

Chapter 21

Advanced Relaxation for Cooperative Question Answering

Farah Benamara and Patrick Saint Dizier

benamara@irit.fr, stdizier@irit.fr

Institut de Recherches en Informatique de Toulouse (IRIT),
118 route de Narbonne, 31062 Toulouse cedex 4, France.

1. Introduction

Advanced reasoning for Question Answering (QA) systems, as described in a recent roadmap (http://www-nlpir.nist.gov/projects/duc/papers/qa.Roadmap-paper_v2.doc), raises new challenges since answers are not only directly extracted from written texts or structured databases but also constructed via several forms of reasoning in order to generate answer explanations and justifications. These systems require the integration of reasoning components operating over a variety of knowledge bases, encoding common sense knowledge as well as knowledge specific to a variety of domains by means, for example, of conceptual ontologies. These kinds of QA systems can be viewed as an enhancement, rather than a rival to retrieval based approaches. Integrating knowledge representation and reasoning mechanisms allow, for example, to respond to unanticipated questions and to resolve situations in which no answer is found in the data sources. Cooperative answering systems are typically designed to deal with such situations by providing useful and informative answers. These systems can e.g. identify and explain false presuppositions or various types of misunderstandings found in questions. Constraints relaxation in questions occur when the system cannot find any response. Intensional responses are provided instead of a large unstructured set of extensional answers. Cooperative answering systems can also provide summaries or conditional responses. An overview of these aspects is given in (Gaasterland et al, 94).

WEBCOOP is a logic based QA system that integrates knowledge representation and advanced reasoning procedures to generate intelligent and cooperative

responses to natural language (NL) queries on the web. To evaluate the cost of the procedures involved, the project is developed, in a first stage, on a relatively limited domain that includes a number of aspects of tourism (accommodation and transportation). Our approach requires the development of a knowledge extractor from web pages (similarly to a knowledge extractor operating on passages resulting from an IR component) and the elaboration of a robust and accurate question parser. In WEBCOOP, responses provided to users are built in web style by integrating natural language generation (NLG) techniques with hypertexts in order to produce dynamic responses (Dale et al, 98). NL responses are produced from semantic forms constructed from reasoning processes. During these processes, the system has to decide, via cooperative rules, what is relevant and then to organize it in a way that allows for the realization of coherent and informative responses.

In WEBCOOP, responses are structured in two parts : (1) the production of explanations that report user misconceptions and then (2) the production of flexible solutions that reflect the cooperative *know how* of the system. This component is the most original. It is based on *intensional description techniques* and on *intelligent relaxation procedures* going beyond classical generalization methods elaborated in AI. This component also includes additional *dedicated cooperative rules* that make a thorough use of the domain ontology and of general knowledge.

To have a more accurate perception of how cooperativity is realized in man-man communication, we collected corpora of question answer pairs found in numerous web sites dedicated to tourism. We first noted that human responses are much more diverse than any machine could produce in the near future. Nevertheless, it is possible to *normalize* these forms to more stereotyped utterances that a machine can produce. Next, we noted the large number of situations where constraint relaxations are used, and their large variety (Benamara et al, 03c). This paper presents an attempt to organize them, to provide a homogeneous model and an implementation of both the reasoning and the natural language generation aspects. First we present related work on the use of inferences in QA systems, and then a description of the current main aspects of WEBCOOP. Then, the advanced relaxation processes and their related NLG issues are presented. The paper ends by suggestions on how to evaluate these types of QA systems.

2. Inferences in QA Systems

In the first generation of QA systems, which were designed to answer NL questions from structured databases, only a few QA tasks were claimed to need inference. In fact, inference complemented the extensional processes of relational database querying (SQL). In today QA systems, where answers can be found in free texts or unstructured data (like the web), inference can enhance Open Domain QA systems capabilities and improve the quality and the accuracy of their answers (Burger et al, 00). This has been illustrated in TREC 2002 QA task where the top system was produced by the LCC team (Moldovan et al, 03) and also by the emergence of the Inference Web (IW) which aims at explaining reasoning paths used to retrieve answers (McGuinness et al, 03). Inference allows enhanced or extended QA services by providing *intelligent* and *cooperative* answers. There are many ways for a query to be answered intelligently, including : (1) providing answer explanations, (2) returning intensional responses and answer summarization, (3) generalizing queries using neighborhood information, (4) comparing answers that have similar questions, (5) realizing information fusion when answers are inferred from multiple data sources, etc. In this section, we deal with (1), (2) and (3) described above. For a general panorama of inference uses for QA tasks, see (Webber et al, 02).

