# Implementing Commonsense Reasoning via Semantic Skeletons for Answering Complex Questions

**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
http://www.dcs.bbk.ac.uk/~galitsky/

**Abstract**

We build the knowledge representation machinery for answering complex questions in poorly formalized and logically complex domains. Answers are annotated with deductively linked logical expressions (semantic skeletons), which are to be matched with formal representations for questions. Technique of semantic skeletons is a further development of our semantic header-based approach to question answering.

The question-answering technique has been implemented for the financial and legal domains, which are rather sophisticated on one hand and requires fairly precise answers on the other hand.

## Introduction

Question-answering (Q/A) systems have become important means of information access, especially in connection with popularity of a wide spectrum of customer response management (CRM) services. Personalized advisors capable of understanding customer circumstances formulated in natural language (NL) are becoming an essential CRM component.

While the keyword search and open domain question answering are oriented for a wide (horizontal) domain, handling relatively simple questions containing an entity and its attribute, this is not the case for a legal, financial or business advisor to be presented here. Representation of the above knowledge, oriented to the general audience, is much less structured and requires much richer set of entities than a natural language interface to SQL databases (Maybury 2000, Popescu et al 2003, Galitsky 2003). Furthermore, the structure of links between these entities is significantly more complex in such domains (Table 1).

Several current statistical and superficial semantic analysis – based technologies are able to provide a framework that approximates the complex problem of answering questions from large collections of texts (Gey et al 1998, Ng et al 2001, Pasca 2003). However, such domains as legal and financial require technique with rather deep semantic and pragmatic analyses to provide an accurate and personalized advice.

Our setting is different from open-domain Q/A targeting finding all or most relevant documents available. Advisor's task is to deliver a limited portion if information which:

1. is adjusted to the question;
2. is linked to additional resources if they are necessary;
3. indicates its position in the taxonomy of given Q/A domain;
4. is consistent with other answers and provides a uniform coverage of Q/A domain.

Earlier studies into design of natural language-based and expert systems showed that adequate commonsense reasoning is essential for answering complex questions (e.g. Winograd 1982). A number of recent studies have shown how application of advanced reasoning is helpful to compensate for a lack of linguistic or domain-dependent knowledge while answering complex questions (Moldovan et al 2002, Rus and Moldovan 2002, Baral et al 2004, Galitsky 2004)).

In this paper we continue development of the knowledge representation and reasoning technique for answering complex questions. The technique of semantic headers (SH, Galitsky 2003) is intended to represent and reason about poorly structured knowledge, manually extracted from text and match it with formalized questions. Having undergone the commercial evaluation, this technique demonstrated the superior performance in the market niche of expensive question answering systems, requiring a substantial domain-representation work of knowledge engineers. However, its accuracy and complexity of delivered advices is much higher than that of open-domain question answering with automatic annotation. SHs are special logical forms oriented to represent a partial (most important) information from a textual document.

Semantic skeletons (SSK) are intended to extend the functionality of question answering by means of commonsense reasoning machinery. Designed for the above market niche, a semantic skeleton – enabled

knowledge domain provides a better coverage of a totality of possible questions. This is due to the fact that an "emergent" question is expected to be deductively linked to one or more of the existing annotated answers by application of commonsense reasoning, inherent to SSK.

| |
|---|
| How much lower is an adjustable rate mortgage compared to a fixed rate loan? |
| Does the "start" rate quoted by lenders on a loan stay in effect for the term of the mortgage? |
| How can I avoid negative amortization on an adjustable rate mortgage? |
| How risky is a 125 percent loan to value second mortgage? |

Table 1: The samples question of *mortgage* domain. NLP system needs to handle up to four entities to perform Q/A in financial domains. Neither keyword search – based nor statistical nor syntactic match can provide satisfactory advising in vertical domains.

To conclude the introduction, we outline the desired suite of features we are attempting to achieve by SSK – based knowledge representation machinery:

1) simplicity and expressive power;
2) capability to reason with incomplete information;
3) existence of a well developed programming methodology;
4) availability of rather efficient reasoning features;
5) encoding defeasible relations, defaults, causal relations, argumentations, and inheritance hierarchies;
6) being elaboration-tolerant knowledge base, i.e., be able to accommodate new knowledge without doing large-scale modification.

