Using a Logic Programming framework to Control Database Query Dialogues in Natural Language

*LuisQuintano**, *IreneRodrigues*†

ljcq@sc.uevora.pt ipr@di.uevora.pt
* Serviço de Computação † Departamento de Informática
Universidade de Évora, Portugal

Abstract. We present a natural language question/answering system to interface the University of Évora databases that uses clarification dialogs in order to clarify user questions. It was developed in an integrated logic programming framework, based on constraint logic programming using the GnuProlog(-cx) language [2, 11] and the ISCO framework [1]. The use of this LP framework allows the integration of Prolog-like inference mechanisms with classes and inheritance, constraint solving algorithms and provides the connection with relational databases, such as PostgreSQL. This system focus on the questions' pragmatic analysis, to handle ambiguity, and on an efficient dialogue mechanism, which is able to place relevant questions to clarify the user intentions in a straightforward manner. Proper Nouns resolution and the pp-attachment problem are also handled.
This paper briefly presents this innovative system focusing on its ability to correctly determine the user intention through its dialogue capability.

Keywords: Natural Language, Logic Programming, Information Systems, Dialog Management, Databases

1 Overview

IIS-UE (Universidade de Évora Integrated Information System) gathers all kinds of information, relevant for students (enrolled courses, grades, class summaries, etc.), for teachers (courses information, projects, students evaluation, personal data, etc.) and staff (data management, statistics, personal data, etc.). Several applications were built around IIS-UE to "deliver" information to the school community, but sometimes that's not enough. To use these applications one must know how they work. A student may know what information he wants but he doesn't know how to get it from the existent applications.

To solve these problems a natural language querying application (NL-Ue) was developed over IIS-UE [1]. Its practical aim is to give to our school community an easy way for retrieving stored information [3] [13].

[1] The system is built for the Portuguese language only interpreting qa and wh-questions

The information is stored in Postgresql relational databases [10]. To access this data, NL must map the user question to a database query language so that the resulting pragmatic interpretations can be evaluated in the databases.

NL-Ue implementation is based on constraint logic programming using the GnuProlog(-cx) language [11] and the ISCO framework [1].

The system architecture is simple and module-based. The system has five distinct modules which are connected through well defined API's [Fig. 1]. A more detailed description of the system can be found in [18] [17].

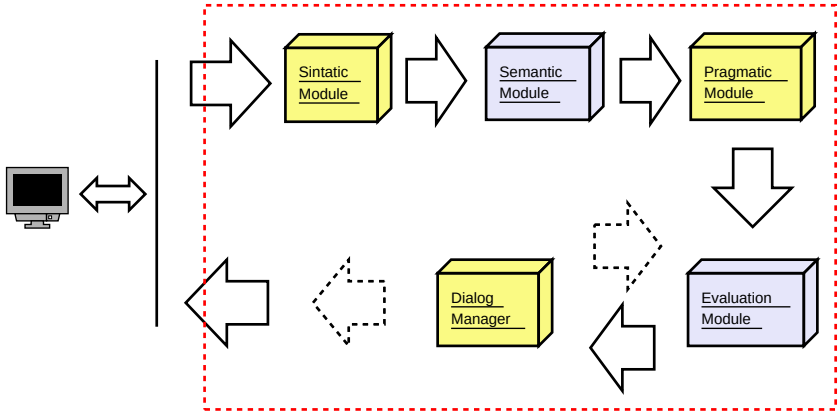

Fig. 1. NL-Ue Architecture

In this paper we discuss how pragmatic interpretation is handled by NL-Ue, but the focus will be on its dialog capabilities and how this system is able to clarify the user intentions with effective and precise questions.

Next (section 2) some related work is presented to enhance this application relevance on the dialogue and natural language development context.

In section 3 the pragmatic interpreter is described along with the strategy that was used to generate the pragmatic evaluation rules (which control the flow and the behaviour of the pragmatic interpreter).

Then (section 4) the dialogue mechanism is shown recurring to practical examples and finally some conclusions and future work are presented (section 6).

2 Related work

Some systems can be found along the last years that touch the problem of relational database querying and clarification dialog. Most of them, as the Precise system, directly generate SQL queries to access relational databases.

