Use of Default Reasoning for Disambiguation under Question Answering

Boris Galitsky

School of Computer Science and Information Systems
Birkbeck College, University of London
Malet Street, London WC1E 7HX, UK
galitsky@dcs.bbk.ac.uk

Abstract

We develop the default logic for pragmatic analysis of natural language queries. Ambiguous sentences are considered so that each set of meanings is assigned an extension of default system. The approach is applied to natural language question answering, where even a correct semantic representation needs to be modified in accordance to the set of default rules to better match a knowledge domain.

Introduction

Although recent years have seen an explosion in the question answering (Q/A) technology, there is still lack of systems satisfactorily providing answers in the domains which are logically complex, poorly structured and hardly formalized. To approach such domains, designers of every Q/A system must find a compromise between a *full-featured natural language processing* (NLP) *system* oriented for complex domains, *advanced reasoning* for semantic processing and knowledge representation, and *shallow processing*, leveraging the issues of performance and automation of domain creation (Creary and Pollard 1985, Romacker et al 1999).

In spite of the high number of commercial Q/A systems functioning in a wide spectrum of domains, it is still unclear what NLP component is essential to achieve satisfactory answer accuracy, high performance and efficient domain preparation. Clearly, increase of the (logical) complexity of queries to be understood can be accomplished by involving more sophisticated reasoning. Let us enumerate the potential components of an abstract Q/A system, considering them from the practical reasoning viewpoint. The reader should bear in mind that a particular implementation is based on a selected subset of these components:

1) Rule-based system for *morphological and syntactic* analysis that reveals the interconnections between the words and their roles.
2) Rule-based *semantic* analysis that obtains the query representation after the maximum possible number of rules have been applied.
3) **Inference-based *pragmatic* analysis, transforming query representation in accordance with the answer knowledge base.**
4) Answer knowledge base, obeying the efficient reasoning-based properties of consistency, completeness, easy update and restructuring.
5) Reasoning-accented *information extraction*.
6) Reasoning-accented textual annotation
7) Interactive domain development tools

The third component, **pragmatic analysis**, usually attracts the least attention with respect to the other items above, probably because it is based on heuristics rather than on a firm linguistic or logical base. Pragmatic component may implement the verification of compatibility between the formal representation of a query and the knowledge base. In particular, our study of generality control (Galitsky 2003) suggested the feedback mechanism of sorting out the invalid hypotheses for formal representation of an input question. Therefore, pragmatic analysis filters out the hypotheses of syntactic and semantic translation of input query, inconsistent with domain representation. This approach is especially viable under fully formalized domains.

In this paper we introduce the technique which allows developers of Q/A systems to use the domain-specific pragmatic information. We believe that the idea of using nonmonotonic reasoning for correction of semantic representation of an input query under Q/A was first suggested in (Ourioupina and Galitsky 2001). The aim is not to achieve a semantic representation which is more precise, but to make the given representation fit better into the domain. This is achieved by the application of the default logic machinery. Default rules can naturally appear in the domain representation itself.

Note that the use of default rules for the transformation of translation formula is independent of the degree of knowledge formalization: it helps from fully to weakly formalized domains. However, one needs to distinguish the default rules for query transformation and for domain representation: the former are linked to NL, and the latter are supposed to be language-independent. Though the default reasoning technique is applicable to a variety of semantic analysis approaches, the particular set of default rules is domain-specific.

Default reasoning seems to be well suited to represent the semantic rules which process the ambiguous terms. In a horizontal domain, a term may have a set of meanings such that each of them is important and needs separate consideration. For a vertical (narrow) domain, an ambiguous term usually has one common meaning and multiple infrequent meanings which still have to be formally represented. The system may assume the default meaning of this term unless it is inconsistent to do so. This inconsistency can be determined based on the occurrence of some words in a sentence, or some terms in the translation formula (formal representation of an input NL question). We refer the reader to (Hirst 1988, Creary and Pollard 1985) for discussions of the disambiguation problem.

In this study we evaluate the application of default reasoning to the domain based on semantic headers (Galitsky 2003). However, the proposed technique is representation-independent and can be combined with any approach to semantic analysis, achieving a certain level of formalization (Sondheimer et al 1984, Ciravegna and Lavelli 1999). Note that nonmonotonic reasoning is suggested here for query processing and not for knowledge representation.