## Semantic headers of answers

The problem of question answering in a vertical domain is posed as building a mapping between formal representations of the fixed set of answers and formal representations of possible questions. The technique of semantic headers is intended to be the means of conversion of an abstract textual document into a form, appropriate to be associated to a question and to generate an advice (Galitsky 2003). There are two opposite common approaches to this problem. The first one assumes that complete formal representation of any textual document is possible, and the second one assumes that the textual information is too tightly linked to the NL and it cannot be satisfactorily represented without it. The former approach relies on the match of formalized query with the full knowledge representation for answers, and the latter is based on syntactic match between the question and the sentences from answers.

The semantic header (SH) technique is an intermediate one in respect to the degree of knowledge formalization. It is intended to be the means by which an abstract textual document is converted into a form appropriate to be associated to a question and to generate an answer (Galitsky 2003). Only the data, which can be explicitly mentioned in a potential query, occurs in semantic headers. The rest of the information, which would be unlikely to occur in a question, but can potentially form the relevant answer, does not have to be formalized.

SH technique is based on logic programming, taking advantage of its convenient handling of semantic rules on the one hand, and its explicit implementation of a domain's common-sense reasoning on the other hand. The declarative nature of coding semantic rules, domain knowledge and generalized potential queries introduces suggests logic programming as a reasonable tool (Baral et.al. 2004).

Let us consider the *Internet Auction* domain, which includes the description of bidding rules and various types of auctions.

*"Restricted-Access Auctions. This separate category makes it easy for you to find or avoid adult-only merchandise. To view and bid on adult-only items, buyers need to have a credit card on file with eBay. Sellers must also have credit card verification. Items listed in the Adult-Only category are not included in the New Items "*

What is this paragraph about? It introduces the "Restricted-Access" auction as a specific class of auctions, explains how to search for or avoid selected category of products, presents the credit card rules and describes the relations between this class of auctions and the highlighted sections of the Internet auction site. We do not change the paragraph in order to adjust it to the potential questions answered within it; instead, we consider all the possible questions this paragraph can serve as an answer to:

*What is the restricted-access auction?* This question is raised when a customer knows the name of the specific class of auction and wants to get more details about it.

*What kind of auctions sells adult-only items? How to avoid adult-rated products for my son? Do you sell adult items?* These are the similar questions, but the class of auctions is specified implicitly, via the key attribute *adult-only.*

*When does a buyer need a credit card on file? Why does a seller need credit card verification?* These are more specific questions about what kind of auctions requires having credit cards on file, and what is the difference in credit card processing for the auction seller/buyer. The paragraph above serves as answer to these questions as well, and since we are not dividing the paragraph into smaller fragments, the question addressee will get more information than he/she has directly requested; however this additional information is relevant to that request.

Below is the list of semantic headers for the answer above.

| |
|---|
| *auction(restricted_access):-restrictedAuction.* |
| *product(adult):-restrictedAuction.* |
| *seller(credit_card(verification,_),_):-restrictedAuction.* |
| *credit_card(verification,_)):-restrictedAuction.* |

*sell(credit_card(reject(_,_),_),_):-restrictedAuction.*
*bidder(credit_card(_,_),_):-restrictedAuction.*
*seller(credit_card(_,_),_):-restrictedAuction.*
*what_is(auction(restricted_access,_),_):-restrictedAuction.*

Then the call to *restrictedAuction* will add the paragraph above to the current answer, which may consists of multiple pre-prepared ones.

Now we will briefly introduce a generic set of semantic headers, taking advantage of the technique of non-conventional logic programming. For an entity *e*, its attributes $c_1$, $c_2$, …, variables over these attributes *C*, $C_1$, as well as other involved entities $e_1$,…, and the ID of resultant answer, SHs look like the following:

*e(A):-var(A), answer(id).* This is a very general answer, introducing (defining) the entity *e*. It is not always appropriate to provide a general answer (e.g. to answer *What is tax?*>), so the system may ask a user to be more specific:

*e(A):-var(A), clarify([$c_1$, $c_2$, …]).* If the attribute of *e* is unknown, a clarification procedure is initiated, suggesting the choice of an attribute from the list $c_1$, $c_2$,… to have a specific answer about $e(c_i)$ instead of just for *e(_).*

*e(A):-nonvar(A),* $A = c_1$ *, answer(id).* The attribute is determined and the system outputs the answer associated with the entity and its attribute.