The Precise System by Etzioni, Kautz and Popescu [20] maps simple english natural language sentences to SQL with graph analysis techniques. Tokens (manually defined - low portability) represent directly attributes of database elements and relations between them. Precise uses a tokenizer to identify all possible complete tokenizations of the question, converting them to SQL with a simple syntactic parser. The system is limited to a restricted set of types of questions. In turn, other systems as Androutsopoulos' Masque/SQL uses an intermediate representation before generating the SQL. This approach is similar to our own, although different in the followed methodology. Androutsopoulos' Masque/SQL [5] [6] system is based on the previous Masque [4] implementation, which in turn was based on Fernando Pereira's CHAT-80 [22].

While Masque maps an english natural language question into Prolog to query a declarative database, Masque/SQL extends it by interfacing directly with relational databases. To do that, Masque/SQL needs some meta-data which has to be reconfigured each time there is a change of working domain. To manage this meta-data Masque/SQL has a domain editor where: (1) the domain entities (represented on the databases) are described (2) the set of "expected words" to appear in the questions and it's logical meaning - Prolog predicates - are describes (3) the connection between each predicate and a database table, view or select is explicitly represented. Once again, portability is one of the main problems of this system.

Although benefiting from the RDBMS SQL optimization, Masque/SQL may sometimes generate redundant SQL queries, decreasing it's efficiency.

Other systems use external integration tools for accessing information repositories. One of them is Katz's START [16] which uses Omnibase [15], a system for heterogeneous database access which uses natural language annotations to describe the data and the kind of questions that can be answered. This system uses the "object-property-value" data model and only works for direct questions about object properties. These annotations are manually built which makes the system's portability difficult.

While the mentioned applications are simple question/answering systems that do not identify wrong interpretations, NL-Ue intends to go a step further adding a clarification dialog capability. Clarification dialogue theory was vastly analyzed by several authors[12] [8] and some works can be found where clarification is applied to open domain or more narrowed domain question answering systems [21].

NL-Ue can be seen as a question/answering dialogue system, identifying erroneous interpretations by means of a clarification dialogue with the user.

3 Pragmatic interpretation

Syntactic and semantic analysis of this system are generically treated by the VISL parser [7] (syntax) and by an internally built semantic parser to gener-

ate DRS structures [14] through first-order logic predicates. At this stage, after syntactic and semantic analysis, the resulting first-order logic predicate (LPO) representation (as seen in Fig. 2) will reach as an input for the pragmatic module. Contextual information is added at this stage.

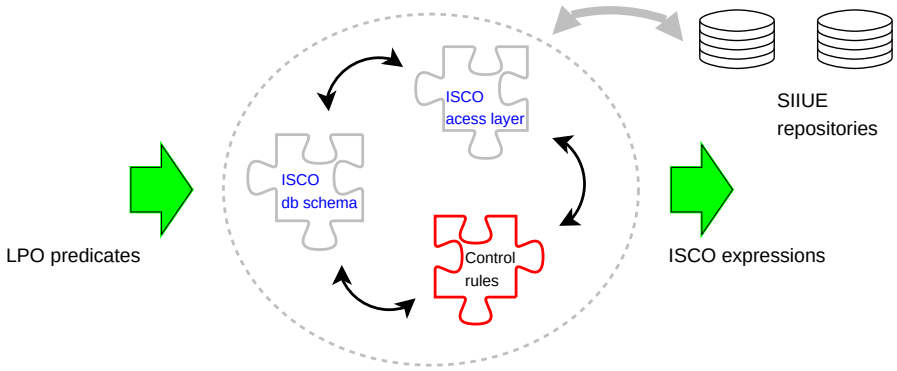

Fig. 2. Pragmatic Interpreter

NL-Ue application context is not only the IIS-UE data but also its structure. This information is added to the interpretation mechanism through a set of pragmatic control rules. To generate these rules, NL-Ue uses ISCO, a logical tool and development framework that enables relational database schema representation and full access to the stored data. This generation is automatic and must be done previously so that the rules are available at interpretation time (runtime).

3.1 Database representation/access framework

ISCO [1] is a logic-based development framework with its roots in the GnuProlog language [11] and is being developed in Universidade de Évora Computer Engineering department. Its use in NL-Ue's pragmatic interpretation adds the system the ability to internally represent and access the IIS-UE relational databases in a logic-based environment.

Figs. 3 and 4 show a fragment of one of IIS-UE's relational databases in entity/relation and SQL representation. It presents the action of "teaching" (leciona) which relates a teacher (individuo), a course (disciplina) and a curriculum (curso).

