducer and Superlatives. We presuppose that this lexicon holds on the level of one single question which contains one single WH-phrase as a question word for the WH questions. The lexicon QECD for questions is exactly the same (allows the same lexical items) as the ECD Lexicon for declarative sentencec.

Definition 4.4.1. QECD Lexicon is the lexicon which contains all the words of natural language except the words (and phrases) previously defined as Simple Negation Introducer, Implication Introducer, Special Negation Introducer and Superlatives.

Recall the general schema of representations of questions 3.1 derived by BOXER. By not allowing any of the negation or implication introduces in the question, what we have achieved is that both φ and ψ from the representation schema 3.1 will contain conjunctions and disjunctions only. The next formula represents the general schema of a question over the QECD Lexicon (when φ and ψ are in Disjunctive Normal Form). P denotes the unary predicates, and Q denotes the binary predicates.

$$\exists \mathbf{y}\, (\underbrace{\bigwedge_{1}^{m}(\bigvee_{1}^{r} P(\mathbf{y}) \vee \bigvee_{1}^{s} Q(\mathbf{y})}_{dnf(\varphi)} \wedge D(x) \wedge \exists \mathbf{z} \underbrace{\bigwedge_{1}^{n}(\bigvee_{1}^{l} P(x,\mathbf{y},\mathbf{z}) \vee \bigvee_{1}^{k} Q(x,\mathbf{y},\mathbf{z}))}_{dnf(\psi)}))$$

In order to get a better insight to the properties of the semantic representations of questions we will re-write this last formula in Disjunctive Normal Form (DNF). After the expansion (with the laws of distribution $(a \wedge (c \vee d) = (a \wedge c) \vee (a \wedge d))$)we get the representation schema 4.20.

Assume that $dnf(\varphi)$ and $dnf(\psi)$ are respectively (α and β are positive atoms):

$$dnf(\varphi) = \alpha_1 \vee \alpha_2 \vee \cdots \vee \alpha_n \qquad dnf(\psi) = \beta_1 \vee \beta_2 \vee \cdots \vee \beta_m$$

$$\exists \mathbf{y} \exists \mathbf{z}(\alpha_1(\mathbf{y}) \wedge D(x) \wedge \beta_1(x, \mathbf{y}, \mathbf{z})) \vee \cdots \vee (\alpha_n(\mathbf{y}) \wedge D(x) \wedge \beta_m(x, \mathbf{y}, \mathbf{z})) \quad (4.20)$$

The general schema which describes all the first-order representations of Boolean Questions built from the QECD Lexicon (given in 4.21) is exactly as schema 4.20 when the atom which represents the domain of the question is omitted. The formula 4.21 *is closed*.

$$\exists \mathbf{y} \exists \mathbf{z}(\alpha_1(\mathbf{y}) \wedge \beta_1(\mathbf{y}, \mathbf{z})) \vee \cdots \vee (\alpha_n(\mathbf{y}) \wedge \beta_m(\mathbf{y}, \mathbf{z})) \quad (4.21)$$

4.4.2 Decidability of Entailment between questions and answers representations

We will now show the decidability of entailment between representations of questions expressed using the QECD Lexicon and representations of texts (sets of formulas) expressed using each of the defined lexicons. The entailment problem we will represent as a satisfiability problem. Assume that the first-order representation of the answer is A and that the first-order representation of the question is Q. For the case of WH Questions, Q will be an open formula. If we close the one free variable in Q with an existential quantifier, the following equivalence holds:

$$A \models Q \equiv A \cup \neg Q \models \bot$$

In the remaining of this section with Q we will represent the existentially closed first order representation of the question. We will also assume that all the formulas which represent all the sentences in the answer are already conjuncted to form the formula A.

Theorem 4.4.1. *Let Q be a first-order representation (obtained by* BOXER *)of a question in natural language over the QECD Lexicon. Let A be a first-order representation (obtained by* BOXER *) of a text in natural language over the EC Lexicon. The entailment $A \models Q$ is decidable in $\sum_2^p$ time over the size of A and Q.*

Proof. The formula of A will be an existentially closed conjunction of positive atoms (here denoted with φ).

$$\exists \mathbf{x}(\varphi_1(\mathbf{x}) \wedge \cdots \wedge \varphi_n(\mathbf{x}))$$

If the question is WH Question its first order representation will be:

$$\exists \mathbf{y} \exists \mathbf{z}(\alpha_1(\mathbf{y}) \wedge D(x^a) \wedge \beta_1(x, \mathbf{y}, \mathbf{z})) \vee \cdots \vee (\alpha_n(\mathbf{y}) \wedge D(x^a) \wedge \beta_m(x, \mathbf{y}, \mathbf{z}))$$

To decide the entailment trough satisfiability we first need to existentially close and negate Q. The result (written in NNF)is the following formula:

$$\forall x^a \forall \mathbf{y} \forall \mathbf{z}(\neg\alpha_1(\mathbf{y}) \vee \neg D(x^a) \vee \neg\beta_1(x^a, \mathbf{y}, \mathbf{z})) \wedge \cdots \wedge (\neg\alpha_n(\mathbf{y}) \vee D(x^a) \vee \neg\beta_m(x^a, \mathbf{y}, \mathbf{z}))$$

The entire formula $A \wedge \neg Q$ will be:

$$\exists \mathbf{x}(\varphi_1(\mathbf{x}) \wedge \cdots \wedge \varphi_n(\mathbf{x})) \wedge$$
$$\forall x^a \forall \mathbf{y} \forall \mathbf{z}(\neg \alpha_1(\mathbf{y}) \vee \neg D(x^a) \vee \neg \beta_1(x^a, \mathbf{y}, \mathbf{z}))$$
$$\wedge \cdots \wedge$$
$$(\neg \alpha_n(\mathbf{y}) \vee D(x^a) \vee \neg \beta_m(x^a, \mathbf{y}, \mathbf{z}))$$

In prenex form (there can be no variable sharing between the question and the answer formulas) the above formula will have the prefix $\exists \mathbf{x}\ \forall x^a\ \forall \mathbf{y}\ \forall \mathbf{z}$, hence the formula $A \wedge \neg Q$ will belong to the Bernays-Schönfinkel-Ramsey class of formulas and as such will be decidable for satisfiability in $\sum_2^p$ time over its size.

If the question is a Boolean Question its first order representation will be the closed formula:

$$\exists \mathbf{y} \exists \mathbf{z}(\alpha_1(\mathbf{y}) \wedge \beta_1(\mathbf{y}, \mathbf{z})) \vee \cdots \vee (\alpha_n(\mathbf{y}) \wedge \beta_m(\mathbf{y}, \mathbf{z}))$$

Now the formula $A \wedge \neg Q$ is much simpler and it still belongs to the Bernays-Schönfinkel-Ramsey class:

$$\exists \mathbf{x}(\varphi_1(\mathbf{x}) \wedge \cdots \wedge \varphi_n(\mathbf{x})) \wedge \forall \mathbf{y} \forall \mathbf{z}(\neg \alpha_1(\mathbf{y}) \vee \neg \beta_1(\mathbf{y}, \mathbf{z})) \wedge \cdots \wedge (\neg \alpha_n(\mathbf{y}) \vee \neg \beta_m(\mathbf{y}, \mathbf{z}))$$

□

If instead of with A we work with A^S (A when Skolemized) the entailment $A^S \models Q$ will again be decidable in $\sum_2^p$ time. Because A^S does not contain any quantifiers, the formula $A^S \models \neg Q$ will have the prefix $\forall x^a\ \forall \mathbf{y}\ \forall \mathbf{z}$ and will still belong in the BSR class of formulas.

Corollary 4.4.1. *Let Q be as defined in Theorem 4.4.1 and A is now the first-order representation obtained by* BOXER *of a text over the ECD Lexicon. The rest of the assumptions for A from Theorem 4.4.1 hold. The entailment $A \models Q$ is decidable in $\sum_2^p$ time over the size of A and Q.*

The formula of A will now be the existentially closed conjunction of disjunctions of positive atoms (assuming we have reduced it to its CNF):

$$\exists \mathbf{x}((\varphi'_1(\mathbf{x}) \vee \cdots \vee \varphi'_n(\mathbf{x})) \wedge \cdots \wedge (\varphi^k_1(\mathbf{x}) \vee \cdots \vee \varphi^k_1(\mathbf{x})))$$

It is evident that the formula $A \wedge \neg Q$ will remain in the Bernays-Schönfinkel-Ramsey class both for Boolean and WH Question because its prefix will be the same as the prefix of the final formula from Theorem 4.4.1.

