

# Logic Reasoning in Question Answering

## Using Answer Set Programming to support Question Answering in a Restricted Domain Interactive setting

Marija Slavkovic

Computer Science and Communication Research Unit, University of Luxembourg

### Introduction

Interactive Question Answering over a Restricted Domain (IQA-RD) is a type of information retrieval system whose task is, with the participation of the user, to retrieve exactly one specific answer to the question of the user posed in natural language. Restricted domains are characterized by small number of documents (from which the answer is retrieved) with good quality, i.e. containing minimal redundant information. To retrieve the required specific answer, IQA-RD can not only rely on keyword-based techniques to locate several "good enough" passages from the document collection (a.k.a. shallow QA), but must also employ deep analysis techniques: named-entity recognition, relation detection, word sense disambiguation, logic reasoning etc.

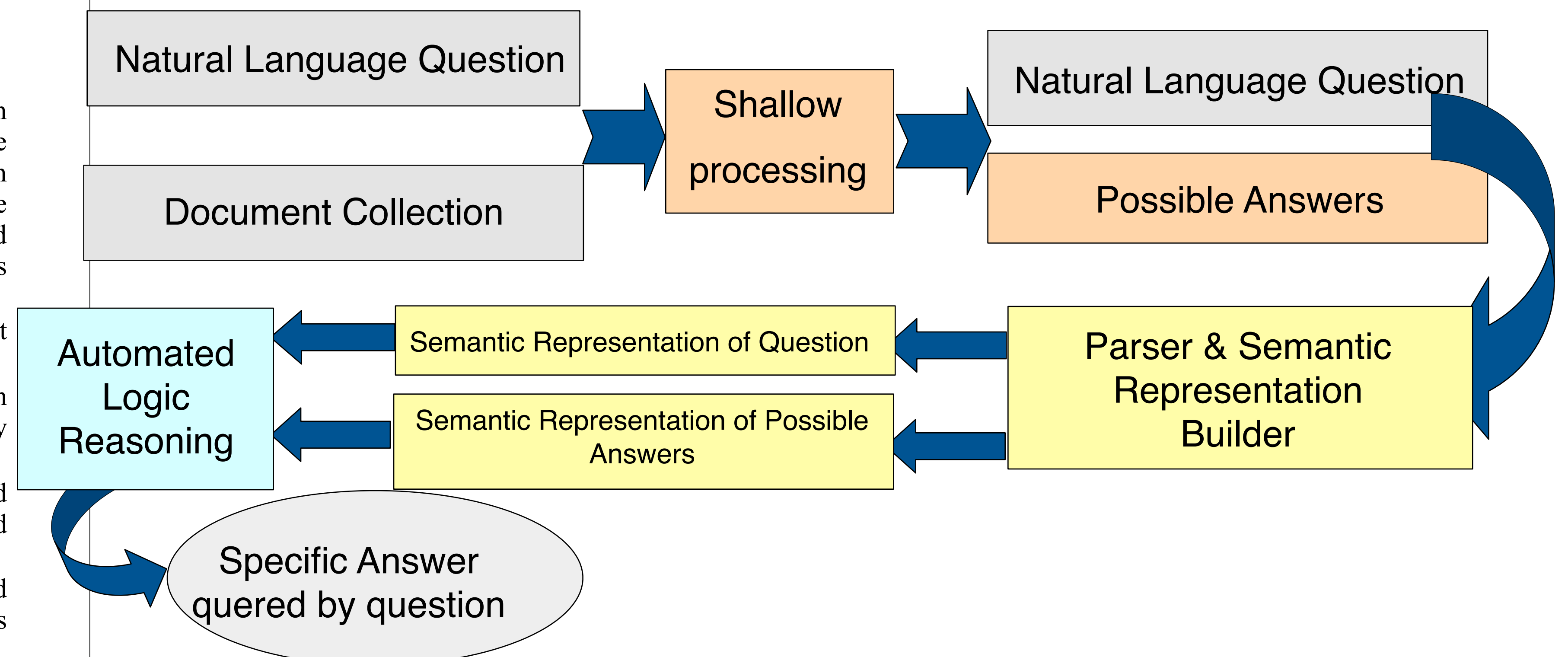
Shallow QA techniques are scalable and perform well for narrowing down possible answers from a document collection, however they can not guarantee precision and correctness in retrieved information.