1) Answers Explaining a Query Failure : this situation generally occurs when a user has a false or unclear understanding of what he is asking for. For example, a direct answer to the question: “*Which train should I take to go from Paris to Algiers?*”, should be “*no*”, while a cooperative and informative answer should be : “*there cannot be any train between Paris and Algiers because of the sea*”. Inference is needed to detect in a question false presuppositions or misconceptions that conflict with the system knowledge base (false presuppositions occur with respect to the database contents, misconceptions usually occur with respect to the database semantics). Several techniques were developed for recognizing presupposition failure and for generating answer explanations. Let us cite the work carried out at Maryland (Gaasterland et al, 94), that detects user misconceptions using logical inference and integrity constraints of the database, and the WhyNot tool (Chalupsky et al, 02) which

generates *plausible partial proofs* that approximate the correct proof that answer a query that fails.

2) Query Rewriting Using Neighborhood Information : is often useful to provide neighborhood information to a query. For example, if the question : “*give me chalets in Corsica region?*” has an empty or a too small set of solutions, the query scope can be extended to provide bungalows or country cottages in Corsica instead of just chalets. One way to increase the yield of potential answers is, for example, to find, within concept hierarchies (such as ontologies), a set of most appropriate concepts which are conceptually close to the relaxed concept in the initial query. A number of techniques were developed within the AI community to address the query relaxation problem. For example, (Gaasterland et al, 94) modify the query either to provide more information to the user, or to repair the query if it fails, and (Minock et al, 96) employ, within their COBASE system, an *abstraction/refinement* method for providing related answers to the original query.

3) Intensional Answer Descriptions : a query can be answered by generalization and summarization of the answer set when this set is too large, making answer underlying implications much more clear. For example, the query : “*which students have good marks in computer science?*”, can be answered intelligently in several ways : (1) “*100% of the graduate students, 25% of the senior students..*”, or (2) “*all of the graduate students and the undergraduate students are the following...*”. To realize this task, inference is needed e.g. for cleaning answers when they partly overlap, for determining whether an answer is more specific than another one and for organizing mutually consistent answers. (Gal, 88) uses heuristics that apply to sets of integrity constraints to produce synthetic responses, i.e. by summarizing, sorting explanations or by focusing on a particular one.

In the remainder of this paper, we focus on the query rewriting task (2).

3 WEBCOOP, a Cooperative Question Answering System on the Web

Similarly to standard web search engines, WEBCOOP is not settled within a dialogue perspective and does not include any user model. WEBCOOP is a

comprehensive QA system, going from question processing to natural language answer generation using advanced reasoning procedures. In the 90's, in a number of cooperative answering systems, most of the efforts were concentrated on fundamental reasoning procedures, while in standard QA systems (as those evaluated in the TREC QA track) focus is on language syntax and semantics issues to improve question analysis and answer extraction, however with little reasoning. In most of these latter systems, there is no NL generation. Furthermore, we claim that responses should make explicit in some way the underlying explanation and justification mechanisms that led to the answer. Answer explanation strategy in WEBCOOP is organized as follows: first a short message explains the reasons of a query failure (integrity constraint violation or no solution), then, by means of underspecified NL patterns, the crucial steps of the inference procedures are given. This strategy is inspired from our corpora analysis (Benamara et al, 03c), where answer justifications are brief and accurate. We believe this is sufficient to understand responses. This is why, unlike Inference Web, WEBCOOP does not give to the user all the inference steps such as axioms, derived rules or all activated nodes within the domain ontology during the reasoning process. We think this latter approach, that resembles expert systems of the past decade, is not very cooperative.

