A natural language interface for information retrieval on semantic web documents

Paulo Quaresma and Irene Pimenta Rodrigues

Departamento de Informática,
Universidade de Évora,
Portugal
Email: *[pq,ipr]@di.uevora.pt*

Abstract. We present a dialogue system that enables the access in natural language to a web information retrieval system. We use a Web Semantic Language to model the knowledge conveyed by the texts. In this way we are able to obtain the associated knowledge necessary to perform the different analysis stages of natural language sentences.
In the context of information retrieval, we aim to develop a system that, by increasing the interaction management capabilities, is able to achieve a better degree of cooperativeness and to reduce the average number of interactions needed to retrieve the intended set of documents.
The documents in the IR system considered here are composed by the set of documents produced by the Portuguese Attorney General since 1940. These documents were analyzed and an ontology describing their structure and content was defined. Then, they were automatically parsed and a (partial) semantic structure was created. The ontology and the semantic content was represented in the OWL language.
An example of a user interaction session is presented and explained in detail.

Keywords: Agents, Knowledge Based Systems, Web Semantics, Information Retrieval, Natural Language Dialogues.

1 Introduction

As the size and complexity of documents bases increase, users develop new desires leading to information retrieval systems that should be able to act as rational, autonomous, and cooperative agents, helping them in their searches.

Actually, we claim that in order to improve the search results, information retrieval systems should be able to represent and to use the knowledge conveyed by the texts, using web semantic languages to model the knowledge.

In fact, in the last years, much work has been done trying to satisfy some of these goals creating what is sometimes called third generation knowledge based systems [YS99].

Our IR system aims to achieve some of these goals and it has an architecture with two layers:

– The information retrieval layer – This layer integrates the text search engine with natural language processing techniques;
– The interaction layer – This layer is the responsible for handling the interaction between users and the information retrieval modules.

These two layers are implemented as autonomous agents, which are able to communicate between them and with other agents. Communication is achieved through the interpretation of the received actions in the context of the agents' mental state. Each agent models its mental state, namely its beliefs, intentions and goals, and plans its own actions trying to be as much cooperative as possible.

Cooperation is achieved through the inference of the other agents' intentions from their actions. The inferred agents' intentions are the input of a abductive planning procedure, which selects the actions needed to satisfy the agents goals.

In the context of information retrieval, one of the main goals of our work is to show the need for interaction management capabilities and to develop a system that is able to achieve a better degree of cooperativeness and to reduce the average number of interactions needed to retrieve the intended set of documents.

Namely, we would like our agents to be able: To model their mental state (beliefs, intentions, goals); To record all the interactions (other agents questions and their own answers) [CL99,CCC98]; To infer the new agents attitudes from their mental state, the interaction history, and the agents actions. In fact, agents queries may not directly state their final goals [Loc98,Pol90]; And to plan their actions, using the inferred agents attitudes and their cooperative rules of behavior.

The integration of specific information retrieval agents with dialogue management agents in a general information retrieval system is a quite new approach and it allows us to model the system's behavior (cooperative, pro-active) and to modularize the solution, specializing the agents, and being able to handle future extensions of the system (through the use of new agents or the extension of the existent ones).

Some of the components of our system can be compared with other existent IR systems. For instance, our IR agent is based on SINO [GMK97] and it was changed in a way that has many similarities with the work described in [BvWM$^+$99], namely, allowing the extraction of textual information using localization, inference, and controlled vocabulary. As in [OS99], we are also able to use concepts and a concept taxonomy in order to retrieve sets of documents. On the other hand, in the dialogue management domain, the use of speech acts to recognize plans has many similarities with the work of Carberry [CL99] and Litman [LA87] and the representation of plans as mental attitudes was also the approach followed by Pollack [Pol90] in her work.

The different analysis stages of a natural language sentence requires specific domain knowledge. Namely, to obtain the sentence pragmatic interpretation we need to have a large knowledge base describing the sentence entities and its relations to other entities. This problem is overcame by using Web semantic Languages to describe an ontology representing our world (documents domain) main classes of the objects, their properties and their relations. The semantic

content of our database documents can then be represented using the defined ontology in the Web semantic language allowing the system to retrieve the information conveyed by the user sentences. The extraction of the semantic content of our documents is done by using specific tools that perform a robust natural language analysis in order to extract and represent some of the document entities and relations.