From the perspective of nonmonotonic reasoning, linking the semantic representation of a query with that of answers, additional (lexical) data in a query may switch the resultant pattern (answer). Such nonmonotonicity would be considered as a non-desired property in the standard (statistical or structural) pattern recognition settings: discovering an additional feature of an input stimulus is not supposed to alter the resultant pattern. This is not the case for the pattern recognition under Q/A: an occurrence of an additional word which may seem insignificant being stand-alone, may cause a totally different answer for a query. We believe explicit nonmonotonic reasoning machinery is an appropriate means to handle this phenomenon of NL-based recognition systems.

Basics of Default Reasoning for Question Answering

An abstract default logic (as proposed by (Reiter 1980)) distinguishes between two kinds of knowledge, usual predicate logic formulas (axioms, facts) and "rules of thumb" (defaults). Default theory (Brewka et al 1995, Bochman 2001) includes a set of facts which represent certain, but usually incomplete, information about the world (query, in our case); and a set of defaults which cause plausible but not necessarily true conclusions (for example, because of ambiguity). In the case of textual knowledge representation, some of these conclusions have to be revised when additional context information becomes available.

Division of knowledge into the fixed (facts) and flexible (rules) components may serve as a good model of a textual message or an arbitrary portion of text, taken separately from its context. Some statements are context-independent and have a single meaning, other statements have context-dependent meanings and can be determined after the context is established, and a third group of statements is ambiguous in as exact a context as possible. As we have said, in this work we use the system of default rules to modify the formal representation of a textual answer for the sole purpose of question answering and not to better represent the text meaning itself (or knowledge of an arbitrary nature). The latter problem is the subject of future studies.

Let us consider the traditional example quoted in the literature on nonmonotonic reasoning:

$$\frac{bird(X):\ fly(X)}{fly(X)}$$

One reads it as *If X is a bird and it is consistent to assume that X flies, then conclude that X flies*. As a Q/A matching rule default, it reads as follows *If the query is about the bird, and it is consistent to assume that there can be a word in this query with the meaning of flying, then conclude that the query is about flying*. If nothing contradictory can be derived from the other words of the query, it is natural to assume that this query is about the *flying of a bird*.

As a traditional default, we obtain the assumption *Usually (typically) birds fly*. Given the information that *Tweety is a bird* (in accordance with the history of nonmonotonic reasoning) we may conclude that he flies. But if we learn later that he cannot fly, for example, because he is a penguin, then the default becomes inapplicable.

The nonmonotonic reasoning technique (see e.g. (Bochman 2001)) helps us to provide the proper Q/A in this situation. Imagine that we have a set of documents answering questions about birds. Let us imagine there is a general answer, which presents information that *birds fly*, and which contains common remarks. Besides, there is also a specific answer, stating that *there is a bird Tweety, who is a penguin, lives at the South Pole, swims, but does not fly*.

If there is no processing of the translation formula, then the explicit restriction for the semantic representation will look like the following. The possible questions are *Do birds fly?, Tell me about birds, Which animals fly?, Do seagulls fly?, What do you know about eagle birds?, Is Tweety a bird?, Does Tweety fly?, Is Tweety a penguin?*; they will cause the same answer that birds fly, unless we mention the *bird Tweety*.

bird(X), not(X=Tweety) → answer(birds fly).

fly(X), not(X=Tweety) → answer(birds fly).

bird(Tweety) → *answer*(*there is a bird Tweety, which is a penguin, lives at the South Pole, swims, but does not fly*).

fly(Tweety) → *answer(there is a bird Tweety, which is a penguin, lives at the South Pole, swims, but does not fly).*

The problem with this approach is that it requires explicit enumeration of constraints in knowledge base. Each exceptional object has to be enumerated in the representation, which includes the properties of normal objects. When default rules come into play to modify the translation this complication is avoided. The first answer does not have to contain explicit constraints that *X is not Tweety* or some other object; it will be reached if it is consistent to assume that *fly(X)*. The latter is verified using the default rule above and the clause that *Tweety does not fly*.

Here is a formal definition of default rule, as proposed in (Antoniou 1997). A default δ has the form:

$$\delta = \frac{\varphi :\ \psi_{1,\dots,}\ \psi_n}{\chi}$$

where φ, $\psi_{1,\dots,}\psi_n$,χ are closed predicate formula and $n>0$. The formula φ is called the prerequisite, $\psi_{1,\dots,}\psi_n$ the justification, and χ the consequent of δ (*pre*(δ), *just*(δ) and *cons*(δ) correspondingly). A default δ is applicable to a deductively closed set of formulae E iff $\varphi \in E$ and $\neg\psi_1 \notin E$, $\dots, \neg\psi_n \notin E$.