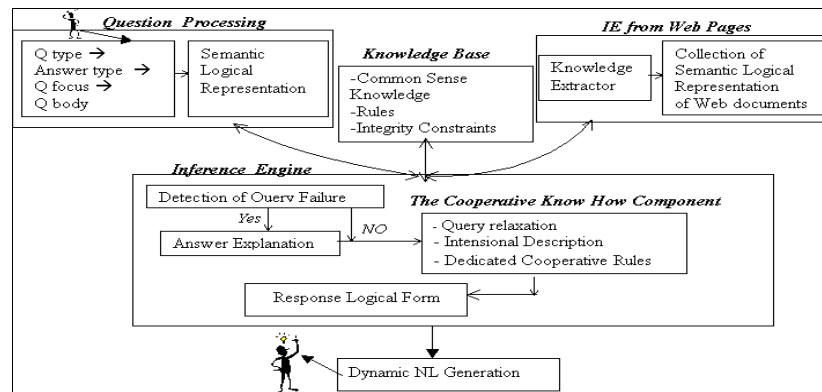


Figure 1 : WEBCOOP Architecture

3.1 WEBCOOP General Architecture

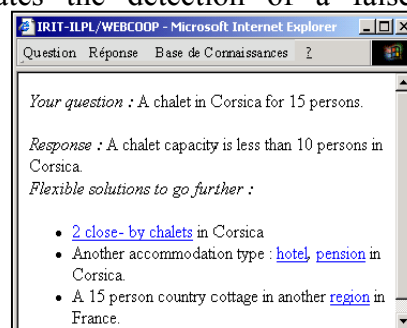
The general architecture of the system is illustrated in figure 1. In WEBCOOP, user questions may range from keywords to comprehensive natural language expressions. Their parse produces a semantic logical representation that includes the question conceptual category, the question focus and the question contents. Our approach requires two, very classical, levels of knowledge representation : general purpose knowledge, specified by hand, and domain knowledge, acquired via knowledge extraction procedures from web pages. Knowledge is represented by means of rules and integrity constraints and by means of an ontology where nodes are concepts with properties. A uniform framework is used, based on a simplified version of the Lexical Conceptual Structure (LCS), (Jackendoff, 90). The *question analysis* and the *knowledge extraction* processes are presented in (Benamara et al, 03b).

3.2 WEBCOOP : a Two Level Structure for Responses

Responses are structured in two parts. The first part contains explanation elements in natural language. It is a first level of cooperativity that reports user misconceptions in relation with the domain knowledge. The second part is the most important and the most original. It reflects the *know-how* of the cooperative system, going beyond the cooperative statements given in the first part. It is based on *intensional description techniques* and on *intelligent relaxation procedures* going beyond classical generalization methods used in AI. This component also includes additional *dedicated cooperative rules* that make a thorough use of the domain ontology and of general knowledge. Responses provided to users are built in web style, by integrating natural language generation (NLG) techniques with hypertext links to produce "dynamic" responses. Hyperlinks are dynamically created at generation time. This leaves up to the user the high-level planning tasks inherent to NLG and improves readability and information access. The standard generation difficulties (lexicalisation, aggregation, argumentation) remain crucial to generate cooperative responses, but their web-style greatly reduces the overall complexity. The *know-how* component also allows for the dynamic determination of those text fragments to be defined as hypertext links, from which the user can get more information. To better characterize our problem, we collected a corpus of question answer pairs in French that reflects different

cooperative behaviors. The example below is extracted from this corpus. For the sake of readability, it is translated into English.

An example : suppose one wishes to rent a 15 person chalet in Corsica and (1) that observations made on the related web pages or (2) that a constraint or a regulation, indicates that the maximum capacity of a chalet in Corsica is 10 persons. The first part of the response relates the detection of a false presupposition or the violation of an integrity constraint for respectively cases (1) and (2) above. This entails the production of the following message, generated by a process that evaluates the question logical formula against the knowledge base: *A chalet capacity is less than 10 persons in Corsica.*



In a second step, the *know-how* component of the cooperative system generates a set of flexible solutions as shown in the figure above, since the first part of the response is informative but not really productive for the user. The three flexible solutions proposed emerge from *know-how* cooperative rules based on *relaxation procedures* designed to be minimal and conceptually relevant. The first flexible solution is based on a cardinality relaxation, while in the last two solutions, relaxation operates gradually on concepts such as the type of accommodation (hotel or pension) or the region (possibly a close-by region, with similar characteristics), via the domain model and the ontology. Dynamically created links are underlined. The user can then, at will, get more precise information, dynamically generated from the data base of indexed web pages.

4. Query Relaxation in WEBCOOP

The cooperative *know how* component has potentially a large number of capabilities and facets. We focus here on a basic, but frequently used, reasoning schema : query rewriting that pairs accurate relaxation techniques with ontological knowledge. Instead of considering a direct ancestor for the relaxed element in the query, our contribution focuses on the taking into account of more subtle forms of constraint relaxation, such as considering sister nodes under constraints or more remote nodes sharing a significant number of

properties with the concept to be relaxed. The reasoning aspects of the relaxation and the related main natural language generation aspects are presented here.