By now, it is only possible to partially represent the documents semantic content because there is a need for more complete ontologies and more powerful natural language analyzers.

As basic semantic language we are using the Web Ontology Language (OWL) - [www00]) language, which is defined using the RDF (Resource Description Framework - [LS99,BG99]) language and it has a XML version.

Using OWL it was possible to represent the documents structure and some of its semantic content. Moreover, the user natural language queries can be semantically analyzed accordingly with the base ontology.

Using this approach, the dialogue system that we propose is able to supply adequate answers to the Portuguese Attorney General's Office documents database (PGR). For instance, in the context of an user interrogation searching for information on injuries in service, the following question could be posed:

Who has service injuries?

The user is expecting to have as an answer some characteristics of the individuals that have been injured at work. He does not intend to have a list of those individuals or the documents that refer to the act of asserting that the injury was in service. In order to obtain the adequate answer our system must collect all individuals referred in the documents database that have a service injury and then it must supply the common characteristics to the user in a dialogue.

As it was referred, our system must have an adequate representation of injuries and individuals in a semantic web ontology and all documents must have the adequate labels in the semantic web language representing the knowledge on 'Injuries' and 'Individuals' and 'Service' conveyed by the document[1].

The answer to the above question could be: 'Individuals that were agents of an injury and of a subaction of working that occurred in a overlapping time interval and the cause of the injury is not exclusively responsibility from the agent'

This information can be obtained by extracting what properties those individuals have in common or by manually representing parts of the Portuguese law. The possibility of extracting what are the common characteristics of a set of objects is a powerful tool for presenting the answers to our users. This behavior can be achieved by choosing an adequate ontology to represent the objects including events present at the documents.

The dialogue system obtains the knowledge necessary for the interpretation of natural language sentences from the Semantic Web description of the documents databases. The vocabulary and the rules for the semantic pragmatic

[1] By now this has be done manually.

interpretation are automatically generated using the existent ontology (this will be explained in more detail in the next sections).

The dialogue module is built with a logic programming language for describing actions and events, EVOLP [ABLP02], which allows the system to make inferences about the user intentions and beliefs and to be able to have cooperative dialogues with the users.

The remainder of this article is structured as follows: in section 2, a brief introduction to the OWL Semantic Web language is presented. In section 3, the EVOLP language is briefly described. In section 4 the overall structure of the system is presented; section 5 deal with the semantic/ pragmatic interpretation. In section 7 a more extensive example is presented and, finally, in section 8 we discuss some current limitations of the system and lay out possible lines of future work.

2 Web semantics

The documents domain knowledge was partially represented using a semantic web language. The first step was to define an ontology adequate for the domain. We have selected a domain of the Portuguese Attorney General documents – injury in service (asserted or retracted) – and, in this domain, we have selected smaller sub-domains, such as, injuries for firemen, or militaries, or civilians; injuries at the traject from home to work or vice-versa, injuries at work place, injuries at labor time.

The ontology was represented using the OWL (Ontology Web Language) language, which is based on the DAML+OIL (Darpa Agent Markup Language - [www00]) language, and it is defined using RDF (Resource Description Framework - [LS99,BG99]). As an example, the *Individual* class is presented below (only some of the class attributes are shown):

```
<owl:Class rdf:ID="Individual">
     <owl:label>Individual</owl:label>
</owl:Class>
<owl:DatatypeProperty rdf:ID="individualCode">
        <owl:domain rdf:resource="#Individual"/>
        <owl:type rdf:resource="&owl;FuncionalProperty"/>
        <owl:range rdf:resource="&xsd;integer"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="individualName">
        <owl:domain rdf:resource="#Individual"/>
        <owl:type rdf:resource="&owl;FuncionalProperty"/>
        <owl:range rdf:resource="&xsd;String"/>
</owl:DatatypeProperty>
<owl:ObjectProperty rdf:ID="individualProfession">
        <owl:domain rdf:resource="#Individual"/>
        <owl:range rdf:resource="#Profession"/>
</owl:ObjectProperty>
```

After defining an ontology, the documents need to be analyzed and their semantic content should be represented in OWL. However, this is a very complex open problem and we have decided to manually represent subsets of the document content for the chosen sub-domains.