$e(e_1(A))$*:-nonvar(A),* $A = c_1$ *,* $e_1(A)$*.*

$e(e_1(A),e_2)$*:-nonvar(A),* $A \neq c_1$ *,* $e_2(\_)$*.* Depending on the existence and values of attributes, an embedded expression is reduced to its innermost entity that calls another SH.

*e(A,id).* This (dead-end) semantic header serves as a constraint for the representation of a complex query, $e_1(A,id)$, $e_2(B,id)$, to deliver just an *answer(id)* instead of all pairs for $e_1$ and $e_2$ . It works in the situation where neither $e_1$ nor $e_2$ can be substituted (into each other).

Note that *var/1* and *nonvar/1* are the built-in PROLOG metapredicates that obtain the respective status of variables.

From the perspective of logic, the choice of semantic header to be matched against a formal representation of a query corresponds to the search for a proof of this representation, considering SHs as axioms.

We conclude this Section by the following definition. *Semantic header*s of an answer are the formal generalized representations of potential questions. Semantic headers contain the essential information of answers and serve to separate them, being matched with formal representations of questions. Semantic headers are built taking into account:

- The set of other semantically close answers.
- The totality of relevant questions, semantically similar to generic questions above.

## Semantic skeletons

Evidently, a set of SH represents the associated answer with the loss of information. What kind of information can be saved given the formal language that supports semantic headers?

When we extract the answer identifying information and construct the semantic headers we intentionally lose the commonsense links between the entities and objects used. This happens for the sole purpose of building the most robust and compact expressions for matching with the query representations. Nevertheless, it seems reasonable to conserve the answer information which is not directly connected with potential questions, but useful for completeness of knowledge being queried. A semantic skeleton (SSK) can be considered as a combination of semantic headers with *mutual explanations* of how they are related to each other from the answers perspective. SSKs are domain-specific and coded manually by knowledge engineers

SSKs serve the purpose of handling the queries not directly related to the informational content of the answers, represented by semantic headers. For an answer and a set of semantic headers, an SSK derives an additional set of *virtual* headers to cover those questions which require a deductive step to be linked with this answer. In other words, a semantic skeleton extends a set of questions which is covered by existing semantic headers towards the superset of questions, deductively connected with the former ones. It happens during a question answering session, unlike the creation of regular SHs which are built in the course of domain construction.

Yielding virtual SHs in a domain can be written as $\forall a$ *SSK :* $\{SH(a)\} \rightarrow \{vSH(a)\}$, where $\{SH(a)\}$ is the set of original semantic headers for answer *a*, and $\{vSH(a)\}$ is the set of virtual semantic headers derived from SSK for an answer *a*. A virtual semantic header (vSH) can be yielded by multiple answers (Galitsky 2003). However, a vSH cannot be a regular header for another answer (note that two semantic headers for different answers are allowed to be deductively linked): $\forall a,a'$ $vSH(a) \cap SH(a')=\varnothing$. Hence, a vSH for a query is an expression that enters a clause of the semantic skeleton and can be matched with the translation formula of a query or its conjunctive component. In the latter case, the terms of mentioned clauses must not match with the negations of the (conjunctive) components of that translation formula.

The idea of SSK is depicted in Figure 2. The input query is matched against the vSHs if there is no appropriate regular semantic header to match with. Virtual semantic headers are obtained given the terms of SSK clauses. The SHs are assigned to answers directly. However, vSHs are assigned to answers via clauses. Both *Answer1* and *Answer2* may have other assigned regular and virtual SHs.

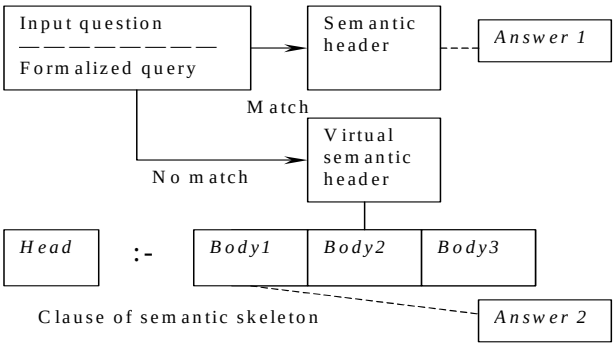


Fig. 1: Illustration for the idea of semantic skeletons

For example, imagine a semantic header *tax(income)* that is intended to handle questions about *tax brackets* in the *Tax* domain: how the *tax* amount depends on *income*. Evidently, this answer would be a relevant one to the question *What would my tax be if I lost my job last year?* Since *losing a job* is not directly related to *tax* (the former is deductively linked to the latter via *income*, *job(lost)→ not income(_)*), it would be unreasonable to have a special semantic header to link *tax* and *job-lost*. Therefore, the expression *job(lost)* serves as a virtual semantic header in the TAX domain, being generated dynamically from the clause *job(lost)→ not income(_)*, instead of being a regular one. If we do not use the regular semantic header instead of the virtual one for the entities which are neither deductively nor syntactically linked in a query, it would damage the domain structure and lead to an excess number of semantic headers. Indeed, this used to happen before the concept of the SSK was introduced.