Theorem 4.4.2. *Let Q be as defined in Theorem 4.4.1 and let A be the first-order representation obtained by* BOXER *of a text over the ECN Lexicon. The rest of the assumptions for A from Theorem 4.4.1 hold. The entailment $A \models Q$ is decidable in $\sum_2^p$ time over the size of A and Q.*

Proof. The formula of A will now be a formula of the Bernays-Schönfinkel-Ramsey class(as shown in Formula 4.10). We will write it shortly as:

$$\exists \mathbf{x}_i \forall \mathbf{x}_i\ \varphi(\mathbf{x}), \mathbf{x}_i \cup \mathbf{x}_i = \mathbf{x}$$

The formulas for a the Question Q will be the same as in Theorem 4.4.1 and consequently $\neg Q$ will be the same. For the WH-Questions we can shortly write it as :

$$\forall x^a \forall \mathbf{y} \forall \mathbf{z} \psi(x^a, \mathbf{y}, \mathbf{z})$$

The corresponding $A \wedge \neg Q$ formula will be:

$$\exists \mathbf{x}_i \forall x^a \forall \mathbf{x}_i \forall \mathbf{y} \forall \mathbf{z}\ \varphi(\mathbf{x}) \wedge \psi(x^a, \mathbf{y}, \mathbf{z}), \mathbf{x}_i \cup \mathbf{x}_i = \mathbf{x}$$

The corresponding $A \wedge \neg Q$ formula when Q is a Boolean question will be the same as above but without the variable x^a:

$$\exists \mathbf{x}_i \forall \mathbf{x}_i \forall \mathbf{y} \forall \mathbf{z}\ \varphi(\mathbf{x}) \wedge \psi'(\mathbf{y}, \mathbf{z}), \mathbf{x}_i \cup \mathbf{x}_i = \mathbf{x}$$

It is evident that both formula $A \wedge \neg Q$ will remain in the Bernays-Schönfinkel-Ramsey class. □

The entailment $A^S \models \neg Q$ will also be decidable in $\sum_2^p$ time over the size, for the same reasons as in the case of the EC Lexicon. This will hold for the following two lexicons as well (the ECN and ECN^S Lexicon).

Corollary 4.4.2. *Let Q be as defined in Theorem 4.4.1 and let A be the first-order representation obtained by* BOXER *of a text over the ECN^S Lexicon. The rest of the assumptions for A from Theorem 4.4.1 hold. The entailment $A \models Q$ is decidable in $\sum_2^p$ time over the size of A and Q.*

Exactly the same proof as presented for Theorem 4.4.2 can be applied here.

The case of the ECND Lexicon and the case of the ECN^SD Lexicon will be considered together, because the proof approach again is the same as the one presented in Theorem 4.4.1.

Corollary 4.4.3. *Let Q be as defined in Theorem 4.4.1 and let A be the first-order representation obtained by* BOXER *of a text over the ECND Lexicon* ***or*** *over the ECN^SD Lexicon. The rest of the assumptions for A from Theorem 4.4.1 hold. The entailment $A \models Q$ is decidable in $\sum_2^p$ time over the size of A and Q.*

Regardless if the A derives from the ECND Lexicon or from the ECN^SD Lexicon it will be a formula of the Bernays-Schönfinkel-Ramsey class, hence the exact the same proof as presented for Theorem 4.4.2 can be applied here as well.

The case of entailment for the ECI Lexicon is more interesting.

Theorem 4.4.3. *Let Q be as defined in Theorem 4.4.1 and let A be the first-order representation obtained by* BOXER *of a text over the ECI Lexicon. The rest of the assumptions for A from Theorem 4.4.1 hold. The entailment $A \models Q$ is decidable in* EXPTIME *time over the joint size of A and Q.*

Proof. In Theorem 4.3.1 we showed that a representation from a sentence of the ECI Lexicon can be reduced to an equi-satisfiable formula of the Loosely Guarded Fragment (LGF). The same can be done for a conjunction of sentences of the ECI Lexicon which is what A in fact is. Let us denote by A^G the equi-sat LGF formula obtain from A in polynomial time as described in the proof of Theorem 4.3.1.

To show that the formula $A \models Q$ is decidable in EXPTIME time we need to show that the formula $A^G \wedge \neg Q$ belongs to the LGF (the bounded arity property of the predicate symbols does not change). The LGF is closed under negation and conjunction, so it is enough to show that the formula $\neg Q$ belongs to the LGF.

If the question is a WH Question its first-order representation will be:

$$\exists x^a \exists \mathbf{y} \exists \mathbf{z}(\alpha_1(\mathbf{y}) \wedge D(x^a) \wedge \beta_1(x, \mathbf{y}, \mathbf{z})) \vee \cdots \vee \exists \mathbf{y} \exists \mathbf{z}(\alpha_n(\mathbf{y}) \wedge D(x^a) \wedge \beta_m(x, \mathbf{y}, \mathbf{z}))$$

This formula is in the LGF because it is a disjunction of LGF formulas. It is evident from Definition 4.1.3 that the formula

$$\exists x^a \exists \mathbf{y} \exists \mathbf{z}(\underbrace{\alpha_i(\mathbf{y}) \wedge D(x^a)}_{\psi} \wedge \underbrace{\beta_j(x, \mathbf{y}, \mathbf{z})}_{\alpha})$$

satisfies the conditions of the LGF. (The under-brace markings α and ψ serve to denote the connection between the formula in Definition 4.1.3.)

To be more precise the formula Q will belong to the Guarded Fragment, but every formula from the GF is also in the LGF, because the LGF is obtained by relaxation of the conditions of the GF.

If the question is a Boolean Question its first-order representation will still be a disjunction of LGF formulas as the representation of a WH Question:

$$\exists \mathbf{y} \exists \mathbf{z}(\alpha_1(\mathbf{y}) \wedge \beta_1(\mathbf{y}, \mathbf{z})) \vee \cdots \vee \exists \mathbf{y} \exists \mathbf{z}(\alpha_n(\mathbf{y}) \wedge \beta_m(\mathbf{y}, \mathbf{z}))$$

□

In the case when A is a first order representation of the ECI Lexicon, the formula A^S will contain function symbols and we will not be able to reduce it to the LGF to prove satisfiability properties for the entailment $A^S \models \neg Q$ in the same way above. But because A^S is equi-satisfiable with A the same complexity result for $A^S \models \neg Q$ will be the same as for $A \models \neg Q$.

As a conclusion of the introspection of the properties of the QECD Lexicon we present Table 4.2where the decidability for entailment results are summed up. The table shows that the highest complexity bound for our questions and answers is EXPTIME. We do not present the results for the A^S formulas because they are the same as for the A formulas.

Table 4.2: Complexity of entailment between the questions and answers of the defined lexicons

	$\models$ **QECD**
EC Lexicon	$\sum_2^\pi$
ECD Lexicon	$\sum_2^\pi$
ECN Lexicon	$\sum_2^\pi$
ECND Lexicon	$\sum_2^\pi$
ECN^S Lexicon	$\sum_2^\pi$
ECN^{S}D Lexicon	$\sum_2^\pi$
ECI Lexicon	EXPTIME

To complete the work of this chapter we will attempt to answer the question of which and how many of the natural language questions can be expressed by using only the QECD Lexicon.

4.4.3 Analysis of Questions asked by users

The question answering system for which we wish to build a Logic Support Unit is interactive and generally speaking both the system and the user can ask questions. While we can completely control the questions posed by the system and impose lexicon restrictions on them, there is no way to directly control the language of the questions posed by the user, which will be the source of most of the questions.

In order to answer the question of how complex are the structures of natural language questions and how frequent are the questions which can be posed by using the QECD Lexicon only, we will here present the results of an analysis of several corpora of questions presented in [10].

The corpora consisted of different domains and asked in different settings:

Clinical Questions: contains users' questions on the clinical domain, mostly asked by doctors to colleagues; 435 questions, vocabulary: 3495, total tokens: 40489 (questions with introduction).

Answer.com: contains questions on different topics (e.g., art, sport, computers) asked by internet users; 444 questions[3], vocabulary: 1639, total tokens: 5791 (without introduction)[4].

TREC: the TREC 2004 corpus that contains 408 questions.

The result of the analysis is shown in Figure 4.2. The chart shows the frequency of each class of terms in the three corpora. Frequency values refer to the normalized relative frequency (number of tokens/total word count multiplied by 100). The details about the number of questions in which universal quantifiers and negations occur (the questions not belonging to the QECD Lexicon) are given in Table 4.3.