These two steps (ontology + document semantic representation) are the basis of the proposed system and allow the implementation of other steps, such as, the semantic/pragmatic interpretation and the dialogue management.

3 EVOLP

As referred, there is also a need for a declarative language to represent actions and to model the evolution of the knowledge.

In [APP+99] it was introduced a declarative, high-level language for knowledge updates called *LUPS* ("Language of UPdateS") that describes transitions between consecutive knowledge states. Recently, a new language, EVOLP [ABLP02], was proposed having a simpler and more general formulation of logic program updates. In this section a brief description of the EVOLP language will be given, but the interested reader should refer to the cited article for a detailed description of the language and of its formalization.

EVOLP allows the specification of a program's evolution, through the existence of rules which indicate assertions to the program. EVOLP programs are sets of generalized logic program rules defined over an extended propositional language L_{assert}, defined over any propositional language L in the following way [ABLP02]:

- All propositional atoms in L are propositional atoms in L_{assert}
- If each of $L_0, \ldots, L_n$ is a literal in L_{assert}, then $L_0 \leftarrow L_1, \ldots, L_n$ is a generalized logic program rule over L_{assert}.
- If R is a rule over L_{assert} then $assert(R)$ is a propositional atom of L_{assert}.
- Nothing else is a propositional atom in L_{assert}.

The formal definition of the semantics of EVOLP is presented at the referred article, but the general idea is the following: whenever the atom $assert(R)$ belongs to an interpretation, i.e. belongs to a model according to the stable model semantics of the current program, then R must belong to the program in the next state. For instance, the following rule form:

$$assert(b \leftarrow a) \leftarrow c \tag{1}$$

means that if c is true in a state, then the next state must have rule $b \leftarrow a$.

EVOLP has also the notion of external events, i.e. assertions that do not persist by inertia. This notion is fundamental to model interaction between agents and to represent actions. For instance, it is important to be able to represent actions and its effects and pre-conditions:

$$assert(Effect) \leftarrow Action, PreConditions \tag{2}$$

If, in a specific state, there is the event *Action* and if *PreConditions* hold, then the next state will have $Effect$.

4 Natural Language Dialogue System

As was already stated the main goal of this work was to build a system that could get a Portuguese natural language sentence sent by a user through a web interface and respond accordingly. To answer the question/sentence the system has to pass it from a web-based interface to a specialized process; the process must analyze the sentence accessing the documents database(s); and finally when acquiring all needed information, it has to build a comprehensive answer and pass it to the web-based interface.

The analysis of a natural language sentence is split in four subprocesses: Syntax, Semantics, Pragmatics, Dialogue manager.

The user asks the question, which is redirected to an active process that already has information about all the documents semantic structure (the ontology). A specialized natura language processing agent manages the conversion of the sentence to declarative logic programming predicates. These predicates allow the access to the documents through the information retrieval system. After analyzing the sentence received, the process has to generate an adequate answer, which will be shown to the user through the web interface.

Syntax Analysis: Our syntactic interpreter was built using *Chart Parsers*[GM89]. This is one of many techniques to build syntactic interpreters. The decision of developing the interpreter using this technique was mainly because chart parsers can parse incomplete sentences. The user can place complete or incomplete questions and the system must be able to answer them accordingly, so the need to parse incomplete sentences is essential. The chart parser uses a set of syntactic rules that identify the Portuguese sentence structures and tries to match these rules with the input sentence(s). As an example, the following sentence:

"Who has a injury in service?"