At the same time, in the IRA domain the *loosing job* scenario is under special consideration, and expressions *ira(tax(income))* and *ira(job(lost))* are expected to be the semantic headers for different answers; one for calculating *tax* on *IRA distribution amount* that depends on *income*, and the other for the special case of *tax* on *IRA distribution* under *employment termination.* Thus a pair (triple, etc.) of entities may form a vSH (that requires a SSK-clause that would yield, generally speaking, multiple links between entities) or, form a regular header, depending on whether these entities are directly semantically or syntactically linked in a query. The clauses of the semantic skeleton are not *directly* used to separate answers, so they can be built as complete as possible irrespectively on the knowledge correlation with other answers. Furthermore, semantic skeletons for a pair of answers may overlap, having the common clauses.

As well as approximation of meaning by SHs, SSKs are capable of involving approximate semantic links. For example, various *forms of payment* questions are addressed to the *Internet retailer* domain which usually has an answer about *credit card payment*. How should we handle the questions mentioning *payment by check, money order, wiring* etc? "Pure" SH technique requires enumeration of SHs:

```
payment(check):-credit_card_payment_answer.
payment(money_order) :-credit_card_payment_answer.
payment(wiring) :-credit_card_payment_answer.
payment(credit_card) :-credit_card_payment_answer.
```

However, using a SSK clause: *payment(X):-member(X, [ check, money_order, wiring, credit_card ]),* one can use the fourth SH above as a regular SH and the first three ones as virtual SHs, involving the clause about forms of payment. The advantages of using a SSK are the lower number of SHs to code, clearer semantic structure of a domain and reusability of encoded commonsense knowledge for the similar domains.

We proceed to another example of a semantic skeleton. Again, the semantic headers need to be deductively linked via the clauses, involving the entities from these semantic headers and other ones. The clauses below present the explanation of how a term *divorce* is linked to the terms *marriage, tax, file, separate,* and *joint*. The first clause, completing the set of semantic headers above, introduces the commonsense fact that being *divorced* is an opposite entity to being *married*. The second clause is saying that if a couple was filing a *joint tax return* before the *divorce*, they are *filing separate tax returns* afterwards. Enumeration of terms within a clause may be used to express the temporal relationship of a sequence in time (read *file(joint) then divorce(_)*).

```
divorce(_):- not marriage(_).
tax(file(separate)):- file(joint), divorce(_).
single(_):-divorce(_);not marriage(_).
```

Using just the semantic headers we can answer the divorce questions without knowing that *divorce* ends *marriage*! Surprisingly, one does not need to know that fact to separate answers about divorce. Intuitively, a speaker would definitely need some basic commonsense facts to *talk* about a topic, but an agent can answer questions about a topic, including rather specific and deep questions, without these facts. SSKs come into play, in particular, to semantically link the basic domain entities. At the same time, a semantic skeleton is the *least* knowledge required to have all the entities linked.

Note that the predicates we use to describe the tax issues of marriage to occur in an SH for Q/A are different from ones we would use to better formalize the domain itself. The SH-oriented predicate *divorce* ranges over its attributes, which are reduced to a single argument, and an *extended* predicate *divorce* in a logic program would have two arguments ranging over the divorce parties and other arguments for the various circumstances of the *divorce* action. Evidently extended predicates better fit the traditions of linguistic semantics, specifying the *roles* of their arguments. Frequently, extended predicates are required to form the semantic skeleton; analogous semantic header predicates should

then be mutually expressed via the extended ones. These definitions do not usually bring in constraints for the attributes of semantic header predicates. Below is the semantic skeleton (the set of clauses of the form *extended_predicate* ↔ *SH_predicate*) for the sample answer above. The first argument of the extended predicate *tax* (that is indeed a metapredicate) ranges over the formulas for *taxpayer's states*; therefore, *tax* is a metapredicate. Note that these clauses need to work both ways (right to left and left to right) to deploy the capability of the vSHs, yielded by the SSK.