4.1 Prerequisites to the Relaxation Process

a. The knowledge base elaboration: we designed a hand crafted ontology of the tourism domain, where nodes are concepts, decorated by means of attributes which describe concept properties or any other relevant information. Since descriptions are hierarchically organized, properties are inherited, if there is no conflict. We define over this ontology a metrics $M(C, C')$ that expresses the ontological distance between two concepts C and C' . Considering the form of our ontology, we are, at the moment, experimenting different kinds of metrics. The simplest one is to consider the sum of the distance in the ontology between the two concepts C and C' in terms of the number of arcs traversed $NbArc$ and the inverse proportion of shared properties, w.r.t the total number of properties on these nodes. This metrics can be written as :

$$M(C, C') = NbArc(C, C') + \frac{Card(prop(C) \cup prop(C'))}{Card(prop(C) \cap prop(C'))}$$

b. The misconception report : NL query parsing produces a logical formula of the following general form : $Q = Q_1(X_{11}, \dots, X_{1m}) \wedge \dots \wedge Q_n(X_{n1}, \dots, X_{nl})$ where $\bar{X}_i = (X_{i1}, \dots, X_{ik})$ is the tuple of arguments of the sub query Q_i . Arguments can be variables, constants or semantic fields of the lexical conceptual structure. The sub queries $Q_i(\bar{X}_i)$ are positive literals of the knowledge base. We have now to check if the query is coherent with the knowledge base. If it is not, either an integrity constraint has been violated by the user (misconception), or the query has an empty set of solutions (false presupposition). In both cases, the system has to identify the sub queries and the set of rules and integrity constraints which are at the origin of the query failure, in order to be able to provide explanations. To reach this goal, we use a complete and a sound method, based on the *positive hyper resolution*, defined in (Chang et al, 73), that detects all possible reasons of a query failure, focusing on the literals of the query.

4.2 The Relaxation Schema

The relaxation mechanisms of the know how component proceed gradually on each of the sub queries at the origin of the failure. They are modeled as a rewriting system, managed by a supervisor which includes an iterative process running till a flexible solution is found that leads to a non-empty and coherent solution. The relaxation strategy returns the most immediate relaxations once the original query is exhausted and leaves up to the user to select the relaxation direction. This can be very useful to correct user misconceptions and to help him to navigate the database of extracted web pages.

The relaxation technique is based on a generic *proximity relation* that refers to inherent properties of objects, a conceptual ontology and lexical semantics relations. It is applied to different ontological or technical domains such as, fares, capacity (example in section 3.2), type of transportation... The notion of proximity is implemented by the predicate $near(CD, V, Y, Result)$ where CD is a type in a conceptual domain, V is the resource on which the relaxation operates and Y is a resource resulting from the relaxation, of the same type than V . $Result$ contains the sorted set of results according to the criteria associated with the conceptual category. The general relaxation schema is modeled by means of rewriting rules. Its general form is : $Relax(Type, QPredicates-To-Relax, Rest-QPredicates, Results) \rightarrow NewQuery(Results)$, where $Type$ realizes several generic types of relaxations as shown below, $QPredicates-To-Relax$ is the conjunction of sub queries Q_i on which the relaxation operates (the reason of the query failure), $Rest-Qpredicates$ is the remainder of the query and $Results$ contains the set of coherent and flexible solutions resulting from the relaxation process, after evaluation of the relaxed query against the knowledge base. Among different cases of query relaxation we have investigated, here are three which are very frequently used :

a. Relaxation based on cardinality addresses the case of a question where the quantity of the resource does not meet the cardinality constraints specified in the question. The query logical formula is : $Q = C(X) \wedge F(X, NB) \wedge R$, where C is a concept with X the focused resource, $F(X, NB)$ is the conjunction of predicates of Q that contains occurrences of X , NB , the desired quantity of X , and R is the remainder of the query. The relaxation duplicates the sequence $C(X) \wedge F(X, NB)$ N times to introduce several instances of the object X till the desired quantity

(NB) of X is reached. The relaxation rule is (for $N=2$) : $Relax(Cardinality, C(X) \wedge F(X, NB), R, Results) \rightarrow C(X1) \wedge C(X2) \wedge F(X1, NB1) \wedge F(X2, NB2) \wedge X1 \neq X2 \wedge (NB1 + NB2 \geq NB) \wedge R \wedge near(CD, X1, X2, Results)$.