The process of pragmatic interpretation needs to know these and other relations structures so that it can "talk" about them. It must also be able to query them to access the stored data and answer to the user question.

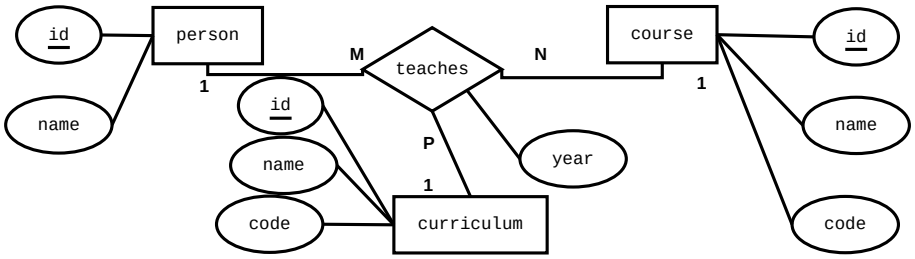

Fig. 3. ER fragment representation

The equivalent ISCO representation (Fig. 5) maps each relation/table to a class that can then be accessed through ISCO predicates. Although NL-Ue only uses "select" predicates, ISCO also supports "inserts", "updates" and "deletes" [1].

This declarative description of the relational database schemas is automatically generated. Based on this class description, ISCO (also) automatically generates predicates to access the stored data. These issues increases decisively the portability level of NL-Ue.

The goal of the pragmatic interpreter is to map the LPO predicates to these ISCO goals so that IIS-UE databases can be directly queried. For example:

```
person(PERSON, PERSON_NAME, _),
teaches(COURSE, 2005, PERSON, _),
course(333, COURSE_NAME, _).
```

collects all persons (teachers) that teach the course with id 333 in the year of 2005.

NL-Ue also benefits the advantages of the contextual branch of GnuProlog: GnuProlog-cx. This is the context-based variant of the well known GnuProlog language which besides all the base features, also enables contextual goal evaluation [2]. While GnuProlog has a flat predicate namespace, it's contextual variant overrides this problem by defining evaluation units. Each unit has it's own namespace which makes possible to have the same goal definition in distinct units. Contextual goals can be called from other units by means of an explicit contextual call.

This feature is essential for controlling the pragmatic interpretation process because there are numerous interpretation possibilities to consider. Contextual evaluation will restrict these possibilities making the process lighter and more efficient.

3.2 Control rules

Pragmatic interpretation is guided by a set of control rules. These rules are based on the ISCO description of the IIS-UE repositories, adding contextual

```
create table "person" (
  "id" integer default (nextval ('is__entity_id'))
       primary key,
  "name" text,
  "gender" integer references "gender" ("id"));
create table "teaches" (
  "course" integer references "course" ("id"),
  "year" integer,
  "teacher" integer references "person" ("id"),
  "curriculum" integer references "curriculum" ("id"));
create table "course" (
  "id" integer default (nextval ('is__course_id'))
       primary key,
  "name" text,
  "code" text);
create table "curriculm" (
  "id" integer default (nextval ('is__curriculum_id'))
       primary key,
  "code" integer unique,
  "name" text,);
```

Fig. 4. SQL fragment representation

information to the user question analysis. ISCO classes, as we've already seen, represent entities and/or relations. Each one of them will have distinct kinds of control rules.

Each rule abducts a set of ISCO goals and incrementally builds the evaluation context of the remaining pragmatic interpretation. The final set of abducted ISCO goals will then be evaluated so that a solution can be found for the user sentence/question. One evaluation unit is generated for each class (IIS-UE relation). Within these units five distinct types of rules can be found:

- entity access rules - one rule is generated so that the stored data can be accessed by NL-Ue for classes that represent entities [2]
- proper noun rules - validate the existence of entities with a specific name in a particular context of evaluation [Fig. 7]
- number rules - identify entities that have a property with a specific number
- relation rules - These rules establish wider connections between referents. Typically they are the most numerous and they are the main responsible for the complexity of pragmatic interpretation. They are also responsible for fixing the pp-attachment problem [19] [9]. There are four kinds of such rules:
  - Self relation rules - generated only for entities, they intend to relate two referents that refer the same type of entity.