Logic reasoning techniques work over formal languages and are exact, i.e. able to prove if information retrieved is the information queried. However, logic reasoning is computationally too demanding and is only employable over very small quantity of data.

IQA-RD demands the best of both worlds. With both the fields of computational linguistic and automated reasoning being mature enough [1] to offer off-the-shelf efficient tools, we show how these can be combined to comply with some of the requirements of a specific IQA-RD system.

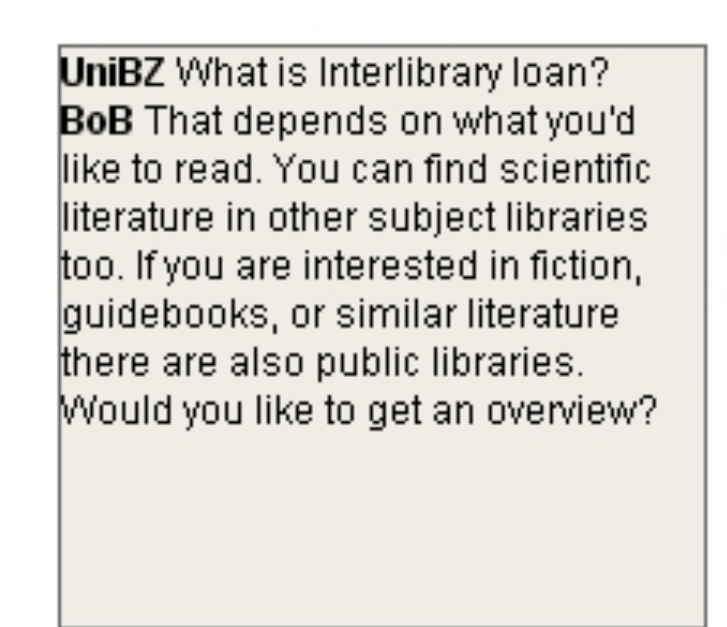
Answer Set Programming (ASP) is a form of declarative logic programming. ASP is intuitive to use and implements both non-monotonic and deductive reasoning. It offers basic reasoning over integers, as well as other extensions which can be useful for the reasoning needs of a IQA-RD system.

By selecting a specific IQA-RD we show, by using state of the art off-the-shelf tools, how to determine the correct answer from the possible answers that the IQA-RD retrieves, as well as to extract from it the specific information queried.



### Setting

Bolzano University chatter bot Bob [2] is a specific IQA-RD that answers users questions about the university library.



Language: en de it

All the answers that BoB can provide are predicted and stored in a tree structure (focus tree) according to topic, with related topics shearing a mother node. BoB uses regular expressions to analyze the question. It then matches the regular expressions to determine the topic and the focus of the dialog at hand. BoB then smart search the focus tree starting from last topic of user utterance and retrieves the answer it finds to belong to the right topic and regular expression pattern. BoB has no way of "knowing" if what it retrieves is really an answer to the users question. For example, BoB would return the same answer for both following questions:

Q1: How can I print the search results in the library?

Q2: How can the printed material of the library be searched?

Answer: With the symbol „print“ on the right side you can print the result list.

The precision of the retrieved information depends on how precisely the answers were predicted and placed in the focus tree. However a too precise focus tree is not robust. The fine tuning of a focus tree is a time consuming process.

To increase the reliability and precision we here propose that BoB is to be augmented with a Logic Support Unit (LSU). The architecture (and intended functionality) of this unit is given on Figure 1. This work deals only with the Representation and Reasoning sub-units.