Note that it is inefficient to encode the rules for typical and non-typical situations on the same level of their potential application. Instead, all abnormalities should be merged and linked with the negation of justification of the default rule. The prerequisite of this rule should include the typical conditions.

Operational Semantics

In an abstract rule-based system, the inference (or recognition) result depends on the order of rule applications. The operational semantics of default logic come into play to handle the possible order of application of the conflicting default rules. In this Section we propose an informal description of operational semantics for default reasoning. Formal definitions, theorems and proofs can be found, for example, in (Antoniou, 1997).

The main goal of applying default rules is to make all the possible conclusions from the given set of facts. If we apply only one default, we can simply add its consequent to our knowledge base. The situation becomes more complicated if we have a set of defaults because, for example, the rules can have consequents contradicting each other or, a consequent of one rule can contradict the justification of another one. In order to provide an accurate solution we have to introduce the notion of *extensions***:** current knowledge bases, satisfying some specific conditions.

Suppose D is a set of defaults and W is a set of facts (our initial knowledge base). Let Δ be an ordered subset of D without multiple occurrences (it is useless to apply the default twice because it would add no information). We denote a deductive closure (in terms of classical logic) of Δ by $In(\Delta)$: $W \cup \{cons(\delta) \mid \delta \in \Delta\}$. We also denote by $Out(\Delta)$ the set $\{\neg\psi \mid \psi \in just(\delta), \delta \in \Delta\}$. We call $\Delta = \{\delta_0, \delta_1, \dots\}$a process iff for every k δ_k is applicable to $In(\Delta_k)$, where Δ_k is the initial part of Δ of the length k.

Given a process Δ, we can determine whether it is successful and closed. A process Δ is called successful iff $In(\Delta) \cap Out(\Delta) = \varnothing$. A process Δ is called closed if Δ already contains all the defaults from D, applicable to $In(\Delta)$.

Now we can define extensions. A set of formulae $E \supset W$ is an extension of the default theory $<D, W>$ iff there is some process Δ so that it is successful, closed, and $E=In(\Delta)$.

The following example (life insurance Q/A domain) shows that there can be multiple extensions for one set of facts and default rules:

$$\delta_1 \quad \frac{dangerous\_job(X):\ insure\_life(X)}{insure\_life(X)}$$

$$\delta_2 \quad \frac{young(X):\ not\ \ insure\_life(X)}{not\ insure\_life(X)}$$

These rules explain that people with dangerous jobs usually insure their lives and young people normally do not. This is knowledge required to answer the question *Am I advised to have life insurance as a teenager?* Let us suppose that we want to conclude something about a young man who has a dangerous job: *W={dangerous_job(X), young(X)}.* After the application of each default, the other one becomes inapplicable. So, both $\{\delta_1\}$and$\{\delta_1\}$are closed and successful processes. Thus, both *{dangerous_job(X), young(X), insure_life(X)}* and *{dangerous_job(X), young(X), not(insure_life(X))}* are the extensions.

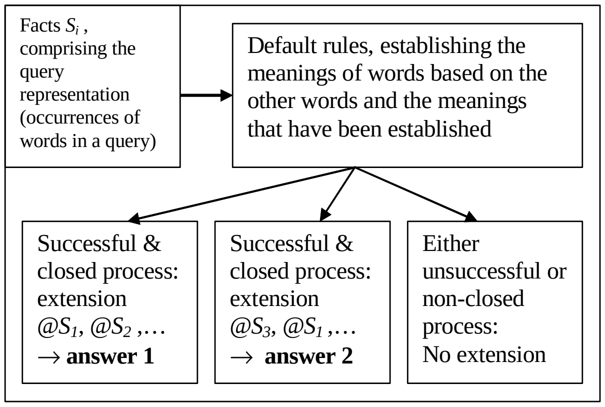
Pragmatic Analysis with the Help of Default Rules