Has the following structure:

```
phrase([np([det(who,_+_+_), n('individual',_+s+m)]),
        vp(v('have',3+p+_)),
           args_v([np([det(a,_+s+_), n('injury',_+s+_),
             pp(in,np([n('service',_+s+m)]))])]).
```

Semantic Interpretation: Each syntactic structure is rewritten into a First-Order Logic expression. The technique used for this analysis is based on DRS's (Discourse Representation Structures [KR93]). This technique identifies triggering syntactic configurations on the global sentence structure, which activates the rewriting rules. We always rewrite the pp's by the relation 'rel(A,B)' postponing its interpretation to the semantic pragmatic module. The semantic representation of sentence is a DRS build with two lists, one with the new sentence rewritten and the other with the sentence discourse referents. For instance, the semantic representation of the sentence above is the following expression:

individual(A), injury(B), service(C), rel(B,C), have(A,B).

and the following discourse referents list:

[ref(A,p+_+_,what),ref(B,s+_+_,undef),ref(C,p+_+_,undef)]

5 Semantic/Pragmatic Interpretation

The semantic/pragmatic module receives the sentence rewritten (into a First Order Logic form) and tries to interpret it in the context of the document database information (ontology). In order to achieve this behavior the system tries to find the best explanations for the sentence logic form to be true in the knowledge base for the semantic/pragmatic interpretation. This strategy for interpretation is known as "interpretation as abduction" [HSAM90]. The knowledge base for the semantic/pragmatic interpretation is built from the Semantic Web description of the document database. The inference in this knowledge base uses abduction, restrictions (GNU Prolog Finite Domain (FD) constraint solver) and accesses the document databases through an Information Retrieval Agent. The knowledge base rules contain the information for each term interpretation in the sentence logic form as logic programming terms. The KB rules are generated from the Semantic Web databases descriptions. This process was described in detail in [QRA01]. From the description of the class *injury*, the KB has rules for the interpretation of the predicates: *injury(A)* and *rel(A,B)*. Suppose there exists the following description of the class $Injury$ and of two subclasses[2]:

```
<owl:Class rdf:ID="Injury">
     <owl:label>Injury</owl:label>
</owl:Class>
<owl:DatatypeProperty rdf:ID="injuryCode">
     <owl:domain rdf:resource="#Injury"/>
     <owl:type rdf:resource="&owl;FuncionalProperty"/>
     <owl:range rdf:resource="&xsd;integer"/>
</owl:DatatypeProperty>
<owl:ObjectProperty rdf:ID="individual">
     <owl:domain rdf:resource="#Injury"/>
     <owl:type rdf:resource="&owl;FuncionalProperty"/>
     <owl:range rdf:resource="#Individual"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="event">
      <owl:domain rdf:resource="#Injury"/>
      <owl:range rdf:resource="#Event"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="supportDocuments">
      <owl:domain rdf:resource="#Injury"/>
      <owl:range rdf:resource="#DocumentList"/>
</owl:ObjectProperty>

<owl:Class rdf:ID="Injury_in_service">
      <owl:label>Injury in Service</owl:label>
      <rdfs:subClassOf rdf:resource="#Injury"/>
</owl:Class>
<owl:Class rdf:ID="Injury_not_in_service">
```

[2] Due to its complexity, in this paper we only present a small subset of the complete ontology.

```
      <owl:label>Injury not in service</owl:label>
      <rdfs:subClassOf rdf:resource="#Injury"/>
</owl:Class>
```

This description means that "injury" is a class which has some properties: the individual that gets the injury, an event describing the action supporting the injury, and a list of supporting documents. Moreover, "injury" has two sub-classes: injury in service and injury not on service. Using the ontology, a set of rules was automatically produced enabling the semantic/pragmatic interpretation of a sentence like " injury" as the Predicate Logic expression *injury(A,_,_,_,_)*. This description will also give rise to rules allowing for the interpretation of noun phrases such as "injury in service".

```
rel(A,B) <-
  injury(A),
  service(B),
  abduct(injury_in_service(A,_,_,_,_)).
```

From the previous description of the class Individual (in section 2) the KB has rules that will allow the interpretation of the noun "Person" and of noun phrases such as: "Individual name", "Individual Profession". One of the generated rules is:

```
rel(B,A) <-
  individual(B),
  profession(A)
  abduct(individual(B,_,A,_)).
```

This rule enables us to obtain the expression *individual(B,_, A,_)* as the interpretation of the noun phrase "profession of individual".