[2] a class represents an entity if it has a simple primary key

```
class person.
  id: serial. key.
  name: text.
  gender: gender.id.
class teaches.
  course: course.id. index.
  year: int. index.
  teacher: person.id. index.
  course: course.id. index.
class course.
  id: serial. key.
  name: text.
  code: text.
class curriculum.
  id: serial. key.
  code: int. unique.
  name: text.
```

Fig. 5. ISCO fragment representation

```
for each CLASS
  build evaluation unit with:
    entity access rules,
    proper nouns rules,
    number rules,
    relation rules
    external domain rules
  end build
end for
```

Fig. 6. Control rules generation algorithm

  - Entity relation rules - For each entity (only), these rules associate its primary key with each one of its domain restricted arguments (foreign keys).
  - Argument relation rules - Generates a rule for each pair of domain restricted arguments, considering their relation as a possible interpretation.
  - External relation rules - Establish wider relations not directly within a class but mentioning other classes that refer the first one as foreign key.
- External domain rules - For each argument of class C1 that refers a class C2 as foreign key, a set of rules for C1 interpretation is added to its evaluation unit.

These rules definition is generic (for any application context), not only for the IIS-UE domain. Evaluation follows the pragmatic interpretation. After determining the ISCO representation(s) of the user question, they are evaluated by

directly querying IIS-UE's databases. This evaluation will restrain the sentence referents to the possible instantiations.

```
name(A, PATTERN, []) :-
  check_class_domain(person) :> item(A),
  check_domain(text, PATTERN) :> item(A),
  add_to_context([person(A, _, _)]).
```

collects in A all teachers which have the string PATTERN in its name. This rule is applied in sentences like: "*the teacher John...*"

Fig. 7. Control rule example for proper nouns: class *person*

4 Clarification mechanism

The sentence pragmatic interpretation may lead to multiple results, reflecting the sentence ambiguity and/or the database structure ambiguity. In this case the system needs to clarify the user intentions by means of a (clarification) dialog.

To show NL-Ue's usage and focus on it's dialogue/clarification mechanism, let's see an example question [3] [Fig. 8].

Que docentes leccionam a disciplina de Gestão?
`(Which teachers lecture the Management course?)`

Fig. 8. Example question

This sentence has one semantic interpretation and one pragmatic interpretation [Fig. 9] which associates all possible teachers of some management course with all possible management courses (that exist within IIS-UE) [4].

Referent `A` is instantiated with all possible teachers (that exist in IIS-UE) because no other restriction was made to it, while referent `B` is instantiated with all courses that contain the name *Management*.

The evaluation module will define a set of possible answers. If no ambiguity is found the dialogue manager will directly answer the user with the (only) answer

[3] The system was developed for the portuguese language but questions will be translated to english for easier understanding

[4] this sentence has more pragmatic interpretations , some of them a little bit awkward (at least within the portuguese language). To simplify and because pragmatic disambiguation is not the aim of this paper, we will consider just this one.

Semantics:

```
teacher(A), teaches(A, B), course(B), rel(B, C),
name('Management',C)
```

Pragmatic instantiation:

```
A in [47105..47787] - male/female - plural
B in [188:194:247:259:346:352:486:550:558:835:1053:1108:1210:1270:
      1287:1293:1320:1355:1412:1414:1423..1424:1466:1487:1657:1659:
      1688..1689:1752:1781:1849:1868:1908:1999:2010:2069:2151:2191:
      2249..2252:2441:2479:2525:2534:2588:2598:2698:2754:3107:3134:
      3152:3161:3189:3257:3331:3393:3502:3584:3593:3696:3865:3886:
      3918:3928:4013] - female - singular
C = B
```

Fig. 9. Semantic/Pragmatic interpretation results

found. But if more than one possible answer is found, then the dialogue manager enters in a clarification process [Fig. 10].

```
while not a terminal condition:
  collects properties for each referent
  proceeds with heuristic evaluation
  chooses the best property
  questions the user / receives the answer
  restrains solutions to the question result
end while
```

Fig. 10. Dialogue Manager: clarification algorithm

A `terminal` condition will be reached if only one answer is achieved.

After the evaluation process, possible solutions will be grouped by referent (in our example we have three, referring to `teachers`, `Management` and `course`). This grouping is done to collect the properties of each referent. The best property will be chosen to make a new question.