The results show that the number of questions which can not be covered by the QECD fragment, namely those which contain negation and universal quantification, is low in percentage – across the three corpora considered, a total of 8.78% (86 out of 1287) do not belong to the QECD Fragment.

[3] `http://clinques.nlm.nih.gov/`

[4] `http://wiki.answers.com/Q/WikiFAQs:Finding_Questions_to_Answer`

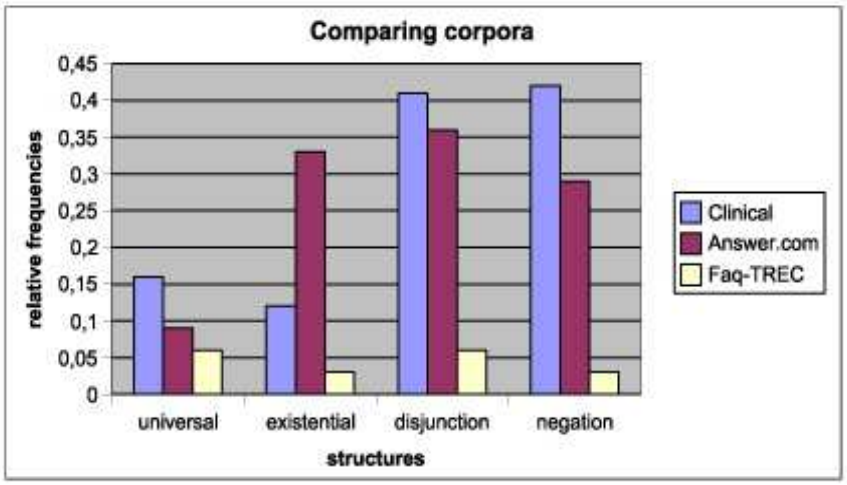

Figure 4.2: Summary of the analysis

Table 4.3: Number of questions

	Clinical Questions Tot: 435	**Answer.com** Tot: 444	**TREC** Tot: 408
universal	12	6	2
negation	52	13	1

Chapter 5

Reasoning for IQA using Answer Set Programming

The second problem handled in this thesis is the development of the reasoning module of the proposed Logic Support Unit for the IQA System.

The main goal of the LSU is to verify that a possible answer is an answer to a question. It also has to be able to extract the specific answers from a verified answer. We want the LSU to be able to extract all possible specific answers when the question is a WH Question and to be able to give justification the the "no" specific answer when the question is Boolean:

WH Question In the case of WH Questions it is possible that a verified possible answer A can contain more then one specific answer. It will be most informative for the user to retrieve all these specific answers. In the case when more then one of the possible answers A_1, A_2, $\cdots$ is verified to be an answer for the WH Question we want to retrieve the union of all the specific answers contained in all the confirmed answers A_1, A_2, $\cdots$

Boolean Question In the case of an answer "no" to a Boolean Question, in a addition to the specific answer "no", we want the Reasoning Module to be able provide the information of exactly which part of the answer is inconsistent with the question, namely, why the answer is "no". This is important from the view point of Interactive question answering because when a users question is answered with "no", knowing why this is so, can assist the user to modify his query and retrieve successfully the information he needs.

In order for the LSU to be able to perform all the reasoning tasks required from it we have to develop a suitable formalism to represent these task in terms of logic. We will briefly recapitulate why the other formalisms we considered were not adequate for the purposes of the LSU.

The formalism stated in Definition 3.3.1 of Section 3.3 only allows us to verify the answer, but does not offer the possibility to extract the specific answer.

In Definition 3.3.2 of Section 3.3 we defined what is an answer and a specific answer to a question given its first-order representation Q in terms of logic entailment. Although this formalism allows for both the answer verification

and specific answer extraction tasks to be accomplished, it is hard to practically implement.

In order to verify that Q is answered by A it is enough to show that the entailment $A \models Q$ holds. This can be achieved by checking if the formula $A \wedge \neg\exists x\ (Q)$ is unsatisfiable. Satisfiability for a formula can be checked by using a first order theorem prover, however this approach will not satisfy the remaining of our desiderata for the Reasoning Module. The theorem prover will tells us only if the formula $A \wedge \neg\exists x(Q)$ is unsatisfiable, however we would need to have the models of A as well in order to be able to extract the specific answer. We would not be able to retrieve all the specific answers from the proof it will derive because the specific answer x had to be existentially closed.

The reasoning module of the LSU has to to be able to build all the models of A in order to extract the specific answer from the models of Q. In order for the automatic derivation of all the possible models to be possible the formula has have the finite model property and the tree model property [35], [18]. In Chapter 4 we specified lexicons for the question (QECD) and the possible answers (EC, ECD, ECN, ECN^S, ECND, ECN^SD or ECI) which yield decidable first-order representations. The first-order representations of the possible answers which are built over the lexicons EC, ECD, ECN, ECN^S, ECND and ECN^SD have the finite model property and the tree model property but even when A is a formula over these lexicons it is difficult to automatically derive all the models of A.

We have chosen to represent the problem of verifying a possible answer and extracting a specific answer in the setting of Disjunctive Logic Programs under answer set semantics (in the remaining of this chapter simply Answer Set Programming (ASP)). We chose to use the following approach.

In Section 3.3 we present that the formula A^S which is obtained from the representation of the answer A is more suitable for the extraction of specific answer. We represent how to translate this formula to a logically equivalent (possibly infinite) logic program P^A comprised of disjunctive logic program rules. The translation procedure we present is suited for the representations of possible answers built from the lexicons EC, ECD, ECN, ECN^S, ECND, ECN^SD and ECI.

The question Q (built over the QECD Lexicon) is represented with a finite set of logically equivalent logic program rules P_L.

The problem of verification of a possible answer is represented trough the task of finding answer sets to the program $P^A \cup P_L$. The task of extracting all the specific answers from all the verified answers, as well as the task of retrieving a justification for a "no" specific answer is represented as a problem of analysis of the obtained answer sets.

For possible answers built over the Lexicons EC, ECD, ECN, ECN^S,ECND and ECN^S, the program P^A will be finite and the answer sets of the program $P^A \cup P_L$ can be obtained using the DLV System [44]. For possible answers built over the ECI Lexicon, the program P^A will be infinite but finitary, and answer sets of the program $P^A \cup P_L$ can be calculated using an extension of the DLV System [19].

As already introduced in Section 2, the reasoning for Question Answering requires more knowledge about the world then what is expressed in the possible answer. The need for a knowledge base that contains supporting information needed for deciding on an answer is eminent. At the end of this chapter we will

only briefly describe what are the minimum required continence of supporting knowledge base for our LSU. The development of a supporting knowledge base for an IQA system even over the small limited domain of the university library is no small task and requiters far more attention then what we can provide in the scope of this theses.

Before showing how Disjunctive Logic Programming under answer set semantics can be used as a setting for our Reasoning Module the most basic notions of ASP will be introduced.

5.1 Basics of Syntax and Semantics of ASP

We begin by presenting the syntax of the disjunctive logic rules. As in the Prolog convention, strings starting with uppercase letters denote variables, while those starting with lower case letters denote constants.

A *disjunctive rule* (rule, for short) r is a formula of the shape $H :- B$ that stands for "if B deduce H" where H (head of the rule) and B (body of the rule) are as following:

$$a_1 \vee \cdots \vee a_m \ :- \ b_1, \cdots, b_k, not\ b_{k+1}, \cdots, not\ b_n.$$

$a_1, \cdots, a_n, b_1, \cdots, b_m$ are classical literals, and $n \geq 0$, $m \geq k \geq 0$. By classical literals we mean atoms or strongly negated atoms. *not* is *negation as failure* or *default negation*. We will also refer to it as *weak negation*.

The rule is called a

fact if $n = 0$ and $m \geq 1$.

(strong)constraint if $m = 0$ and $n \geq 1$.

basic if $n = k$ and $m \geq 1$.

non-disjunctive if $m = 1$.

normal if it is non-disjunctive and $a_1, \cdots, a_n, b_1, \cdots, b_m$ are positive atoms.

ground if all the literals grounded.

A logic **program** P is a set of rules r_i. A program is disjunctive, basic, non-disjunctive, normal or ground if all of the rules in it are disjunctive, basic, non-disjunctive, normal or ground.

Before proceeding with the semantic the following notions need to be introduces as well.

Herbrand Universe U_P of any logic program P is the set of all the terms which can be built over the constant symbols and function symbols which appear in P

Herbrand Base B_P of any logic program P is the set of all ground (classical) literals constructible from the predicate symbols appearing in P and the terms of U_P

Herbrand Interpretation I of any logic program P is a subset of B_P.