During the semantic/pragmatic interpretation the evaluation of a predicate like "Individual(A)" is done by an access to the Semantic Web documents. The result of such an evaluation is the constraint of variable A to database identifiers of objects from class individual. The interpretation of nouns (eg. injury(A)) is done by accessing the documents database in order to collect in (constraint) A all entities identifiers that have in their name the word 'injury'. The result of interpreting the sentence represented by [3]:

individual(A),injury(B),services(C),rel(B,C),rel(A,B)

[ref(A,s+_+_,what),ref(B,s+_+_,undef),ref(C,s+_+_,undef)]

is the following expression:

$injury\_in\_service(B, A, \_, \_, \_), individual(A, \_, \_, \_)$. Where:

- $B =_{\#}$ (1046..1049 : 1345 : 1456..1457) – B constrained to all injuries in service.
- $A =_{\#}$ (7001...7852) – A is constraint to individuals

The above LP expression contain the possible interpretations of the sentence in the context of our documents database. The dialogue manager is responsible to interact with the user by supplying him an answer or by posing him pertinent questions.

[3] The interpretation of A *have* B is the same of B *of* A, so $have(A, B)$ is equivalent to $rel(A, B)$

6 Dialogue Manager

The Dialogue Manager must recognize the speech act associated with the sentence (in this domain it can be an $inform$, a $request$, or a $askif$ speech act), to model the user attitudes (intentions and beliefs), and to represent and make inferences over the dialogue domain. In order to achieve this goal the system needs to model the speech acts, the user attitudes (intentions and beliefs) and the connection between attitudes and actions. This task is also achieved through the use of the EVOLP language (see [QR01,QL95] for a more detailed description of these rules). For instance, the rules which describe the effect of an inform, a request, and a ask-if speech act from the point of view of the receptor are:

$$assert(bel(A, bel(B, P))) \leftarrow inform(B, A, P). \quad (3)$$
$$assert(bel(A, int(B, Action))) \leftarrow request(B, A, Action). \quad (4)$$
$$assert(bel(A, int(B, inform\_if(A, B, P)))) \leftarrow ask\_if(B, A, P) \quad (5)$$

In order to represent collaborative behavior it is necessary to model how information is transferred between the different agents:

$$assert(bel(A, P)) \leftarrow bel(A, bel(B, P)). \quad (6)$$
$$assert(int(A, Action)) \leftarrow bel(A, int(B, Action)). \quad (7)$$

These two rules allow beliefs and intentions to be transferred between agents if they are not inconsistent with their previous mental state. There is also the need for rules that link the system intentions and the accesses to the databases:

$$assert(yes(P)) \leftarrow query(P), one\_sol(P), int(A, inform(A, B, P)). \quad (8)$$
$$assert(no(P)) \leftarrow query(P), no\_sol(P), int(A, inform(A, B, P)). \quad (9)$$
$$assert(clarif(P)) \leftarrow query(P), n\_sol(P), int(A, inform(A, B, P)). \quad (10)$$

These three rules update the system's mental state with the result of accessing the databases: yes, if there is only one solution; no, if there are no solutions; and clarification, if there are many solutions (the predicates that determine the cardinality of the solution are not presented here due to space problems). After accessing the databases, the system should answer the user:

$$assert(confirm(A, B, P)) \leftarrow yes, int(A, inform(A, B, P)). \quad (11)$$
$$assert(notint(A, inform\_if(A, B, P))) \leftarrow yes, int(A, inform(A, B, P)). \quad (12)$$
$$assert(reject(A, B, P)) \leftarrow no, int(A, inform(A, B, P)). \quad (13)$$
$$assert(notint(A, inform\_if(A, B, P))) \leftarrow yes, int(A, inform(A, B, P)). \quad (14)$$
$$assert(ask(A, B, C)) \leftarrow cluster(P, C), clarif(P), int(A, inform(A, B, P)). \quad (15)$$

The first two rules define that, after a unique solution query, the system confirms the answer and terminates the intention to answer the user. The next two rules define that, after a no solution query, the system rejects the question and terminates the intention to answer the user. The last rule defines that, after a multiple solution query, the system starts a clarification answer, asking the user to select one of the possible solutions. In order to collaborate with the user we have defined a cluster predicate that tries to aggregate the solutions into coherent sets. The strategy behind this predicate is to aggregate the solutions accordingly with the range of property values of the selected objects. For instance, in the presented example the selected individuals might be clustered by their profession, or by their support documents, or by the events in which they are actors. In the next section, this strategy will be described in more detail.