Suppose *S* is a semantic representation of a query. Our intention is to transform *S* into another well-formed semantic representation, *@S* (about *S*), which fits our narrow domain better because it takes into account default domain knowledge. Note that the latter is not necessarily explicitly mentioned in a NL query, so the transformation above is required. To perform it, the following algorithm is proposed: using a set of facts like *word X is used in S* and *word X is not used in S* as an initial knowledge base, we can apply default rules (created manually by knowledge engineers) to obtain all the possible extensions. These extensions contain facts about elements of *@S* (as well as initial statements). After doing that we can use the extensions to build up the @*S* representation. It includes the *n*-conjunction of initial facts (word occurrences in a query X_i) and their specific meanings @X_j : $@S = \underset{i,j<n}{\&} (X_i, @X_j)$

Note that *S* (and not @*S*) is indeed the most precise representation of the query meaning, taken separately. However, *S* needs to be transformed to point to the desired answer with higher accuracy. This transformation is intended to eliminate the least important entities so as not to interfere with the most important ones, as well as to add the implicitly assumed elements.

There are two possible ways to use default systems to modify semantic representations:

- Application of defaults in the fixed order. This can be used when there are no conflicts in the consequents of the default rules.
- Building extensions for conflicting defaults. We employ the operational semantics of default logic in more complex situation, for example, when we have multiple ambiguous terms in a query (Fig. 1).



Facts S_i, comprising the query representation (occurrences of words in a query)
Default rules, establishing the meanings of words based on the other words and the meanings that have been established
Successful & closed process: extension $@S_1, @S_2,\dots$ → **answer 1**
Successful & closed process: extension $@S_3, @S_1,\dots$ → **answer 2**
Either unsuccessful or non-closed process: No extension

Fig.1 Question answering architecture which provides multiple answers as a result of default reasoning

If we do not want to modify initial representation at all (S=@S), we can apply trivial defaults to each element of S:

$$\frac{X \text{ is used in } S : X \text{ is used in } @S}{X \text{ is used in } @S}$$

All the facts in our knowledge base are about the element occurrence: it is either used in the representation or not. In our rules we can write *X* instead of *X is used in S* and @*X* instead of *X is used in @S*, not causing any confusion. Sometimes we will speak about entities *X* and *Y* connected semantically or syntactically. In that case we would write *X(Y)*, which means that *X is used in S*, *Y is used in S* and *X and Y are connected by semantic or syntactic link*. In this form trivial rules look like:

$$\frac{X : @X}{@X}$$

We can use these defaults not only when we want to have *S* unmodified. In fact, we should apply trivial rules to every representation. Elements which are not affected by nontrivial defaults are simply moved from *S* to @S. As far as other elements are concerned, trivial and nontrivial rules can interact with each other, leading to specific extensions.

Processing ambiguous terms

Ambiguous elements correspond to multiple default rules. Let us consider an example from an insurance domain. The word *company* is ambiguous: it can refer either to *insurance company* or to *company where the customer works*. In default form it looks like:

$$\delta:\ \frac{company():@insurance\_company(), \text{not } @place\_of\_work}{@insurance\_company()}$$

$$\varepsilon:\ \frac{company():\text{not } @insurance\_company(),\ @place\_of\_work}{@place\_of\_work}$$

If we have no other entities in the query which can help us to make a decision on what *company* is meant, both rules (δ and ε) can lead to an extension. As a result we have two representations $@S_\delta$ and $@S_\varepsilon$, and two respective answers.

However, if the query contains some words incompatible with one of the meanings proposed, then one of the rules does not lead to an extension. For example, if the query is about companies ratings (*What is the best company?*) then it is about *insurance_company* rather than about *place_of_work*. Note that this rule of thumb holds for the narrow domain on insurance. If a person looks for a job-related domain, the Q/A system's decision should be the opposite one. And in the everything-about-the-world domain we cannot create such rules at all. As a default rule, it looks like the following:

$$\alpha:\ \frac{company(rating):\ @insurance\_company()}{@insurance\_company(rating)}$$

Consequent of α contradicts a justification of ε. That is why there can be no extension created by means of ε: ε is inapplicable after α, α is still applicable after ε, but it makes the process unsuccessful.

If the query contains no information which can help us to perform the query disambiguation, default rules lead to multiple extensions, corresponding to multiple meanings of ambiguous elements. If the query contains some specific concepts, our rules lead to single extension and the system proposes a single answer. We believe that the optimal solution for a Q/A system is to provide multiple answers in case of multiple extensions (i.e. in case there are no words indicating the particular meaning).