The conceptual domain CD in the proximity relation $near$ can be, for example, *localization* (geographical proximity) if the type physical localization can be derived from the ontological type of X ; or *temporal* for a temporal duration or the simultaneity of events. Let us consider again the example of section 3.2, if the maximum chalet capacity in Corsica is 10 persons, the query : $Q = chalet(X) \wedge region(corsica) \wedge in(place, X, corsica) \wedge capacity(X, 15)$ is relaxed into : $chalet(X1) \wedge in(place, X1, corsica) \wedge capacity(X1, NB1) \wedge chalet(X2) \wedge in(place, X2, corsica) \wedge capacity(X2, NB2) \wedge region(corsica) \wedge X1 \neq X2 \wedge (NB1 + NB2 \geq 15) \wedge near(localization, X1, X2, Results)$ which searches for two close-by chalets in the Corsica region for 15 persons. To evaluate proximity of the two chalets, $near$ consults the set of facts *chalets* in the database of extracted web pages.

b. Relaxation on the ontological type of the focus provides minimal and gradual relaxation on the ontological type of the question focus which is generally at the origin of the failure. The question general form is : $Q = C(X) \wedge R$, where C is the type of the resource denoted by the variable X and R the remainder of the query. The general relaxation schema is : $relaxation(Ono-Type, C(X), R, Result) \rightarrow NewQuery(Result)$, where, depending on *Ono-Type*, we have one of the following two cases:

Case b.1 : the set $S = \{C'_1, \dots, C'_n\}$ of sisters of the relaxed concept C can be used to meet the question requirements. This is the case in the example in section 3.2, where a hotel or a pension are proposed instead of a chalet, which has a too small capacity. This case occurs mainly when the set S is not empty and when there is a subset $S' \subseteq S$ where the distance computed by $M(C, C')$ is small for every sister concept C' in S' (section 4.1). The relaxation schema is the following : $\forall (C' \in S') Relaxation(Sister-Node, C(X), R, C'(X)) \rightarrow C'(X) \wedge R$.

Case b.2 : if the set S of sister nodes of the concept C is empty or if the distance computed by $M(C, C')$ is large, for every C' in S , then a mother node $Mnode$ of C is considered . Two cases may occur :

(i). if the ontological distance $M(C, Mnode)$ between the relaxed concept C and its direct ancestor $Mnode$ is small, the relaxation process is a **generalization** and the general rewriting rule is : $Relaxation(Generalization, C(X), R, Mnode(X)) \rightarrow Mnode(X) \wedge R$.

(ii). if the ontological distance $M(C, Mnode)$ between the relaxed concept C and its direct ancestor $Mnode$ is large then the relaxation process searches for a more remote concept C' in the ontology which shares a maximum of properties with C . Then, when searching for responses in the indexed web pages, the system looks for texts satisfying $C'(X) \wedge R$ and containing some or possibly all the properties of C not present in C' . This is implemented by the predicate $similar(C, C', prop(C-C'))$. The general rewriting rule is : $Relaxation(Similarity, C(X), R, C'(X) \wedge prop(C-C')) \rightarrow C'(X) \wedge R \wedge similar(C, C', prop(C-C'))$. For example, if one is looking for *village de vacances en Bretagne* (holiday village in Brittany), and that there is none in Brittany, a *hotel-restaurant* can be proposed (subset of the properties of holiday village : accommodation and meals) with sport activities, children day-care, etc. close-by, i.e. specified in the hotel restaurant web pages, but not attached to the hotel-restaurant resource.

c. Relaxation on constants : may occur when the origin of the query failure is a constant. The query logical form is $Q=C(cst) \wedge R$, where cst is a constant of type C . The relaxation process replaces in Q the constant cst by a variable Var which denotes the same kind of resource with a set of potential values close to cst . If the number of solutions is small, they can be enumerated, otherwise a hyperlink is generated. The relaxation schema is then : $Relaxation(Constant, C(cst), R, Result) \rightarrow C(Var) \wedge near(CD, cst, Var, Result) \wedge R$, where CD can be e.g. a *localization* if the type physical localization can be derived from the ontological type of cst , or *temporal* (flights between 8pm and 9pm are proposed if there is no flight at 8:30pm.). For example, if there is no commercial airport in Albi, the query : $Q=flight(ID, paris, albi) \wedge city(albi) \wedge city(paris)$, is relaxed into : $Q=flight(ID, paris, Var) \wedge city(Var) \wedge city(paris) \wedge near(place, albi, Var, Results)$ i.e. searching for the nearest cities to Albi served from Paris, ordered by increasing distance from Albi.