The answer set semantics, or answer set programming, relies on algorithms that compute *stable models*. The underlying semantics is the stable models semantics as introduced by Gelfond and Lifschitz [30]. The stable model semantics defines a family of models ("answers") of a logic program and the minimal of these models according to the knowledge of information ordering is returned.

We will describe here only the answer set semantics of programs which are *normal* and *basic*. The answer set semantics of all other programs consists of first reducing the rules of the program to normal basic rules and then treating the program as a regular normal basic program. The procedure of reducing disjunctive programs to normal basic programs can be found in [44].

The work of [30] includes programs which contain function symbols, however in [44], which is based on semantics of stable model introduced in [30], the logic programs considered do not contain function symbols.

As we mentioned a (normal) logic program P, is a set of rules r:

$$a :- b_1, \cdots, b_k, \neg c_1, \cdots, \neg c_n.$$

where $a, b_1, \cdots, b_m$ and $c_1, \cdots, c_n$ are positive atoms.

Let us denote by B_P the Herbrand Base of P and by I ($I \subseteq B_P$) an (Herbrand) interpretation for P. By `ground(`P`)` we will denote the set of all ground(variable free) rules of P. We define a transformation `pos(`P`,`I`)` to be a a transformation which, given a logic program P and a Herbrand Interpretation I for P returns a positive logic program P' by applying the following procedure:

P`'= ∅ ;`
`for each` r `=` $a :- b_1, \cdots, b_k, \neg c_1, \cdots, \neg c_n.$ $\in$ `ground(`P`) do`
`if` $c_i \in I$`,` i `=` $1, \ldots n$ `then`
P`'=` P`'` $\cup \{a :- b_1, \cdots, b_k.\}$

Note that the program P' will be infinite in general if it has function symbols because it will be grounded over an infinite Herbrand Base.

Definition 5.1.1. A Herbrand Interpretation I of P is a *stable model for* P if:

$$I = T^{\infty}_{pos(P,I)}(\emptyset),$$

i.e. I is the *least fixed point* of the positive logic program defined by `pos(`P`,`I`)`.

Answer Sets of a disjunctive logic program P are its minimal stable models, with respect to set inclusion.

A disjunctive logic program may have one answer set, more then one answer set or no answer sets at all. Whether or not there will be one or more answer sets is directly determined by the presence of disjunctions in the rules. The complexity in which an answer set will be derived (apart from the efficiency of a particular solver used) directly depends on the number of disjunctions and weak negations in the rules.

An answer set of a program P is the set of all the facts which are entailed by P (the consequences of P).

5.2 Representing Questions and Answer with Disjunctive Logic Program Rules

In this section we will show how to transform the formula A^S deriving from a possible answer and the first-order representation of the question Q with disjunctive logic program rules. We assume that Q corresponds to a question built over the QECD Lexicon.

Translating the Answer

We will transform the formula A^S (written in Conjunctive Normal Form) into a logically equivalent a disjunctive logic program P_{CA} by using the following algorithm:

- P_{CA} `:=` $\emptyset$
- `for each conjunct` K_j `in the formula` A^S `do:`
  `if` K_j `contains only ground atoms then` P_{CA} `:=` $P_{CA} \cup \{K_j.\}$
  `else`

  1. `if` $K_j = \alpha_1 \vee \cdots \vee \alpha_k$ (all positive atoms)
     `then`

     P_{CA}`=` $P_{CA} \cup \{\alpha_1 \vee \cdots \vee \alpha_k.\}$
  2. `if` K_j `=` $-\alpha_1 \vee \cdots \vee \neg\alpha_k$ (all negative atoms) `then`

     $P_{CA} = P_{CA} \cup \{-\alpha_1 \vee \cdots \vee \neg\alpha_k :-.\}$
  3. `if` K_j `=` $\alpha_1 \vee \cdots \vee \alpha_k \vee \neg\beta_1 \vee \cdots \vee \neg\beta_m$
     `then`
     P_{CA} `=` $P_{CA} \cup \{\alpha_1 \vee \cdots \vee \alpha_k :- \beta_1, \cdots, \beta_m.\}$

The logic programs under answer set semantics are treated separately and do not allow for variable sharing among rules. This means that a variable X in a rule r_1 and a variable X in a rule r_2 in the same program will be treated independently as two different variables.

The algorithm we described builds one rule for each conjunct K_j of the formula A. Because it can happen that in A a variable is shared between conjuncts we need to add *constraints* to the program P_{CA} to ensure logical equivalence with A.

For each two literals α_1 and α_2 in A which share a variable x we add the to P_{CA} the constraint:

$$:\!-\alpha_1\texttt{(X1),}\alpha_2\texttt{(X2),X1!=X2.}$$

We denote with P_A the resulting program to which all of the above constraints were added.

Translating the Question

Recall that the question Q is represented as the query $q(x)$ where the x sought by the query is the specific answer to the question. In the query $q(x)$, α_i and β_j are conjunctions of positive atoms: $\alpha_i = a_i^1 \wedge \cdots \wedge a_i^r$ and $\beta_j = b_j^1 \wedge \cdots \wedge b_j^s$

$$q(x) = \{x | \exists \mathbf{y} \exists \mathbf{z} (\underbrace{\alpha_1(\mathbf{y}) \wedge D(x) \wedge \beta_1(x, \mathbf{y}, \mathbf{z})) \vee \cdots \vee \exists \mathbf{y} \exists \mathbf{z} (\alpha_n(\mathbf{y}) \wedge D(x) \wedge \beta_m(x, \mathbf{y}, \mathbf{z}))}_{\psi})\}$$

Semantically the query $q(x)$ is a union of the individual queries $q_1(x), \cdots, q_1(x)$:

$$q_1(x) = \{x | \exists \mathbf{y} \exists \mathbf{z} (\underbrace{\alpha_1(\mathbf{y}) \wedge D(x) \wedge \beta_1(x, \mathbf{y}, \mathbf{z})}_{\psi_1})\}$$

...

$$q_m(x) = \{x | \exists \mathbf{y} \exists \mathbf{z} (\underbrace{\alpha_n(\mathbf{y}) \wedge D(x) \wedge \beta_m(x, \mathbf{y}, \mathbf{z})}_{\psi_m})\}$$

This semantic connection allows us to represent each $q_i(x)$ as an individual logic program P_Q^i and build the final P_Q as a union of all the P_Q^i.

We will now show an algorithm which takes the query $q(x)$ as an input and returns a set of logic rules P_Q (depending on the type of the question).

given $q(x)$ test if $D(x)$ exists in ψ:
if YES do

P_Q := $\emptyset$

- for each disjunct ψ_j in ψ of $q(x)$ do:
  P_Q := $P_Q \cup$ {query_answer(Y):-$D(Y)$, a_j^1, ... a_j^r, b_j^1, ..., b_j^s.}
  else

if NO do

- for each disjunct q_j in $q(x)$ do:
  P_Q := $P_Q \cup$ {query_answer(y):- a_j^1, ... a_j^r, b_j^1, ..., b_j^s.}

  P_Q := $P_Q \cup$ {query_answer(n_1):- $\neg a_j^1$.}
  ...
  P_Q := $P_Q \cup$ {query_answer(n_r):- $\neg a_j^r$.}
  ...
  P_Q := $P_Q \cup$ {query_answer(n_r+1):- $\neg b_j^1$.}
  ...
  P_Q := c $\cup$ {query_answer(n_r+s):-$\neg b_j^s$.}

We will state the properties of P_Q.

The logic program P_Q will not contain function symbols. Because it is derived from a first-order formula which does not contain constant symbols, the only constant symbols in P_Q are the y and $n\_1, \ldots n\_r + s$ appearing in the head of the rules created for Boolean Questions. The bodies of all the rules in P_Q do not contain constant symbols.

In the query $q(x)$ all the variables in ψ are existentially closed which means $\exists \mathbf{y} \exists \mathbf{z} \psi$ will be satisfied if there is *at least one* interpretation for $\mathbf{y}$ and $\mathbf{z}$ which satisfies ψ. This is consistent with the semantic of the logic rules, namely a rule in P_Q will fire if there is *at least one interpretation* for the variables in the bodies of the rules. The bodies of the rules in P_Q are a translation of ψ.

With this we complete the translation to logic program rules of the candidate answer. Because the translation algorithm is built over a CNF of a first-order formula without relying on any of the properties of the formulas which would be derived from the Lexicons defined in Chapter 4, this same algorithm can be used without modifications on any possible extensions on the Lexicons defined in the future.