In wh-questions (which is the case) the referent which is target of the user question (in this case, referent `A`) is excluded from the set of referents to analyze. It makes no sense to make questions referring to what the user wants to know. For each (relevant) referent, different kinds of properties will be collected:

- Class properties: identifies as a referent property its type/class within the scope of IIS-UE - a referent may be a course, a teacher, a student, etc.

– Direct properties: identifies as referent properties direct attributes of its class - if a referent refers to a student, it may have properties as its name, its student number, its gender, etc.
– Indirect properties: identifies as referent properties, attributes of classes that refer the referent class as a foreign key - if a teacher lectures a specific course, then the act of teaching it may be considered as a property of the teacher referent.

```
name('Control Management'), name('Computer Management'),
name('Personal Management'), belongs_to('Sociology department'),
belongs_to('Economics department'), lectured_hours(3.5),
lectured_to(Computer Engineering curriculum'), etc.
```

Fig. 11. Properties Set

After collecting the referent properties [Fig. 11] they will be evaluated with the aim to find the "better one". Heuristic evaluation is used to weight the quality of each property found. The evaluation of a property is a linear sum of three distinct criteria [Fig. 12].

∀ `properties p: weight(p) = w1(p) + w2(p) + w3(p)`
where

– `w1(p)` - evaluates the ability that the property has to split equally the referents set based on the total number of referents (`Rt`) and the number of those which have the property (`Rp`): `w1(p) = 1 - Rp/Rt`.
– `w2(p)` - evaluates the semantic potential of the property preferring textual (`T(p)`) to numeric ones: `If T(p) then w2(p) = 0.5 else w2(p) = 0.1`.
– `w3(p)` - evaluates the probability that the user knows this kind of property, preferring more generic properties. It assumes the user has more knowledge of conceptual properties (`C(p)`), than specific ones: `If C(p) then w3(p) = 0.5 else w3(p) = 0.1`.

Fig. 12. Properties weight heuristics

After the properties evaluation, each one of them is associated with a specific weight that reflects its potential to generate a relevant question [Fig. 13].[5]

For example the properties `belongs_to('Management department')` and `belongs_to('Mathematics department')` are semantically equal (weight 2)

[5] the heuristics shown and the relation between them (relative weights) were incrementally built and based on a set of examples and their evaluation results

```
weight(belongs_to('Management department'), 0.987),
weight(lectured_hours(2), 0.987), weight(belongs_to('Economics
department'), 0.696), weight(belongs_to(Mathematics department'),
0.696), etc.
```

Fig. 13. Properties evaluation

and refer to the same concept - department (weight 3) but the first one as a greater ability to split the referents set which gives it a larger weight. While the property `belongs_to('Management department')` is semantically richer (weight 2), the property `lectured_hours(2)` has a greater ability to split the referent sets, making both properties to have the same weight.

Having weighted the quality of the properties, they are grouped by type (`belongs_to, lectured_hours, name, lectured_to, etc.`). Each group inherits the best weight of its members and the best group is chosen to build the question (and the possible answers). Properties are grouped because the clarification process consists of questions with alternative answers, being the alternatives the elements of the chosen group. If the number of alternatives exceeds a previously determined limit, this property is ignored and the system recursively gets the next one. In our example the chosen group is composed by all the `belongs_to` properties [Fig. 15].

```
"Management course" belongs to:

1) Education department
2) Sociology department
3) Agricultural department
4) Management department

USER: 3
```

Fig. 14. Clarification #1

Besides specifying one of the possible alternatives, the user has only one other possibility which is to say *"?"* meaning *"I don't know"*. In this case the system will recursively get the next "best property" and make a new question.

If the user answers one of the alternatives (1-6) the system will restrain and re-evaluate the possible solutions (according to the users answer) and the clarification algorithm returns to its beginning. After re-evaluating the possible solutions, the chosen property referred to the number of lectured hours. As the user didn't knew the answer, the system recurred to the next "best property" and tried to clarify using the curriculums to which the course is lectured, leading to a final solution according to the user additional information.

```
"Management course" is lectured:

1) 2 hours/week
2) 3 hours/week

USER: ?

"Management course" is lectured to:

1) Biophysics curriculum
2) Agriculture Engineering curriculum
3) Computer Engineering curriculum

USER: 2

Answer
Teacher: Francisco João Santos Silva
Course:  Water resources management
```