This relaxation can also occur when the question focus is a free variable which is constrained in a relational predicate by means of a constant. The query logical

representation is : $Q=C(X) \wedge P(X, \bar{Y}) \wedge R$, where P is a predicate that constrains the variable X of type C and \bar{Y} a tuple of variables that exclude X. We relax X w.r.t. the constraints that hold on it, namely $P(X, \bar{Y})$. For example the query : “give me country cottages in the north of France” can be relaxed into : “give me country cottages in the center of France”, assuming that it is the closest region that has solutions.

4.3 The Relaxation Control Strategies

A supervisor manages the calls to the different relaxation procedures. Each relaxation type runs iteratively till a solution is found. This avoids the problem of having to guess the user’s intent. The relaxation process returns all the possible answers related to the detected conflict without sorting them. The use of the proximity relation *near* and the ontological metrics *M* guarantee that the nearest and the most appropriate set of solutions that meet the question requirements is found. Then, important display strategy problems have to be solved, such as : choosing the relevant display order of the relaxed answers and presenting to the user a manageable number of flexible responses. This can be done, for example, by adding a priori preferences to query literals. Different directions are under investigation considering the following criteria : (1) syntactic criteria e.g. the inverse reading order of the question: *give me (chalets)¹ in (Corsica region)² (for 15 persons)³*, we can choose to relax first 3, then 2 and finally 1, or (2) pragmatic criteria if we consider that the most precise information is the most reliable. In the above example, we would then relax first 1, then 2 and finally 3.

4.4 Natural Language Responses Involving Relaxation

NLG techniques can make explicit in some way the explanation and the justification mechanisms that led to the answer. In (Benamara et al. 03a), we show that using hypertext links avoids, to a large extent, to manage the general structure of the response, and leaves open navigation possibilities for the user. For each type of relaxation, a general and somewhat underspecified natural language pattern can be defined that translates the relaxation in accessible terms. Underspecified elements depend on the question, on local semantic factors and on the type of solution elaborated. Their generation relies on ontological knowledge, general linguistic knowledge and lexicalisation and aggregation

functions. Patterns have been defined from a number of question-response pairs found in the tourist domain on the web. Responses have been normalized without losing their accuracy in order to get stereotyped response forms usable in NL generation frameworks. A large portion of the pattern is presented as an hyperlink to the user as illustrated in the example of section 3.2. Let $C(X)$ the predicate being relaxed in the query and let $lexicalisation(C)$ the function that generates the adequate NL fragment expressing the concept C . The set of possible lexicalisations for a given concept is defined in the domain ontology. The figure below exemplifies some of these generation patterns for French, for each of the three types of relaxation introduced in section 4.2. Underlined portions are generated as hyperlinks.

<p>*Cardinality $\rightarrow N lexicalisation(C(X), plural) proximity(C(X)) R.$ Example : <u>2 séjours consécutifs d'une semaine en Tunisie</u> (<u>Two consecutive one-week stays in Tunisia</u>).</p> <p>*Ontological Type</p> <p>- Sister node $\rightarrow Un(e) autre lexicalisation(Mnode(X)) : lexicalisation(sister(C)) R.$ Example : <u>Un autre type d'hébergement touristique : hôtels, pensions, en Corse pour 15 personnes.</u> (another type of touristic accommodation: <u>hotels, pensions</u>, in Corsica for 15 persons).</p> <p>- Generalisation $\rightarrow Seulement lexicalisation(Mnode(X)) R.$</p> <p>- Similarity $\rightarrow lexicalisation(C'), R et les éléments suivants à proximité: lexicalisation(prop(C)).$</p> <p>*Constant (CST) $\rightarrow Un(e) autre lexicalisation(C) proche de CST, R. If CST is the question focus$</p>
--

Figure 2 : Basic Generation Patterns for the Relaxation Schema

5. Conclusion and Future Directions

In this paper, we propose a logic based QA system that integrates knowledge representation and advanced reasoning procedures to generate intelligent responses to NL queries on the web. We focus on a specific reasoning procedure, that pairs accurate relaxation techniques with ontological knowledge. We propose several kinds of minimally and conceptually accurate relaxations and show that, for each type of relaxation, a general and underspecified natural language pattern can be defined that translates the relaxation in accessible terms.