7 Example

Considering the already presented question:

Who has an injury in service?

The dialogue manager receives this sentence semantic/pragmatic interpretation, as we presented in the previous sections it will be the following expression:

```
injury_in_service(B,A,_,_,_), individual(A,_,_,_).
```

with the following restrictions:

- $B =_{\#} (1046..1049 : 1345 : 1456..1457)$ – B is constraint to all injuries in service.
- $A =_{\#} (7001...7852)$ – A is constraint to individuals
- $[ref(A, p + \_ + \_, what), ref(B, s + \_ + \_, undef), ref(C, p + \_ + \_, undef)]$

After having the sentence re-written into its semantic representation form, the speech act is recognized and we'll have:

```
request(user, system, inform(user, system,
        [injury_in_service(injuryCode=B,individual=A)]))
```

Using the "request" and the EVOLP transference of intentions rules we'll have:

```
int(system,inform(system, user,
        [injury_in_service(injuryCode=B,individual=A)]))
```

Now, using the rules presented in the previous section, the system accesses the databases (via the *query* predicate). Suppose there are several possible solutions. We'll have A and B constrained to related individuals and injuries in service:

- $B =_{\#} (1046..1049 : 1345 : 1456..1457)$
- $A =_{\#} (7030...7842 : 7850)$ – A is constrained to individuals that have an injury in service.

As a consequence of having several solutions, predicate $clarif$ will hold and the system launches the *cluster* predicate to aggregate the obtained solutions (accordingly with the rule presented in the previous section).

```
cluster([injury_in_service(injuryCode=B,individual=A)],C).
```

The $cluster(P, C)$ rule identifies the variable which is the focus of the query (obtained in the syntactical analysis) and aggregates the property values for the associated objects. For instance, in this example it will detect that the query is about individuals (variable A) and it tries to cluster its constrained values accordingly with their professions, events, and documents relation. After having clustered the property values, the system uses an heuristic to choose the property that better divides the objects (by better we mean that the cardinality of the obtained sets has the same magnitude order) and it performs the *ask_select* action.

In this example the answer might be: 'Individuals that are firemen, or militaries or civilians'.

Or, using another property (event list): 'Individuals that were agents of an action getting injuries and of an action that is a subaction of working that occurred in a overlapping time interval and the cause of the injury is not exclusively responsibility from the agent'

8 Conclusions and Future Work

The dialogue system described in this paper is quite complex and some of its modules still require a manual intervention. Namely, it is not possible to have a complete automatic semantic interpretation and representation of the documents.

However, we have developed a small prototype, which is being tested and, in the future, it will be made available to all users in the context of the Portuguese Attorney General's web information retrieval system (http://www.pgr.pt).

Regarding future work, it is clear that all modules have aspects that may be improved: the syntactical coverage of the Portuguese grammar; the coverage of the semantic analyzer (plurals, quantifiers, ...); the ontology coverage; the semantic representation of the documents content; and the capability of the dialogue manager to take into account previous interactions and the user models.

As we have a modular approach to the systems' architecture, there is a complete separation of all the processing phases. This separation will enable us, for instance, to use grammars already developed in other formalisms such as the project described in [Bic00] (in fact, this is an actual project ongoing task).

As a final conclusion, we believe that this system represents a relevant step in the transformation of "traditional" information retrieval systems into semantic-aware IR systems.

References

[ABLP02] J. Alferes, A. Brogi, J. Leite, and L. Pereira. Evolving logic programs. In S. Flesca, S. Greco, N. Leone, and G. Ianni, editors, *JELIA'02 – Proceedings of the 8th European Conference on Logics and Artificial Intelligence*, pages 50–61. Springer-Verlag LNCS 2424, 2002.