Fig. 15. Clarification #2

5 Evaluation

A dialogue/clarification system to query integrated databases in Natural Language should be evaluated from different aspects:

- Portuguese language coverage: The syntactic analysis is done using VISL [7] which is one of the best portuguese full parsers for portuguese. The semantic coverage is restraint by the databases vocabulary (tables, attributes names and entity names) and a synonym dictionary that can be updated by the users. The dialogue system is able to answer Yes/No and Wh questions.
- The pertinence of the user question pragmatic interpretation: This aspect is reflected in the system clarification dialogues and answers. Our system enables the inference of some relations that clearly natural language forbids due to the rules for pragmatic interpretation of pp-sentences. However, we can state that normally this does not constitutes a problem for our users, since on the average they are able to obtain their answer in 3 dialogue steps.
- The heuristic function quality for choosing the question to clarify user sentence: The evaluation of this aspect must be done using the results of a group test, which isn't done yet. Probably the use of some other heuristics may be more efficient. This is an aspect that must be analyzed in the future.
- The efficiency of the system: How long does the system takes to answer a question? Some results are shown in this section.

The dialog/clarification mechanism is not fully tested but this set of heuristic functions lead the system in 70% of the questions to reach an answer after 3 questions/answers or less.

The results presented in table 1 refer to tests made with the IIS-UE main relational repository, which contains 159 tables with an average of 2281 rows/table. The previously built control rules ascend to the number of 1400 which makes an average of 8 rules for each class. Results are based on distinct sentences[6] representative of possible questions to NL-Ue. No specific treatment is given to questions asked multiple times with slight variations in its structure.[7]

Sentence	Context	CPU Time	Real Time	Clarification
Does Mary teaches Databases?	no	0.878	4.970	yes
	yes	0.640	3.285	
Does Mary Higgins teaches the Informatics curriculum?	no	0.910	5.111	no
	yes	0.488	3.164	
Which teachers teach the Management course?	no	5.864	22.324	yes
	yes	3.544	14.225	
Which are the Physics teachers?	no	6.321	20.970	yes
	yes	4.216	12.960	

Table 1. Results

Times shown (in seconds) refer to the time the system takes from the input of the user to the answer (or to the first clarification question[8]). Further evaluation must be done considering real tests because its conditioned to the clarification process and to the users subjectivity.

Each sentence was tested with and without contextual evaluation. Its use - with GnuProlog-cx - is relevant in the pragmatic interpretation module, minimizing its complexity by reducing the search space. It leads to a gain of efficiency in the order of 80% in more complex sentences and 50% in more simpler ones - cpu time. Yes/no questions (second and third) have a lower cpu processing time than wh. Clarification was needed in three of the sentences.

[6] sentences in english may not present the same structure or complexity than in portuguese

[7] possible future development which would require the recording of the sentences analysis for future reference

[8] This includes: syntactic and semantic analysis; the pragmatic interpretation (each sentence may have more than one interpretation) and finally the evaluation of each sentence interpretation. During this the evaluation, the discourse referents that are constraint variables restrained to a set of database entities, are validated by testing the truth value of the sentence conditions, that are database relations. This way, the process time of a query sentence will depend on the number of database entities associated to each discourse referent. This fact explains why sentence 2 (which has more pragmatic interpretations then sentence 3) takes less time then sentence 3.

6 Conclusions and Future Work

Our system aims to be a natural language interpretation system that uses specific tools for increasing efficiency and getting results in real time.

This paper shows how pragmatic interpretation of complex sentences can be handled through a set of contextual control rules which guide the process of interpretation, increasing it's efficiency and without the loss of any results.

The use of the ISCO development framework gives NL-Ue a high level of portability in accessing and querying different sets of databases.

The contextual evaluation strategy (available through the use of Gnuprolog-cx) is applied in the control rule generation and usage, making the pragmatic evaluation process practical and more efficient once the search space is smaller.

NL-Ue is a question/answering system capable of a clarification dialog when needed. It is able to identify the users needs and to extract the desired information from relational databases.

The use of a dedicated development framework and its ability to describe and access distinct data sources in a homogeneous way increased the systems portability rate.

Contextual evaluation gives pragmatic interpretation a more linear and simple treatment decreasing its computational complexity, which can be decisive in systems with large information repositories.

NL-Ue supports yes/no and wh-questions. Besides, it includes a clarification mechanism in which the system dialogues with the user when the desired information is ambiguous.