We can now proceed to show how the problem of verifying an answer for a question and extracting a specific answer can be solved by analysis of the program $P_Q \cup P_A$.

5.3 Question Answering in Terms of Analysis of Answer Sets

Given a possible answer represented with the logic program P_A and a question represented with a logic program Q we will show how to retrieve a verification for an answer to a question, a specific answer as well as a justification for the specific boolean answers. The underlining idea of this approach is the same as in [7].

The problem of deciding if a possible answer represented with P_A is certainly an answer to a question represented with P_Q is stated in Definition 5.3.1.

Definition 5.3.1. Let P_A be a logic program which represents a possible answer A^{TL} and let P_Q be a logic program which represents a question Q^{TL}. A^{TL} is an answer for Q^{TL} if and only if at least one of the answer sets of $P_A \cup P_Q$ contains the (grounded) fact `query_answer`.

We will elaborate the intuition behind this definition. The answer sets of the program P_A are the minimal Herbrand Models of P_A. When a formula φ admits a minimal model $\mathcal{M}$ in order to verify that $\varphi \models \psi$ (ψ being also a formula) it is enough to verify that $\mathcal{M} \models \psi$. Because P_A is logically equivalent to A^S, the minimal Herbrand model of P_A is a minimal Herbrand model of A so $A^S \models Q$ if the minimal Herbrand model of P_A is a model for Q.

Because P_Q and Q are logically equivalent, the answer set of P_A will be a model for Q if and only if at least one of the rules in P_Q is executed ("fires"). If a rule is executed the (grounded) fact `query_answer` will be present in the answer set (assume for now there is only one answer set) of $P_A \cup P_Q$.

However, the program $P_A \cup P_Q$ can have more then one answer set. This will happen when there is at least one disjunctive rule in $P_A \cup P_Q$. In this case we would find it sufficient proof of $A^S \models Q$ if the (grounded) fact `query_answer` appears in *at least one of the answer sets* derived.

To answer to the query $q(x)$ is in fact to find a possible grounding to the atom `query_answer`. By considering all the answer sets of $P_A \cup P_Q$ we are in fact retrieving the union of the possible grounding to the atom `query_answer`. This approach to the retrieval of a query answer is also known as *brave reasoning*. Opposite to brave reasoning is the *cautious reasoning* which is what we are doing if we only consider a possible grounding c as an answer if `query_answer(c)` appears in all the answer sets of $P_A \cup P_Q$.

We chose to work under brave reasoning because we want to consider all possible answers even when they are partial and hold only under certain conditions. To illustrate this we will use an example.

Assume that the question asked is "*Can I borrow a DVD?*" and the possible answer is "A user can borrow a DVD on Monday or a book on Tuesday." In this case, due to the disjunction, the program $P_A \cup P_Q$ will have two answer sets. In one of them the fact `query_answer(y)` will appear and in the other no `query_answer` atom will be derived at all. Evidently we want to inform the user that a DVD can be borrowed, because even when this is only possible on a Monday it is still possible.

With this we have reduced the decision of whether an answer certainly answers a question to tho a decision of whether a given fact is present in the answer set of the union of the corresponding programs. We will now direct our attention to the task of extracting the specific answer to a question by using the answer sets of $P_A \cup P_Q$.

We have defined the specific answer to a question in Definition 3.3.2; that would be the variable x from $q(x)$. As we mentioned, to answer the query $q(x)$ is in fact to find a possible grounding to the atom `query_answer`. Hence, all the specific answers to a question will be the union of all the possible grounding to the atom `query_answer` found in all the answer sets of $P_A \cup P_Q$. This is fairly easy to achieve, but it is not sufficiently informative.

In accordance of the discussion in Section 3.3, the individual x we are searching for, named by the retrieved constant which grounds `query_answer`, is fully identified by the properties which hold for that individual. This means that, in order to fully identify x we do not only need its name c but also all the atoms from the answer sets of $P_A \cup P_Q$ in whose grounding c appears.

If we look at the rules of the program P_Q we can see that `query_answer` can be grounded by *three kinds of constant symbols*:

1. In the case of an answer to a WH Question, `query_answer` will be grounded to the same constants which grounds the atom of the domain `D`.

2. In the case of an `yes` answer to a Boolean Question, `query_answer` will be grounded by the constant y.

3. In the case of an `no` answer to a Boolean Question, `query_answer` will be grounded by at least one of the constants n_1,...,$n\_r + s$.

For one answer set we will describe how to retrieve the specific answer(s) for each of the cases (1)-(3).

In the case (1), the specific answer to the answered question will be represented by the set $\mathcal{S}_a$ of all the predicates the answer set which contains `query_answer(c)` which have c in their grounding. If c appears in the binary predicate $P(c, d)$ we also need to retrieve and add to $\mathcal{S}_a$ all the unary predicates which are grounded by the constant d to complete the information. If there is more then one `query_answer` atom, namely `query_answer(c_1)`, ..., `query_answer(c_n)` in the answer set, we repeat the above described actions for each `c_i` producing as many sets $\mathcal{S}_a$ as there are specific answers.

In the case (2), the specific answer to the answered question will be `yes`. The justification for this answer will be the set $\mathcal{S}_j$ of the grounded atoms from the answer set which appear in the body of the rule in P_Q which has `query_answer(y)` as its head. There can be only one such rule in P_Q. If in one answer set there is more then one `query_answer(y)` atom we retrieve the sets $\mathcal{S}_j$ with justifications for each of them.

In the case (3), the specific answer to the answered question will be `no`. According to the number i of the grounding constant n_i, the justification for this answer will be the set $\mathcal{S}_j^i$ of the grounded atoms from the answer set which appear in the body of the rule in P_Q which has `query_answer(n_i)` as its head. For each i there can be only one such rule in P_Q. If there is more then one different i, the justification will be the union $\mathcal{S}_j$ of all the individual $\mathcal{S}_j^i$ generated.

It should be noted that the program P_A will not have an answer set if the rules in it are inconsistent. This corresponds to the situation when the answer from which P_A derives from contains contradictory information. We will assume that all the answers we are dealing with are consistent, consequently P_A will always have at least one answer set. We will assume also that the representations of the answers are always satisfiable (the answers convey a true information) hence A^S will have at least one model. If A^S had no models and Q has models then $A^S \models Q$ will trivially hold.

P_A can also have the empty set as an answer set. This will happen when the rules in P_A are consistent but no facts can be derived from them. For an example, the following program will have the empty set as an answer set.

```
material(X):-book(X).
thing(X):-material(X).
```

If this happens then the minimal model of A^S is the empty set and the $A^S \models Q$ will trivially never hold.

The programs P_A which are obtained as a translation the A^S of the lexicons EC, ECD, ECN, ECN^S,ECND and ECN^S will be finite (because the A^S formula does not contain function symbols) and the answers sets for $P_A \cup P_Q$ can be evaluated using any ASP solver which handles disjunctive logic programs. When the program P_A is obtained as a translation of the A^S which derives from the ECI Lexicon, P_A will be infinite because A^S will contain function symbols in its signature. We will look more into this case.

The case of Programs with function symbols

So far we have spoken of disjunctive logic programs under answer set semantics without making a difference whether they contain function symbols or not. Recall that in the case when our answer is a text over the ECI lexicon, the formula A^S will contain function symbols. Because of the presence of the function symbols, the Herbrand Base will be infinite, and the program, in the general case will have infinitely many rules. The logic programs which contain function symbols are usually referred to as *infinite programs*. However, for certain programs which have infinite Herbrand Bases answer sets can be evaluated. The specifics of such programs are treated in [12].

We will derive infinite programs P_A from the answers which belong to the ECI Lexicon. Generally speaking determining if there is an answer set for an infinite program is indecisive, so there is a need to separately consider the case of P_A corresponding to an ECI Lexicon answer. We will show that in order to derive an answer set for P_A it is not necessary to consider the entire infinite program.