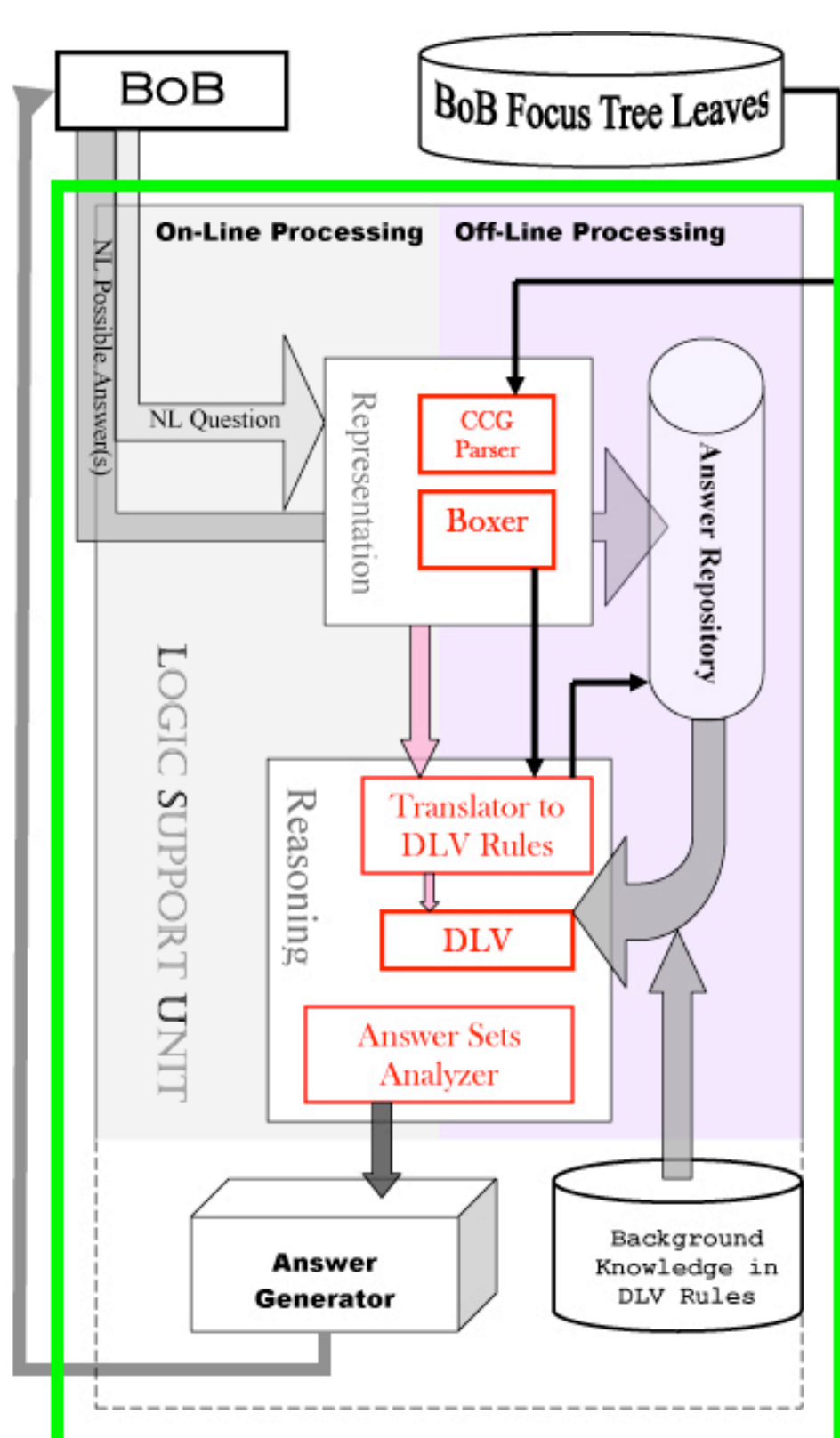


Figure 1. After receiving the question from the user, BoB retrieves a set of possible answers from the focus tree and sends them together with the question to the LSU. The LSU builds semantic representations from them. The semantic representations of the answers are pre-build and stored in the Answer Repository. The LSU transforms the semantic representations into corresponding logic program in the DLV\* syntax (DLV is a state of art ASP solver) which are then sent to the DLV system together with domain background knowledge. The LSU analyses the answer sets obtained to verify the correct answer and extract the required specific information from it. The specific answer is then used to generate a natural language answer which is returned to BoB. BoB has to only present this answer to the user.