The clarification/dialogue module uses referent properties to find relevant questions trying to reach an answer the faster it cans. Heuristic evaluation is used to ensure the quality of the questions.

Having developed the core mechanism, the next steps will be to confirm the correctness of the followed methodology. For that, the system must be evaluated considering:

- its usefulness (for the users)
- the correctness of pragmatic rules generation
- the types of questions treated
- the heuristic function quality

The system will be available to the public through Universidade de Évora: http://www.uevora.pt

References

1. Salvador Abreu. Isco: A practical language for heterogeneous information system construction. In *Proceedings of INAP'01*, Tokyo, Japan, October 2001. INAP.

2. Salvador Abreu and Daniel Diaz. Objective: In minimum context. In Catuscia Palamidessi, editor, *ICLP*, volume 2916 of *Lecture Notes in Computer Science*, pages 128–147. Springer, 2003.
3. Salvador Pinto Abreu. A Logic-based Information System. In Enrico Pontelli and Vitor Santos-Costa, editors, *2nd International Workshop on Practical Aspects of Declarative Languages (PADL'2000)*, volume 1753 of *Lecture Notes in Computer Science*, pages 141–153, Boston, MA, USA, January 2000. Springer-Verlag.
4. I. Androutsopoulos. Interfacing a natural language front-end to a relational database, 1992.
5. I. Androutsopoulos, G.D. Ritchie, and P. Thanisch. Natural language interfaces to databases–an introduction. *Journal of Language Engineering*, 1(1):29–81, 1995.
6. Ion Androutsopoulos and Graeme Ritchie. Database interfaces. In H. Moisl R. Dale and H. Somers, editors, *Handbook of Natural Language Processing*, pages 209–240. Marcel Dekker Inc., 2000.
7. Eckhard Bick. A constraint grammar based question answering system for portuguese. In *EPIA*, pages 414–418, 2003.
8. Marco De Boni. An analysis of clarification dialogue for question answering.
9. Michael Collins and James Brooks. Prepositional attachment through a backed-off model. In David Yarovsky and Kenneth Church, editors, *Proceedings of the Third Workshop on Very Large Corpora*, pages 27–38, Somerset, New Jersey, 1995. Association for Computational Linguistics.
10. Development and T. User. The postgresql development team. postgresql user's guide, 1996.
11. D. Diaz. http://www.gnu.org/software/prolog, 1999.
12. J. Ginzburg. Clarifying utterances, 1998.
13. Joaquim Godinho, Luis Quintano, and Salvador Abreu. Universidade de Évora's Integrated Information System: An Application. In Hans Dijkman, Petra Smulders, Bas Cordewener, and Kurt de Belder, editors, *The 9th International Conference of European University Information Systems*, pages 469–473. Universiteit van Amsterdam, July 2003. ISBN 90-9017079-0.
14. H. Kamp and U. Reyle. *From Discourse to Logic*. Kluwer, Dordrecht, 1993.
15. B. Katz, S. Felshin, D. Yuret, A. Ibrahim, J. Lin, G. Marton, A. McFarland, and B. Temelkuran. Omnibase: Uniform access to heterogeneous data for question answering, 2002.
16. Boris Katz and Jimmy J. Lin. Start and beyond.
17. Irene Rodrigues Luis Quintano and Salvador Abreu. Relational Information Retrieval through natural language analysis. In *INAP'01, Tokyo, Japan*, 2001.
18. Irene Rodrigues Luis Quintano, Paulo Quaresma and Salvador Abreu. A Natural Language dialogue manager for accessing databases. In Mamede N. and Ranchhod E., editors, *Proceedings of PorTAL'02, Faro Portugal*, 2002.
19. Paola Merlo. Generalised pp-attachment disambiguation using corpus-based linguistic diagnostics. In *EACL*, pages 251–258, 2003.
20. H. Kautz O. Etzioni and A. Popescu. Towards a theory of natural language interfaces to databases. In *Intelligent User Interfaces (IUI)*, 2003.
21. Ellen M. Voorhees. Overview of the TREC 2001 question answering track. In *Text REtrieval Conference*, 2001.
22. David H. D. Warren and Fernando C. N. Pereira. An efficient easily adaptable system for interpreting natural language queries. *American Journal of Computational Linguistics*, 8(3-4):110–122, 1982.