Let α be the first-order representation of an answer from the ECI lexicon. In accordance to the analysis in Section 4.3.4, φ will be of form:

$$\exists \mathbf{x}\ (\phi(\mathbf{x})\ \wedge\ \forall\ \mathbf{y}\ (\varphi(\mathbf{x}_i, \mathbf{y}) \rightarrow \exists\ \mathbf{z}(\eta(\mathbf{x}_j, \mathbf{y}, \mathbf{z})))),\ \mathbf{x}_i \subseteq \mathbf{x},\ \mathbf{x}_j \subseteq \mathbf{x} \tag{5.1}$$

When we Skolemize it α will become (($\mathbf{c}$) is a tuple of constants, $\mathbf{y}$ is a tuple of variables and $\mathbf{f}$ is set of functions):

$$\phi(\mathbf{c})\ \wedge\ \forall\ \mathbf{y}\ (\varphi(\mathbf{c}, \mathbf{y}) \rightarrow \eta(\mathbf{c}, \mathbf{y}, \mathbf{f}(\mathbf{y}))) \tag{5.2}$$

The Herbrand Base of the Formula 5.2 will be the following set:

$\{\phi(\mathbf{c}), \varphi(\mathbf{c}, \mathbf{c}), \eta(\mathbf{c}, \mathbf{c}, \mathbf{f}(\mathbf{c})), \varphi(\mathbf{c}, \mathbf{f}(\mathbf{c}), \eta(\mathbf{c}, \mathbf{f}(\mathbf{c}), \mathbf{f}(\mathbf{f}(\mathbf{c}))), \ldots\}$

The possible Herbrand Models of the Formula 5.2 are:

$\{\phi(\mathbf{c}), \varphi(\mathbf{c}, \mathbf{c}), \eta(\mathbf{c}, \mathbf{c}, \mathbf{f}(\mathbf{c}))\}$

$\{\phi(\mathbf{c}), \varphi(\mathbf{c}, \mathbf{f}(\mathbf{c}), \eta(\mathbf{c}, \mathbf{f}(\mathbf{c}), \mathbf{f}(\mathbf{f}(\mathbf{c}))))\}$

...

All of these models are in fact equivalent to each other. We will argue this claim for the first two in the sequence but the same argument can be extended to the rest of the possible Herbrand Models. We consider the sets $\{\phi(\mathbf{c}), \varphi(\mathbf{c}, \mathbf{c}), \eta(\mathbf{c}, \mathbf{c}, \mathbf{f}(\mathbf{c}))\}$ and $\{\phi(\mathbf{c}), \varphi(\mathbf{c}, \mathbf{f}(\mathbf{c}), \eta(\mathbf{c}, \mathbf{f}(\mathbf{c}), \mathbf{f}(\mathbf{f}(\mathbf{c}))))\}$.

The grounded function terms within the predicates of one set are assignments for the variables which were arguments of the predicates. As we discussed in Section 3.3, the variables are assigned to individuals from the domain of the model. The individuals are represented with a "symbolic name", which is not a rigid designator. The grounded function terms and the constants $\mathbf{c}$ are representations for the individuals from the domain of the model. They are a different name for the same individual. The "symbolic name" bares no relevance on the meaning of the model – two models over one formula are different if they contain different grounded predicates (if different predicates are evaluated to true), but they will be the same if they assign different "symbolic names" to assign to the same variable and if they have the same grounded predicates (the same predicates are evaluated to true). In our case, the two models offer different assignments for the same variables (in the first model $\mathbf{y}$ is assigned the "symbolic name" $\mathbf{c}$ and in the second $\mathbf{y}$ is assigned the "symbolic name" f($\mathbf{c}$)), but they have the same grounded predicates and hence they are the same.

The Herbrand Models of a program P_A (built by translating the Formula 5.2 to logic rules) and the Herbrand Models of the Formula Formula 5.2 are the same. Consequently, in order to determine the answer set to P_A there is no need to ground the P_A with the entire Herbrand Universe of P_A. Grounding it with terms up to depth 1 is sufficient. (We consider f(x) a term with depth 1, f(f(x)) a term with depth 2 and so on.)

This means that answer sets can be derived for our infinite program P_A derived from an answer of the ECI Lexicon. The entire (infinite) set of logic rules will not be considered to calculate the answer sets. The infinite set will be split to a finite part and an infinite part. In our case it is enough to put rules which contain grounded terms up to depth 1 in the finite part. The answer sets of the finite part will also be answer sets of the entire program. The idea

of splinting the program in the context of answer set semantics is discussed in [46].

The class of infinite normal logic programs for which answer sets can be efficiently computed are called *finitary programs* [12]. The logic programs we obtain for the formulas of the ECI lexicon as we showed allow for efficient computing of answer sets. They also are normal (not contain disjunction in the head of the rule).

The first-order formulas of sentences from the ECI lexicon do not contain negation and disjunction explicitly. They are of form conjunction of positive atoms implies a conjunction of positive atoms. For example consider the simple formula of two conjuncts on each side of the implication(α and β are positive literals on the left and γ and δ are the positive literals on the right):

$$\alpha \wedge \beta \rightarrow \gamma \wedge \delta$$

This will be translated to the following set of normal logic rules:

$$\gamma :- \neg\alpha.\ \ \gamma :- \neg\beta.\ \ \delta :- \neg\alpha.\ \ \delta :- \neg\beta.$$

Next we briefly present the ASP solver which we chose to derive the answer sets to our program $P_A \cup P_Q$.

The DLV-System

There are several efficient ASP solvers available today as off-the-shelf the shelf tools, among which The DLV System [44] and Smodels [51] are the today state-of-the-art. Both are built to handle disjunctive logic programs without function symbols. We chose to use the DLV System because there has been an expected release of an extension to the DLV System [19] which is able to handle programs with both disjunctive and finitray rules[1].

The syntax and the underlying semantics of the programs of the DLV System is the same as those described described for the ASP earlier in this section. In addition to constant symbols and variables, the DLV System also supports integer constants and arbitrary string constants. The DLV System also offers several built in comparison and arithmetic predicates. The details of the full syntax, as well as semantics of the DLV System can be found in [44].

We can use the DLV System on our programs $P_A \cup P_Q$ directly as we obtain them from the translation algorithms with one modification.

The DLV system enforces a *safety rule* for its programs which refers to the head of the rules, the weakly negated atoms *not b* and for the comparison predicates. Out of those, to our programs only the safety rule for the head of the rules applies because there will be no comparison predicates and weakly negated atoms in our $P_A \cup P_Q$. The safety rule for the heads of the rules states that if a variable X appears in the head of the rule, X must also appear in the body of that rule.

The rules in the program P_Q will always be safe. To ensure that the rule safety rule is obeyed in P_A we will extend the translating algorithm which creates P_A by adding the following procedure to the end of it.

[1]The prototype of the DLV-VI we obtained directly from the developers of the University of Calabria

```
for each unary literal α in P_A do P_A ∪ {dom(X):- α(X).}
for each unsafe rule r in P_A
for each unsafe variable X in the head of r
   add dom(X) to the body of r.
```

5.4 Generating Background Knowledge

The work we have done so far considered if a question is answered only based on the representation of the possible answer. However, reasoning in question answering is not at all possible without background knowledge (lexicl knowledge, ontological knowledge, world knowledge, situation knowledge). Even the most trivial example of a question-answer pair such as:
Who dances? represented with:
$\exists y(person(x) \wedge event(y) \wedge dance(y) \wedge agent(y, x))$
Marija dances. represented with:
$\exists x \exists y(marija(x) \wedge event(y) \wedge dance(y) \wedge agent(y, x))$
we would not be able to make a confirmation without including the background knowledge that if x is a *marija* is is also a *person*.

The problem of generating background knowledge for a question answering system is very difficult task and it exceeds the scope of this thesis. We will show only what we found to be the minimum requirements for a background knowledge for our university domain and indications on how to satisfy these requirements.

The first comment is that it is most advisable for the background knowledge to be written directly in the language of the system which does the automated reasoning for the question answering entailment. In our case our background knowledge should be written directly in DLV news.

In Section 3.2, we discussed that the predicate special predicate `eq` will denote the verb *is*. The knowledge base should be extended with additional rules so that the semantic of the predicate `eq` to be preserved. For each unary predicate $P(x)$ and binary predicate $Q(x, y)$ we will add the rules:

```
p(X):- p(Y),eq(X,Y).
q(X,_):-q(Y,_),eq(X,Y).
q(_,X):-q(_,Y),eq(X,Y).
eq(X,Y):-eq(Y,X).
```

Because we know before hand the continence of the answers that can be retrieved by the IQA system, we can build a small ontology for those answers to represent the membership to the domain of the questions for the concepts that can be found in the answers. Namely, to declare which concepts are persons, time units, locations and so on. To the best of our knowledge there is no automated way to do this concept domain knowledge extraction and it has to be done by hand. While this job is fairly easy for the domains such as *person* and *location*, it is more difficult to establish membership to the domain *reason*(corresponding to the question "why") and the membership to the domain *manner* would be harder to establish because it will extend over more then one concept.

This "is-a" ontology should also be extended to contain the domain specific knowledge about the is-a relations which hold among the individual concepts.