### Obtaining Semantic Representations

To obtain semantic representation of the user question, as well as of the possible answers, we employ the CCG Parser and the semantic builder BOXER provided by C&C Tools\*. BOXER builds First Order Logic (FOL) formulae from the derivations of the CCG Parser.

The advantage of using these lies in the following. The CCG Parser is probabilistic and builds only one most likely derivation thus resolving ambiguity, name entity recognition, word sense disambiguation etc. Both Boxer and the CCG Parser are wide – coverage tools which are highly efficient – they process up to 35 newspaper sentences per second. BOXER FOL is not able to represent all linguistic phenomena. Plural, tense, modal verbs are not captured, while cardinal expression are normalized.

\* <http://svn.ask.it.usyd.edu.au/gandc/wiki>

### From BOXER FOL to Reasoning

FOL is computationally very expressive. The problem of deriving answer sets from programs semantically equivalent to full FOL formulae is undecidable. (Answer sets are the stable models of the logic program and for a general FOL formula there are infinitely many models) . To ensure that the answer sets can be derived only a fragment of natural language can be handled by the LSU.

Boxer builds semantic representations compositionally using  $\lambda$ -calculus:

1. assigns ( $\lambda$ ) semantic representations to the lexical items;
2. uses CCG rules in terms of functional application;
3. applying  $\beta$ -reduction to the resulting tree structure.

To determine the natural language fragments that can be handled, as well its computational properties we use the following observation. In the BOXER formulae, a finite set of lexical items (N/I/D Introdurers) introduce the operators  $\neg$ ,  $\forall$ ,  $\rightarrow$ , and  $\vee$ ; while the rest of the items introduce only  $\exists$  and  $\wedge$ . By restricting the appearance of the N/I/D Introdurers in the NL sentences, we are able to obtain only formulae with desirable properties.

#### Implication (and $\forall$ ) Introdurers:

all, any, anybody, anyone, anything, anywhere, each, either, every, everybody, everyone, everything, everywhere, few, if.

#### Disjunction Introdurer: or

#### Negation Introdurers:

another, instead of, neither, nobody, none, noone, no-one, not, nothing, other, previous, no, neither, nowhere.

### Fragments of Natural Language handled by LSU

By restricting the appearance of the N/I/D Introdurers in the NL sentences, we are able to obtain only formulae with desirable properties.

#### Answers

ECND Lexicon - one Negation Introdurer per sentence and optionally many Disjunction Introdurers

Allowed: Items marked with 14 or 15 can not be borrowed.  
Forbidden: All items not marked with 17 or 18 can not be borrowed.

ECI Lexicon - One Implication Introdurer (all, any, if, every ...) in a sentence

Allowed: All library users can use the Interlibrary Loan service.  
Forbidden: If you do not have a student card you can buy a library card.

	Answers Tot: 31	Sentences Tot: 68
EC Lexicon	7	13
ECD Lexicon	7	9
ECN Lexicon	3	4
ECND Lexicon	2	2
ECI Lexicon	4	5
Not Covered	6	5

To get an insight what are the coverage powers of the restricted lexicons we analyzed the FAQ sheets of library. The analysis showed that 86% of the FAQ written from the domain experts without instructions belong to the restricted lexicons. Further analysis showed that with rewriting some of the non covered sentences, 91% of the FAQ belongs in the restricted lexicons.

#### Questions

The Questions are built similarly as answers. The general formula of the question is  $D(x) \wedge \exists y \varphi(x,y)$ , where  $D(x)$  is the domain of the question, which is determined according to question word.

who  $\rightarrow$  person, where  $\rightarrow$  location, when  $\rightarrow$  unit\_of\_time, which  $K \rightarrow K$ , what  $\rightarrow$  thing, how  $\rightarrow$  manner, why  $\rightarrow$  reason

Yes/No questions do not have a domain.

The variable  $x$  marks the specific answer queried. The formula  $\exists y \varphi(x,y)$  represents the body of the question, namely all the necessary conditions which have to hold for  $x$  to be an answer to the posed question.

QECD Lexicon – only optionally many Disjunction Introdurers

Allowed: How can I order or borrow books?  
Forbidden: Who can not borrow books?