[APP+99] J. J. Alferes, L. M. Pereira, H. Przymusinska, T. C. Przymusinski, and P. Quaresma. Preliminary exploration on actions as updates. In M. C. Meo and M. Vilares-Ferro, editors, *Procs. of the 1999 Joint Conference on Declarative Programming (AGP'99)*, pages 259–271, L'Aquila, Italy, September 1999.

[BG99] D. Brickley and R. Guha. *Resource Description Framework (RDF) - Schema Specification*. W3C, 1999.

[Bic00] Eckhard Bick. *The Parsing System PALAVRAS: Automatic Grammatical Analysis of Portuguese in a Constraint Grammar Framework*. PhD thesis, Århus University, Århus, 2000.

[BvWM+99] Tania Bueno, Christiane von Wangenheim, Eduardo Mattos, Hugo Hoeschl, and Ricardo Barcia. Jurisconsulto: Retrieval in jurisprudencial text bases using juridical terminology. In *Proceedings of the ICAIL'99 – 7th International Conference on Artificial Intelligence and Law*, pages 147–155. ACM, June 1999.

[CCC98] J. Chu-Carroll and S. Carberry. Response generation in planning dialogues. *Computational Linguistics*, 24(3), 1998.

[CL99] Sandra Carberry and Lynn Lambert. A process model for recognizing communicative acts and modeling negotiation subdialogs. *Computational Linguistics*, 25(1), 1999.

[GM89] Gerald Gazdar and Chris Mellish. *Natural Language Processing in PROLOG*. Addison-Wesley, 1989.

[GMK97] G. Greenleaf, A. Mowbray, and G. King. Law on the net via austlii - 14 m hypertext links can't be right? In *In Information Online and On Disk'97 Conference, Sydney*, 1997.

[HSAM90] Jerry Hobbs, Mark Stickel, Douglas Appelt, and Paul Martin. Interpretation as abduction. Technical Report SRI Technical Note 499, 333 Ravenswood Ave., Menlo Park, CA 94025, 1990.

[KR93] H. Kamp and U. Reyle. *From Discourse to Logic*. Kluwer, Dordrecht, 1993.

[LA87] Diane Litman and James Allen. A plan recognition model for subdialogues in conversation. *Cognitive Science*, 11(1), 1987.

[Loc98] Karen E. Lochbaum. A collaborative planning model of intentional structure. *Computational Linguistics*, 24(4), 1998.

[LS99] O. Lassila and R. Swick. *Resource Description Framework (RDF) - Model and Syntax Specification*. W3C, 1999.

[OS99] James Osborn and Leon Sterling. A judicial search tool using intelligent concept extraction. In *Proceedings of the ICAIL'99 – 7th International Conference on Artificial Intelligence and Law*, pages 173–181. ACM, June 1999.

[Pol90] Martha Pollack. Plans as complex mental attitudes. In Philip Cohen, Jerry Morgan, and Martha Pollack, editors, *Intentions in Communications*. MIT Press Cambridge, 1990.

[QL95] P. Quaresma and J. G. Lopes. Unified logic programming approach to the abduction of plans and intentions in information-seeking dialogues. *Journal of Logic Programming*, 54, 1995.

[QR01] Paulo Quaresma and Irene Rodrigues. Using logic programming to model multi-agent web legal systems – an application report. In *Proceedings of the ICAIL'01 - International Conference on Artificial Intelligence and Law*, St. Louis, USA, May 2001. ACM.

[QRA01] Luis Quintano, Irene Rodrigues, and Salvador Abreu. Relational information retrieval through natural lanaguage analysis. In *Proceedings of INAP'01*, Tokyo, Japan, October 2001. INAP.

[www00] www.daml.org. *DAML+OIL – DARPA Agent Markup Language*, 2000.

[YS99] John Yearwood and Andrew Stranieri. The integration of retrieval, reasoning and drafting for refugee law: a third generation legal knowledge based system. In *Proceedings of the ICAIL'99 – 7th International Conference on Artificial Intelligence and Law*, pages 117–125. ACM, June 1999.