A particular case of what we call a non-direct link between entities is the temporal one. If a pair of answers describe two consecutive states or actions, and a given query addresses a state before, in between, or after these states (actions), the SSK clauses are expected to link the latter states with the former ones and to provide a relevant (although possibly indirect) answer.

*divorce(Husband, Wife, Date, Place,…) ↔ divorce(_).*
*marriage(Husband, Wife, Date, Place,…) ↔ marriage(_).*
*tax( (pay(Husband)&pay(Wife)), file (separate),_) ↔ tax(file(separate))).*
*tax( (pay(Husband)&pay(Wife)), file (joint),_) ↔ tax(file(joint)).*
*divorce(Husband, Wife, Date, Place) ↔ not marriage(Husband, Wife, Date1, Place1)*
*tax( (pay(Husband)&pay(Wife)), file (separate),_) ↔*
*tax( (pay(Husband)&pay(Wife)), file (joint),_), divorce(Husband, Wife, Date, Place).*

For a scenario, a SSK may include alternating sequences of states, interchanging with actions intermediate states (we assume no branching plans for simplicity). Not unlike the situation calculus considerations, states and actions can be merged into the same sequence from the perspective of being explicitly assigned with an answer. The set of these states and actions falls into subsets corresponding to the answers (based on the expressions for these states and actions which are vSHs as well). It can naturally happen that answers are not ordered by this sequence; they may be assigned by the SHs for alternating states and actions. Then, if a question addresses some *unassigned states* or *actions*, those answers should be chosen which are assigned to the *previous* and *following* elements of the sequence. We present the state-action sequence for the *tax return preparation* scenario:

*file(tax(return)):- tax(minimize(speculate), _),* % state
*collect(receipts), calculate(deduction),* % action
*collect(form(_)), consult(accountant),* % action
*fill(form(_)),* % action
*calculate(tax(property )),* % action
*calculate(tax(income)),* % action
*estimate(tax(value)),* % state
*send(form),* % action
*tax(return(_), expect(_)).* % state

Semantic skeletons are helpful for formalizing queries which are conjunctions of multiple terms (This happens for complex queries consisting of two or more components, for example *Can I qualify for a 15-year loan if I filed bankruptcy two years ago with my partner? → loan(qualify)& bankruptcy(file, 2years)*). If a term is either matched against none of the semantic headers or delivers too many of them, then this term can serve as a virtual one. In Table 2 below we analyze various cases of the satisfaction (matching) of a translation formula with two terms against regular and virtual SHs.

## Evaluation of question answering

A series of tax return assisting, investment, mortgage (see Fig.1) and financial companies has employed Knowledge-Trail's advisors. It can replace human agents, automatically answering tax questions in up to 90% of all cases. Human agents were ready to intervene Q/A process in case of a failure of the automatic system.

In particular, the suite of legal (family law) domain has been created, which covers sufficient information for the general audience of using about 1000 answers in the principle and accompanying domains. The domain includes more than 240 entities and more than 2000 of their parameters in these sub-domains. More than 3000 semantic headers and semantic skeletons were designed to provide an access to these answers.

During the beta testing, the Family Law Adviser was subject to evaluation by a few hundred users starting from the summer of 2000. Customers had the options to provide the feedback to the system concerning a particular answer if they were dissatisfied or not fully satisfied with it (too long, non-relevant, partially relevant, etc.). With the answer size not to exceed 6 paragraphs, the system correctly answers more than 70% of all queries, in accordance to the analysis of the Q/A log by the experts. Even with 82% resultant accuracy (Table 2), which is relatively low for traditional pattern recognition systems, over 95% of customers and quality assurance personnel agreed that the legal advisor is the preferable way of accessing information for non-professional users. The reader may see a demo at
hades.dcs.bbk.ac.uk/users/galitsky/www/ira.pl.