The relaxation mechanisms of the know how component are implemented in Prolog via meta-interpretation. We use a *case based* implementation method : first, we collected a set of question examples then ranked them by increasing order of complexity (such as complex WH questions (Diekema et al, 03)). We concentrated our implementation around these examples. Then, we evaluated the

results at two levels: (1) the quality of the services offered to a user by comparing our results to human answers in Frequently Asked Question corpora and (2) the re-usability to other corpus examples. We evaluated each kind of relaxation separately first and the overall functioning of the system via the supervisor. In the future, we plan to use what Barr and Klavans (Barr et al, 01) call *component performance evaluation* which consists of assessing the performance of system components and determining their impact on the overall system performance.

This project has obviously several future directions among which we plan to : (1) develop new cooperative know-how strategies and their related logical expressions and implementations, (2) specify different strategies for generating intensional responses in the know how component, for example when the number of direct responses is very large, and (3) investigate advanced natural generation issues in order to elaborate specific explanation schemas for each kind of cooperative responses that WEBCOOP proposes.

6. References

- Barr, V, and Klavans J. 2001. Verification and Validation of Language Processing Systems: Is It Evaluation?, ACL Workshop on Evaluation Methodologies for Language and Dialogue Systems, pp : 34 - 40.
- Benamara, F, and Saint Dizier, P. 2003a. Dynamic Generation of Cooperative Natural Language Responses, Ninth European workshop on Natural Language Generation. EACL, Budapest, Hungary.
- Benamara, F, and Saint Dizier, P. 2003b. WEBCOOP: A Cooperative Question-Answering System on the Web, EACL project notes, Budapest, Hungary.
- Benamara, F and Saint Dizier, P. 2003c. Construction de réponses coopératives : du corpus à la modélisation informatique. Revue Québécoise de Linguistique RQL, special issue : TALN, corpus et Web, Forthcoming.
- Burger J, Cardie C, Chaudhi V, and al. 2000. Issues ,Tasks and Program Structures to Roadmap Research in Question Answering. Technical report, NIST.

- Chalupsky, H and Russ, T.A. 2002. WhyNot: Debugging Failed Queries in Large Knowledge Bases. In Proceedings of the Fourteenth Innovative Applications of Artificial Intelligence Conference, pp : 870-877, Menlo Park, AAAI Press.
- Chang, C.L, and Lee, R. C.T. 1973. Symbolic Logic and Mechanical Theorem Proving. Academic Press.
- Dale, R, Oberlander, J, Milosavljevic, M, Knott, A. 1998. Integrating Natural Language Generation and Hypertext to Produce Dynamic Documents. *Interacting with Computers* 11(2), pp. 109-135.
- Diekema, A, Yilmazel, O, Chen, J, Harwell, S, He, L and Liddy, E. 2003. Finding Answers to Complex Questions. Chapter 5 in this volume.
- Gaasterland, T, Godfrey, P, Minker, J. 1994. An Overview of Cooperative Answering. *Papers in Non-standard Queries and Non-standard Answers*, in series *Studies in Logic and Computation*, Clarendon Press, Oxford.
- Gal, A. 1988. Cooperative Responses in Deductive Databases, Phd Thesis, University of Maryland, Department of Computer Science.
- Jackendoff, R. 1990. *Semantic Structures*. MIT Press, Cambridge M.A.
- McGuinness, D and Pinheiro da Silva, P. 2003. Trusting Answers on the Web. Chapter 22 in this volume.
- Minock, M, Chu, W, Yang, H, Chiang, K, Chow, G and Larson, C. 1996. CoBase : A Scalable and Extensible Cooperative Information System. *Journal of Intelligent Information Systems*, volume 6, number 2/3, pp : 223-259.
- Moldovan, D, et al. 2003, LCC Tools for Question Answering, to appear in the Proceedings of TREC-11, NIST.
- Webber, B, Gardent, C, Bos, J. 2002. Position Statement : Inference in Question Answering. LREC.