	Clinical Questions Tot: 435	Answer.com Tot: 444	TREC Tot: 408
universal	12	6	2
existential	111	22	1
disjunction	132	15	2
negation	52	13	1

To get an insight on how many of the users questions the LSU would be able to handle by working only over the QECD lexicon, we used a Question Corpora Analysis (presented in [8]). According to this analysis 91% in these corpora are questions belonging to the QECD.

### Obtaining and Using the Answer Sets

The translation from FOL to logic programs, as well as the analysis of the answer sets for these logic programs are shown through examples.

Every curious person borrowed books. Marija is curious. Manuel did not borrow books.

Who borrowed the books? Are books borrowed?

$\forall x1(\text{person}(x1) \wedge \text{curious}(x1)) \rightarrow \exists x2 \exists x3(\text{borrow}(x2) \wedge \text{event}(x2) \wedge \text{book}(x3) \wedge \text{agent}(x2,x1) \wedge \text{patient}(x2,x3))$

$\exists x1 \exists x2 \text{maria}(x1) \wedge \text{event}(x2) \wedge \text{curious}(x1)$

$\exists x1(\text{manuel}(x1) \wedge \neg (\exists x2 \exists x3(\text{book}(x3) \wedge \text{borrow}(x2) \wedge \text{event}(x2) \wedge \text{agent}(x2,x1) \wedge \text{patient}(x2,x3)))$

$\text{person}(x1) \wedge (\exists x2 \exists x3(\text{book}(x3) \wedge \text{borrow}(x2) \wedge \text{event}(x2) \wedge \text{agent}(x2,x1) \wedge \text{patient}(x2,x3))$

$\exists x1 \exists x2(\text{book}(x1) \wedge \text{borrow}(x2) \wedge \text{event}(x2) \wedge \text{patient}(x2,x1))$

$(\text{borrow}(Y1)) \wedge \text{person}(Y1) \wedge \text{curious}(Y1) \wedge \text{maria}(a1)$

$\text{event}(Y1) \wedge \text{person}(Y1) \wedge \text{curious}(Y1) \wedge \text{event}(a2)$

$\text{book}(g(Y1)) \wedge \text{person}(Y1) \wedge \text{curious}(Y1) \wedge \text{curious}(a1)$

$\text{agent}(Y1, Y1) \wedge \text{person}(Y1) \wedge \text{curious}(Y1) \wedge \text{manuel}(a3)$

$\text{patient}(x1, g(x1)) \wedge \text{person}(Y1) \wedge \text{curious}(Y1)$

$\neg \text{book}(Y3) \wedge \text{borrow}(Y2) \wedge \text{event}(Y2) \wedge \text{agent}(Y2,a3) \wedge \text{patient}(Y2,Y3) \wedge \text{manuel}(a3)$

$\text{query\_answer}(x) \wedge \text{person}(x) \wedge \text{book}(Y3) \wedge \text{borrow}(Y2) \wedge \text{event}(Y2) \wedge \text{agent}(Y1, X) \wedge \text{patient}(Y2, Y3)$

$\text{query\_answer}(yes) \wedge \text{book}(Y1) \wedge \text{borrow}(Y2) \wedge \text{event}(Y2) \wedge \text{patient}(Y2, Y1)$

$\text{query\_answer}(no\_1) \wedge \neg \text{book}(Y1)$

$\text{query\_answer}(no\_2) \wedge \neg \text{borrow}(Y2)$

$\text{query\_answer}(no\_3) \wedge \neg \text{event}(Y2)$

$\text{query\_answer}(no\_4) \wedge \neg \text{patient}(Y2, Y1) \wedge \text{book}(Y1) \wedge \text{borrow}(Y2)$

$\text{person}(a1) \wedge \text{person}(a3)$

#### Analyzing the Answer Sets

Given a logic program  $P_q$  corresponding to a possible answer  $P_{ab}$  corresponding to background knowledge and a program  $P_q$  (or  $P_{q'}$ ) corresponding to a question (or a yes/no question) respectively we consider that:

An answer is verified if the special predicate `query_answer` is in the answer sets of  $P_q \cup P_{ab} \cup P_{q'}$