Default rules can help us to add new elements to semantic representation, using the basics of commonsense knowledge. Frequently, it is necessary to insert an entity which links the specific attribute with the more general concept occurring in a query. For example, if the query contains word *school* it is likely to be about education. So *Can I deduct my school expenses?* should be interpreted as *deduct(expense(education(school())))*. We propose the following default rule:

$$\frac{school():\ @education()}{@education(school())}$$

This rule should be accompanied by the clause presenting situations of justification inconsistency. If we have a query *Can I deduct my donation to a catholic school?*, it is about a donation rather than about education. The following clause provides the proper solution:

@education :- deduct(Attribute),
not(Attribute =expense).

The rule above expresses the fact that if there are attributes of *deduct* in a query, then the query is likely about these attributes (*donation*, *construction*, *moving*, etc.). If *Attribute* is rather general (*expense*), then the clause fails and justification stays consistent.

Let us consider another example when we add an entity (*donation*) to the query representation to link the specific word (*church*) to a general concept (*deduct*): *Can I deduct what I gave to church?* We have similar default rule:

$$\frac{church():\ @donation()}{@donation(church)}$$

Furthermore, what would happen if we have more than one specific word potentially connected with the general concept, as in a query *Can I deduct my church school expenses?* Occurrence of multiple attributes may require analysis of conflicting defaults (operational semantics). If the justification failure clause for *@donation* is similar to that proposed above, we have two extensions for this query - {…, *@donation*} and {…,*@education*}. But if we consider *education* to be more general than *donation*, the justification failure clause looks like

@donation:- deduct(Attribute),not(Attribute=expense),
not(Attribute=education),

and the only possible extension is *{…, @donation}.*

The default technique indeed brings in significant advantages in the processing of poorly structured (NL) knowledge representation. If we do not want to use default technique for our *school* example then, either we are not able to substitute *school()* in the formula at all or, we have to use *deduct(expense(school()))* representation. In the first case we lose important information and obtain a wrong answer. In the second case we have to provide the possibility to substitute *university, college, institute*, and other terms in *expense()* as well. As a result the domain becomes less structured and query processing loses efficiency.

As we have seen, default rules can help us to improve domain hierarchy. It affects not only the performance but also the quality of query processing. Let us imagine that we have no information about *school expenses*. Instead, we know how to treat *educational expenses* in general. If the system cannot connect *school* with *education*, users get wrong answers because the formula *deduct(expense(school()))* is interpreted literally.

Semantic representation can contain more or less specific entities. The most general parts can lead to vague answers. For example, the query *How to file time extension for my tax return?* has a representation *tax(return)&extension(file)*. It leads to two answers: about *tax returns* in general and about *extension of time for filing*. It is obvious that the first part is redundant and must be eliminated. In fact, in *tax* domain we can interpret *extension* only as *extension of time to file*, and not for example, as *extension of computer file*. Therefore, *tax(return)* can be deleted from the formula without loss of information. We first present the naïve stand-alone rule:

$$\frac{extension(time):\ tax}{@extension()}$$

This rule can be read as follows: if a query mentions *extension of time* and it is consistent to assume that it is about *tax*, then the query is indeed only about *extension*. Although this rule follows our intuition better, it is not appropriate for interaction with other rules, because it does not actually eliminate *tax* from the current knowledge base. Therefore, we suggest the following rule instead:

$$\delta:\ \frac{extension():\ not\ @tax()}{not\ @tax}$$

It is read as follows: if a query contains the word *extension,* then *tax* should be eliminated. Note that the elimination of redundant elements is connected with the disambiguation problem. If we have no idea about the topic of the query, both meanings (*filing time extension* and *computer file extension*) are probable, so we need an additional information to apply our analysis to. However, if we provide the Q/A for a narrow domain, only one meaning is expected and the other is exceptional. That is why this additional information becomes redundant.

To comment on the rule δ we present the simplified default "without *extension*", which would mean that we can always eliminate *tax* from a query.

$$\frac{\perp:\ not\ @tax()}{not\ @tax}$$

However, this rule would misinterpret the queries *What is tax?, Can I deduct the tax I paid last year?,* and others.

Depending on the order in which rules of eliminating redundant parts and trivial rules are used, we can obtain several extensions containing more or less elements of the initial structure, because these two types of defaults make each other inapplicable. If we begin with trivial rules, the representation remains unmodified. Otherwise, some entities are eliminated. The order in which defaults are applied can be chosen by knowledge engineers depending on the task specifics.