For example, in the FAQ Sheets in the library domain we have considered, we can often find rules talking about the *material* which we as humans can easily conclude that in this setting refers to the library material and it refers to books, DVD and so on. For the purpose of automated reasoning, the background knowledge should contain the information on exactly which concepts are "children" of the concept *material* in the library domain.

The natural language is rich and there are many words or phrases to express one and the same thing. If we want to be able to do question answering for questions and answers which contain the same information expressed in various linguistic ways we need basic lexical background knowledge. The relations in the lexical database WordNet [28] are widely used for the purpose of background knowledge in reasoning over natural language. WordNet is a very big database and we would not need to use all the concepts in it for our background knowledge for a restricted domain. For indications on how to extract only the information we need from WordNet we can use the the work presented in [15]. For using WordNet to extract ASP rulers over a given domain we can use the experience presented in [7].

These are only a few indications on what the background knowledge for the reasoning purpose for IQA over the university library domain should unavoidably contain. They are presented only to serve as a starting point to any future work on completing the Logic Support Unit for the IQA System.

Chapter 6

Conclusions and Future Work

In this thesis we have studied the problem of supporting probabilistic Interactive Question Answering, over a restricted domain, with logic reasoning. We concentrated on finding theoretic and applicable solutions to two particular problems: confirming (or refuting) that a probabilistically retrieved answer answers a given question and extracting the specific answer from a verified question. The solution to these two problems comprised of three tasks: finding a way to automatically and efficiently build wide-coverage semantic representations for natural language, ensuring that those representations allow for efficient logic reasoning to be performed on them and developing a formalism to represent the answer verification and specific answer extraction problem as a automated logic reasoning task. We state here how we handled these tasks and compare our approach to related work. We find that there are many fruitful research paths that remain unclosed by this thesis. At the end of this chapter we will outline some of the possibilities for future work in the course of the work of this thesis.

6.1 Overview of Results

The first task in this thesis was to choose a suitable semantic representation for the questions and answers. We decided to use first-order semantic representations. We obtain these representations by using a tool which efficiently builds wide-coverage semantic representations (BOXER) from derivations of a wide-coverage statistical parser (CCG Parser). The benefit of using a probabilistic approach to parsing is that the resolution of possible ambiguities (lexical and structural) in the parsed natural language sentence is solved – the parser is trained to provide only one, the most likely, derivation. The wide-coverage feature of the tools allows for semantic representations to be obtained from a wide range (versatile in context and expression level) of natural language.

The second task in this thesis was to ensure efficient reasoning over the semantic representations. To this end we defined lexicon restrictions for the natural language sentences comprising the answers and the language of the question which yield decidable (for satisfiability and entailment) first-order representations. Corpora analysis showed that the restricted lexicons have high

coverage. The joint answer lexicons cover 91% of the general answers over the library. The question lexicon cover also 91% of questions from large question corpora.

Establishing decidable fragments of natural language by means of lexicon restrictions can be seen in [49]. Unlike the restrictions proposed in [49], which restrict entire categories of words (ditransitive verbs, numbers, relative pronouns etc.), the lexicon restrictions we propose are easier to use because we restrict the usage of a short list of specific words on the level of a sentence, and no linguistic knowledge is needed to use the restricted language (one does not need to know what are ditransitive verbs, what are reflexive pronouns etc.).

The third task was to choose a logic formalism which is suitable to represent as an automated reasoning problem the problem of answer verification and specific answer extraction (IQA problems). We proposed a formalism of representing the IQA problems as a task of reasoning over answer sets. The basic idea behind our reasoning solution is the same as the ASP formalism for retrieving an answer to a question presented in [7].

In [7] the logic programs, which represent the natural language possible answer and question, are being built by an algorithm. This algorithm constructs the logic rules from derivations of the Link Grammar Parser (used to parse the answer and question). Our approach is to obtain the logic rules by translation of the first-order representations of the answer and question. By doing so, we are able to take advantages of the ambiguity resolution feature of the CCG Parser, as well as the wide-coverage feature of both BOXER and the CCG Parser.

We show how to translate into disjunctive logic rules the formulas of answers and questions built using the restricted lexicons we have earlier defined. We show how to verify an answer and extract the specific answer(s) by analyzing the answer sets of the obtained logic program. The answer sets of our logic programs can be calculate using the DLV System.

The logic program we build to represent the answer and question is small in size because we represent a possible answer and not a large knowledge base (which would correspond to a long text). For this reason, the calculation of the answer sets will be efficient in spite of the fact that the problem of calculating an answer set to a disjunctive logic program is Σ_π^2 hard [44].

6.2 Future Work

Discovering ways to improve the precision of statistical Question Answering by using KR and Reasoning techniques is a new field which is rapidly developing. The work of this master thesis aims not to offer final solutions for using KR&R techniques in the field of IQA, but to show that the possibility of combining the NLP methods with KR&R methods in the field IQA exists and deserves to be assigned further research attention. Consequently, many issues remain open for future work. We will outline the ones we consider to be the most promising.

Future Work in Fragments of Natural Language

Although the coverage of the Restricted Lexicons we define is high there are sentences (answers) and questions which are not covered.

The answers from the analyzed corpora which could not be covered by the Restricted Lexicons are those which contain both a negation and implication introducing lexical item. We have observed that in the case when the scope of the implication introducer and the negation introducer do not overlap in the sentence, the semantic representation of that sentence will be decidable. This claim remains to be formally proved. For these additional semantic representations, the algorithm for translating the first-order representations for answers can be used without modifications.

We observed that a question which contains one negation introducer and no disjunction introducer also yields decidable entailment with the existing answer lexicons. This claim also remains to be formally proved. The first-order formula of a question with one negation introducer can be represented with logic programing rules without modifications to the procedure.

Future Work in Reasoning for IQA

boxer normalizes cardinal and date expressions and represents them with special predicate symbols. It will be useful to represent integer cardinal expressions that appear in a possible answer as numbers over which arithmetic operations can be performed. Answers as for example "A user can borrow at most 40 books" can only be verified for a questions as *"Can I borrow 15 books?"* if there is a way to reason that 40 is greater then 15.

The DLV System (which we employ as an ASP solver) allows for the use of integers and basic arithmetic operations over integers. This possibility should be further researched and incorporated in the current answer verification and specific answer extraction ASP formalism by re-representing the normalized integer cardinal expressions with DLV integers.

Casting the problem of answer verification and specific answer extraction in terms of Answer Set Programing offers more reasoning possibilities then we use. In the work presented in this thesis we use the ASP to automatize deductive reasoning (classical reasoning). One of the features of the ASP framework, as framework for Declarative Problem Solving, is that it allows for simple implementation of non-monotonic reasoning, in particular default reasoning. We consider that this opportunity should be taken advantage of.

Non-monotonic reasoning is considered to be closer to human reasoning then classical reasoning [3], and as such useful for the purposes of NLP [17], [27], [48]. We give a few pointer on how non-monotonic reasoning can be used in our framework.

Default reasoning. We can often find information that typically holds in the answers from the library domain. For example the answer "*The loan period normally lasts 30 days with the exception of textbooks from Bozen/Bolzano marked with the number 15, journals and DVD which have a loan period of 14 days.*" . The syntax of ASP includes weak negation(negation as failure to derive a proof). We can make use of the weak negation to represent rules of the form "A normally holds with the exception B" as default rules of form "A if *not* B" where *not* is a weak negation. The problem which has to be solved before being able to use the default reasoning is how to automatically obtain semantic representations for sentences which contain default information.

Abductive reasoning. It is fairly simple to extend the Background knowledge with information about membership to the domains corresponding to the

questions *where*, *who*, *when*, and *what* by simply declaring which noun phrases are persons, locations, time units or things. What falls in the domain of the questions *how* and *why* is much harder to be determined because in this case the domain is not a single noun phrase, but consists of an entire phrase. In this case, reasoning over one possible answer may not longer be sufficient.

It has been argued that abductive reasoning is what needs to be employed for answering "why" questions [17]. Abductive reasoning can be represented in many different ways, however the most simple one is by using Abductive Logic Programming (ALP). ALP as a framework of Declarative Problem Solving is being rapidly developed to extend logic programs to perform abductive reasoning [38]. We believe that this framework should be explored for the needs of Question Answering.

Bibliography

[1] K. Ajdukiewicz. *Die syntaktische Konnexitaet.* Oxford University Press, 1935.

[2] Hajnal Andreka, Istvan Nemeti, and Johan van Benthem. Modal languages and bounded fragments of predicate logic. *Journal of Philosophical Logic*, 27(3):217–274, 1999.