The specific answer asked by the question is the constant in `query_answer` + the predicates followed by it

The answer set for  $P_q \cup P_{ab} \cup P_{q'}$

```
{maria(a1),curious(a1),event(a2),manuel(a3),person(a1),person(a3),borrow((a1)),event((a1)),book(g(a1)),agent((a1),a1),patient((a1),g(a1)),query_answer(a1)}
```

The answer set for  $P_q \cup P_{ab} \cup P_{q_{no}}$

```
{maria(a1),curious(a1),event(a2),manuel(a3),person(a1),person(a3),borrow((a1)),event((a1)),book(g(a1)),agent((a1),a1),patient((a1),g(a1)),query_answer(yes)}
```

### Conclusions & Future Work

By only using off-the-shelf tools from computational linguistic (the C&C Tools) and automated reasoning (DLV System) we showed how the basic reasoning demands of an IQA-RD system can be met. We find that the ASP can offer much more for QA.

In this work we use only deductive reasoning, which is the automated reasoning used in most contemporary QA systems [3],[4]. ASP as a reasoning framework can implement both non-monotonic and deductive reasoning. Non-monotonic reasoning is considered [5], [6] to be more adequate for reasoning over natural language. By using ASP for reasoning, the non-monotonic reasoning is also made available for the purposes of QA. To this end, the semantic representation builder should be modified to represent lexical items such as "normally", "usually" etc.

One interesting extension of the ASP is its built in integer comparison predicates. This can be used in QA for reasoning over quantities. BOXER normalizes cardinal and date expressions and represents them with special predicate symbols. To access the reasoning over integers with ASP, BOXER should be modified to not do this normalization, but to represent the cardinal and date expressions as integers.

Lastly, it should be noted that the grammar of the English language allows for more sentences to be constructed than what is common to be used among human users. Additional corpus analysis is needed to determine which grammar restrictions are "natural" to occur in the sentences containing intoducer lexicon items and what are the ASP properties of such sentences.

### Literature cited

- [1] Johan Bos. Three Stories on Automated Reasoning for Natural Language Understanding. Proceedings of ESCoR (IJCAR Workshop): Empirically Successful Computerized Reasoning. Pages 81-91, 2006
- [2] Manuel Kirschner. Building a multi-lingual interactive question answering system for the library domain. In Proc. of the 10th Workshop on the Semantics and Pragmatics of Dialogue (SemDial06), Potsdam, Germany
- [3] Norbert E. Fuchs and Uta Schwertel. Reasoning in attempto controlled english. Principles and Practice of Semantic Web Reasoning, 2901:174-188, 2003.
- [4] Johan Bos and Michael Kohlhase : ICoS-2, Workshop Proceedings. Inference in Computational Semantics. International Conference and Research Center for Computer Science, Schloss Dagstuhl. July 29-30, 2000.
- [5] D. Burhans and S. Shapiro. Abduction and question answering. In Proceedings of the IJCAI-01 Workshop on Abductive Reasoning, AAAI, Seattle, WA, 2001.
- [6] Uri Zernik and Allen Brown. 'Default reasoning in natural language processing', Proceedings of the 12th conference on Computational linguistics, 801-805, (1988).
- [7] Chitta Baral and Luis Tari. Using ansprolog with link grammar and wordnetfor QA with deep reasoning. ICTI, pages 125-128, 2006.
- [8] Raffaella Bernardi, Francesca Bonin, Diego Calvanese, Domenico Carbotta, and Camilo Thorne. English querying over ontologies: E-quoito. In The 10th Congress of the Italian Association for Artificial Intelligence (AIIA 2007), Roma, Italy.

### For further information

The master thesis presented in this poster was conducted at the KRDB Center of the Free University of Bozen-Bolzano, under the supervision of Dr. Raffaella Bernardi. Please contact [marija.slavkovic@uni.lu](mailto:marija.slavkovic@uni.lu) for a copy of the masters thesis.

Some starting points and further information on ASP can be obtained from:

<http://www.cs.uni-potsdam.de/~torsten/asp/>

\* <http://www.dbai.tuwien.ac.at/proj/dlv/>