Conclusions

In conclusion, we outline the semantic role of the suggested approach. Generally speaking, default rules are used to convert a formal representation of an NL question in the form which is adequate as a knowledge base query. Such the conversion is required even for a semantically correct representation of a given query. The set of default rules with consequents that correspond to a single knowledge base query covers semantically different, but pragmatically similar questions. Syntactically different questions are handled on the earlier stage (at a lower level with respect to a knowledge base); however, application of default rule is capable of correcting a wrong syntactic or semantic processing.

As to the quantitative measure of how the Q/A accuracy can be improved by applying default rules, we present the data for the domain on retirement saving, which is quite narrow. This domain contains 1200 generalized questions (semantic headers), 400 answers prepared by an expert, and is subject to testing by 5000 questions most of which were supplied by customers. In the Table 2 below we show the percentages of the above questions which were properly answered, caused situations where there was no appropriate knowledge available, led to a lack of understanding or a wrong answer. As we see, the overall accuracy was increased by 15 percent. It included 4.6 percent cases when the query modification lead to a successful run against a knowledge base versus a void run without such mofification, and 10.4 percent when a better answer was returned.

Domain develop-ment step	Correct answer	No knowledge	No understan-ding	Misunders-tanding
Before using default rules	65	7	9	19
After using default rules	75	7	6	12

Table 2: Percentages of correct/wrong answers before and after default rules were used to transform the formal representation of input query.

One of motivations of our study of default reasoning for Q/A is the lack of convincing, from our viewpoint, examples of practical default reasoning, and especially the operational semantics of default reasoning. The verbal description highlights “usual” (default) and “unusual” circumstances but, as only the data is formalized and there are conflicting rules while computing the extensions, intuitive perception of being usual and unusual dissipates. Only thorough consideration of natural language scenarios, presented within a vertical domain, may serve as a satisfactory illustration of practical nonmonotonic reasoning. Nontrivial extensions induced by a query appear when this query has multiple ambiguities. In particular, the situations where two words or multi-words have double possible meanings each such that the sentence has at least three overall meanings, are relatively rare (Krijevsky and Galitsky 2002).

NLP offers unique possibilities to apply nonmonotonic reasoning to a domain which is sufficiently complex in terms of knowledge structure on one hand and possess the intrinsic feature of having a typic and an atypical meanings on the other hand. Advanced theoretical investigations have been carried out in order to build the formal background for nonmonotonic intuition and a series of counter-examples (see e.g. Pascu and Carpentier 2002). However, the number of practical applications of nonmonotonic reasoning is far below the number of the theoretical results and their value. We believe that this study contributes to restoration of the balance between theoretical and applied issues of default reasoning.

References

Antoniou, G. 1997. Nonmonotonic reasoning. MIT Press Cambridge, MA London England.

Brewka, G., Dix, J., Konolige, K. 1995. Nonmonotonic reasoning: an overview. *CSLI Lecture Notes* **73**.

Bochman, A. 2001. A Logical Theory of Nonmonotonic Inference and Belief Change. Springer Verlag.

Ciravegna, F. and Lavelli, A. 1999. Full text parsing using cascades of rules: an information extraction perspective. In *Proc 9th European ACL Conf.,* Bergen Norway.

Creary, L.G. and Pollard, C.J. 1985. A computational semantics for natural language. *In Proc ACM-85* pp. 172-179.

Galitsky, B. 2003. *Natural Language Question Answering System: Technique of Semantic Headers*. Advanced Knowledge Intl., Adelaide Australia.

Hirst, G. 1988 . Semantic interpretation and ambiguity. *Artificial Intelligence* **34** 2, pp.131-177.

Krijevskiy, P., Galitsky, B. 2002. Building extensions of default logic to represent ambiguous sentences. DIALOGUE: International Workshop on Computational Linguistics, Russia.

Ourioupina O., Galitsky B. 2001. Application of default reasoning to semantic processing under question-answering *DIMACS Tech Report* 2001-16, May 2001.

Pascu,A. Carpentier, F.-G. 2002. Object Determination Logic - A Categorization System. FLAIRS Conference, 178-180.

Reiter R. 1980. A logic for default reasoning. Artificial Intelligence **13**, pp. 81-132.

Romacker, M., Markert, K., Hahn, U. 1999. Lean semantic interpretation, In Proc *International Joint Conference on Artificial Intelligence*-99, pp.868-875.

Sondheimer, N.K., Weischedel R.M., and Bobrow, R.J. 1984. Semantic interpretation using KL-One. In COLING-84, 101-107.