[3] Grigoris Antoniou. *Nonmonotonic Reasoning.* The MIT Press, 1997.

[4] Kathryn L. Baker, Alexander M. Franz, and Pamela W. Jordan. Coping with ambiguity in knowledge-based natural language analysis. In *In Proceedings of the 15th International Conference on Computational Linguistics COLING-94*, 1994.

[5] Y. Bar-Hillel. *A quasi-arithmetical notation for syntactic description.* Language 29, 1953.

[6] Chitta Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving.* Cambridge University Press, 2003.

[7] Chitta Baral and Luis Tari. Using ansprolog with link grammar and wordnet for qa with deep reasoning. *ICIT*, pages 125–128, 2006.

[8] H.P. Barendregt. *The Lambda Calculus (Studies in Logic and the Foundations of Mathematics)*, volume 103. North Holland, 1984.

[9] Farah Benamara. Language and reasoning for question answering:state of the art and future directions. In *Proceedings of the workshop KRAQ06:Knowledge and Reasoning for Language Processing*, Trento,Italy, 2006.

[10] Raffaella Bernardi, Francesca Bonin, Diego Calvanese, Domenico Carbotta, and Camilo Thorne. English querying over ontologies: E-quonto. In *The 10th Congress of the Italian Association for Artificial Intelligence (AIIA 2007)*, Roma, Italy.

[11] Patrick Blackburn and Johan Bos. *Working With Discourse Representation Theory.* http://www.cogsci.ed.ac.uk/∼jbos/comsem/book2.html, 2006.

[12] Piero A. Bonatti. Reasoning with infinite stable models. *Artificial Intelligence*, 156(1):75–111, 2004.

[13] Egon Borger, Erich Gradel, and Yuri Gurevich. *The Classical Decision Problem*. Springer, 1996.

[14] Johan Bos. Implementing the binding and accommodation theory for anaphora resolution and presupposition projection. *Computational Linguistic*, 29(2):179–210, 2003.

[15] Johan Bos. Towards wide-coverage semantic interpretation. In *In Proceedings of Sixth International Workshop on Computational Semantics IWCS-06*, pages 42–53, 2005.

[16] Johan Bos and Malte Gabsdil. First-order inference and the interpretation of questions and answers. 1999.

[17] D. Burhans and S. Shapiro. Abduction and question answering. In *In Proceedings of the IJCAI-01 Workshop on Abductive Reasoning, AAAI*, Seattle,WA, 2001.

[18] Ricardo Caferra, Alexander Leitsch, and Nicolas Peltier. *Automated Model Building*. Springer, 2004.

[19] Francesco Calimeri, Susanna Cozza, and Giovambattista Ianni. External sources of knowledge and value invention in logic programming. *Annals of Mathematics and Artificial Intelligence*, 50(3-4):333–361, 2007.

[20] Diego Calvanese, Giuseppe Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable reasoning and efficient query answering in description logics: The dl-lite family. *Journal of Automated Reasoning*, 39(3):385–429, 2007.

[21] E. Charniak. A maximum-entropy-inspired parser. In *In Proceedings of the FirstMeeting of the North American Chapter of the Association for Computational Linguistics*, pages 132–139, Seattle, WA., 2000.

[22] Stephen Clark and James R. Curran. Wide-coverage efficient statistical parsing with ccg and log-linear models. *Computational Linguistics*, 33(4), 2007.

[23] M. Collins. *Head-Driven Statistical Models for Natural Language Parsing*. PhD thesis, Computer and Information Science, University of Pennsylvania, 1999.

[24] James R. Curran, Stephen Clark, and Johan Bos. Linguistically motivated large-scale NLP with C&C and Boxer. In *Proc. of the Demonstrations Session of the 45th Annual Meeting of the Association for Computational Linguistics (ACL 2007)*, Prague, Czech Republic, 2007.

[25] H. B. Curry and R. Feys. *Combinatory Logic,*, volume I. North-Holland, Amsterdam, 1958.

[26] Rob A. Van der Sandt. Presupposition projection as anaphora resolution. *J Semantics*, 9(4):333–377, 1992.

[27] Harry C. Bunt (editor) and William J. Black (Editor). *Abduction, Belief and Context in Dialogue: Studies in Computational Pragmatics*. John Benjamins Publishing Co, 2000.

[28] Christiane Fellbaum. Towards a representation of idioms in WordNet. In Sanda Harabagiu, editor, *Use of WordNet in Natural Language Processing Systems: Proceedings of the Conference*, pages 52–57. Association for Computational Linguistics, Somerset, New Jersey, 1998.

[29] Norbert E. Fuchs and Uta Schwertel. Reasoning in attempto controlled english. *Principles and Practice of Semantic Web Reasoning*, 2901:174–188, 2003.

[30] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In Robert A. Kowalski and Kenneth Bowen, editors, *Proceedings of the Fifth International Conference on Logic Programming*, pages 1070–1080, Cambridge, Massachusetts, 1988. The MIT Press.

[31] Erich Grädel. On the restraining power of guards. *Journal of Symbolic Logic*, 64(4):1719–1742, 1999.

[32] Jeroen Groenendijk. The logic of interrogation: Classical version. In *In Proceedings of the Ninth Semantics and Linguistics Theory Conference (SALT IX)*, Ithaca, NY, 1999.

[33] Julia Hockenmaier. *Data and models for statistical parsing with Combinatory Categorial Grammar*. PhD thesis, School of Informatics, The University of Edinburgh, 2003.

[34] Julia Hockenmeir and Mark Steedman. Generative models for statistical parsing with combinatory categorial grammar. In *In Proceedings of the 40th Meeting of the ACL*, pages 335–342, Philadelphia,PA, 2002.

[35] Wilfrid Hodges. *Model Theory*. Cambridge University Press, 1993.

[36] Ian Hodkinson. Loosely guarded fragment of first-order logic has the finite model property. *Manuscript*.

[37] Ian Horrocks, Oliver Kutz, and Ulrike Sattler. The even more irresistible sroiq. In *KR*, pages 57–67. AAAI Press, 2006.

[38] Antonis C. Kakas, Robert A. Kowalski, and Francesca Toni. Abductive logic programming. *Journal of Logic and Computation*, 2(6):719–770, 1992.

[39] H. Kamp and U. Reyle. *From Discourse to Logic: Introduction to Model-theoretic Semantics, Logic and Discourse Representation Theory*. Kluwer Academic Publishers, 1993.

[40] Manuel Kirschner. The bob iqa system: a domain experts perspective. In *In Proc. of the 11th Workshop on the Semantics and Pragmatics of Dialogue (SemDial07)*, Rovereto, Italy. to be published.

[41] Manuel Kirschner. Building a multi-lingual interactive questionanswering system for the library domain. In *In Proc. of the 10th Workshop on the Semantics and Pragmatics of Dialogue (SemDial06)*, Potsdam, Germany.

[42] Tobias Kuhn. Acerules: Executing rules in controlled natural language. In *Proceedings of the First International Conference on Web Reasoning and Rule Systems (RR2007)*. Springer, 2007.

[43] Alexander Leitsch. *The Resolution Calculus*. Springer, 1998.

[44] Nicola Leone, Gerald Pfeifer, Wolfgang Faber, Thomas Eiter, Georg Gottlob, Simona Perri, and Francesco Scarcello. The DLV system for knowledge representation and reasoning. *ACM Trans. Comput. Logic*, 7(3):499–562, 2006.

[45] David Lewis. *Papers in Philosophical Logic: Relevant Implication*. Cambridge University Press, 1998.

[46] Vladimir Lifschitz and Hudson Turner. Splitting a logic program. In *International Conference on Logic Programming*, pages 23–37, 1994.

[47] K. F. McCoy and J. Cheng. Focus of attention: Constraining what can be said next. In *Natural Language Generation in Artificial Intelligence and Computational Linguistics*, pages 103–124, Boston, 1991. Kluwer.

[48] F. Nouioua and P. Nicolas. Using answer set programming in an inference-based approach to natural language semantics. *ArXiv Computer Science e-prints*, 2006.

[49] Ian Pratt-Hartmann and Allan Third. More fragments of language: the case of ditransitive verbs'. *Notre Dame Journal of Formal Logic, 47(2)*, pages 151–177, 2006.

[50] Mark Steedman and Jason Baldridge. Combinatory categorial grammar. *Non-Transformational Syntax*, 2007.

[51] Tommi Syrjänen and Ilkka Niemelä. The smodels system. *Logic Programming and Nonmonotonic Reasoning: Proceedings of the 6th International Conference, LPNMR 2001.*