Usually, customers tried to rephrase questions in case of the system's misunderstanding or failure to provide a response. Reiteration (rephrasing the question) was almost always sufficient to obtain the required information. At the beginning of the evaluation period, the number of misunderstood question was significantly exceeded by the number of answers not known by the system. This situation was dramatically reversed later, however the number of misunderstood questions was monotonically decreasing in spite of an increase in overall represented knowledge.

| Development step | Sourc of questions | Correct answer | No knowledge | No understanding | Misunderstanding |
|---|---|---|---|---|---|
| Initial coding | Initially designed (expert) questions for SH | 47 | - | 37 | 18 |
| Testing & reviewing of initial coding | Initially designed & accompanying questions | 52 | 18 | 25 | 10 |
| Adjustment to testers' questions | Additional and reformulated and rephrased testers' questions | 60 | 15 | 10 | 15 |
| **Adding SSKs** | **Commonsense domain knowledge** | **67** | **17** | **4** | **12** |
| Adjustment to content providers' questions | More questions, reflecting a different viewpoint of the subject | 74 | 8 | 4 | 14 |
| Adjustment to users' questions | No additional questions | 82 | 4 | 4 | 10 |

Table 2: The progress of question answering enhancement at consecutive steps of domain development (%). SSK step is shown in bold. *Commonsense domain knowledge* helps to yields questions which were not encoded during initial phase of domain development, but are nevertheless relevant.

Use of SSK allowed increasing the percentage of correctly answered questions from 60 to 67: about 7 % of questions are indirect and require to apply a commonsense reasoning to link these questions to formalized answer components. In 2% of cases vSHs were built but they derived multiple inconsistent SHs because of a lack of a specific knowledge (which has been added later). As one would expect applying SSK technique, the decrease of cases with a lack of understanding (6%) was higher then (twice as much as) the decrease of cases with misunderstanding 3%.

## Conclusions

Application of the SSK technique to NL Q/A showed the following. There is a superior performance over the knowledge systems based on the syntactic matching of NL queries with the previously prepared NL representation of canonical queries, and the knowledge systems based on fully formalized knowledge. Moreover, the domain coverage of SSK is better than that of SH because a new question can be reduced to existing pre-coded ones by means of commonsense reasoning.

The SSK approach gives a higher precision in answers than the SH and syntactic –matching based ones because it involves the semantic information in higher degree. The SSK technique gives more complete answers, possesses higher consistency to context deviation and is more efficient than the latter approach because the full knowledge formalization is still not required.

The achieved accuracy of providing an advice in response to a NL question is much higher than an alternative approach to advising in a vertical domain would provide, including open-domain question answering, an expert system on its own, a keyword search, statistical or syntactic pattern matcher. Indeed, SSK technique approaches the accuracy of a Q/A in a fully-formalized domain, assuming the knowledge representation machinery obeys the features outlined in the Introduction.

Using semantic resources like WordNet, automated statistic-based annotation systems and commonsense ontology resources would decrease the cost of domain development at the expense of lower accuracy. In our future studies we plan to perform the above integration for less narrow domains and less complex questions.

## References


Gey, F.C., Chen, A. 1998. Phrase Discovery for English and Cross-language Retrieval at TREC-6, *Text Retrieval Conf*, NIST Special Publication.

Galitsky, B. 2003. Natural Language Question Answering System: Technique of Semantic Headers. Advance Knowledge International, Australia.

Galitsky, B. 2004. Use of Default Reasoning for Disambiguation Under Question Answering *FLAIRS – 04*, May 16-18, Miami FL.

Pasca, M. 2003. Open-Domain Question Answering from Large Text Collections *CSLI Publication series.*

Baral, C., Gelfond, M. and Scherl, R. 2004. Using answer set programming to answer complex queries. In *Workshop on Pragmatics of Question Answering at HLT-NAAC2004*

Moldovan, D., Pasca, M., Harabagiu, S. and Surdeanu, M. 2002. Performance issues and error analysis in an open-domain question answering system. In ACL-2002.

Rus, V. and Moldovan, D. 2002. High Precision Logic Form Transformation, International Journal on Artificial Intelligence Tools, vol. 11 no. 3.

Maybury, M.T. 2000. Adaptive Multimedia Information Access - Ask Questions, Get Answers. *First International Conference on Adaptive Hypertext AH 00.* Trento, Italy.

Ng, H.T., Lai Pheng Kwan, J. and Xia, Y. 2001. Question Answering Using a Large Text Database: A Machine Learning Approach. *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing EMNLP 2001.* Pittsburgh, PA

Popescu, A.-M., Etzioni, O. and Kautz, H. 2003. Towards a Theory of Natural Language Interfaces to Databases. *Intelligent User Interface.*

Winograd, T. 1972. Understanding natural language. NewYork: Academic Press.