DIMACS Technical Report 2000-29
October 2000

Technique of semantic headers: a manual for knowledge engineers

by

Boris Galitsky[1]

[1] iAskWeb, Inc., 681 Main Str. # 1b #B Waltham MA 02451.
http://dimacs.rutgers.edu/~galitsky
(galitsky@dimacs.rutgers.edu)

The author gratefully acknowledges the grant NSF STC 91-19999 for the support while visiting DIMACS.

ABSTRACT

This manual addresses the issues of knowledge representation in the form of *semantic headers* (SH) for conversion of a textual document into a form, appropriate for question answering. A set of semantic headers is intended to formally represent the essential idea of a document with respect to possible questions, such that this document would serve as an answer. The knowledge base then includes the textual answers, structured by means of assigned semantic headers. The formal representation of the natural language (NL) query is then matched against the knowledge base in the form of a set of these headers. The logical mechanisms of matching query translations against semantic headers and derivation of semantic headers themselves is presented, as well as applications of meta-reasoning, default reasoning, reasoning about action and time, and graph representation of answer classification.

This technique shows superior performance over the knowledge systems based on syntactic matching of NL queries with the prior prepared NL representation of canonical queries, and the knowledge systems based on the fully formalized knowledge. Our approach gives the higher precision of answers than the former one because it involves the semantic information in higher degree. At the same time, in the logically complex and poorly structured domains SH 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 not required.

The manual of semantic header technique is intended to assist the linguists and knowledge engineers in the creation of question answering systems for the sophisticated vertical domains. The manifold of coding samples and extended discussions help to deeper understand the peculiarities of using logic programming in the knowledge representation. The tutorial part is followed by the qualification test using the multiple-choice questions.

1 Introduction

The recent explosion of information available on the World Wide Web determines question answering (Q/A) to be an appealing framework for finding relevant information, particularly in logically and structurally complex commercial domains. The success of Q/A services in horizontal domains like AskJeeves serves as a proof of the popularity of this technique. Usually, in horizontal domains customers have to evaluate multiple answers themselves to find the relevant ones (multi-step Q/A). Due to the fact that both questions and answers are expressed in natural language (NL), Q/A methodologies deal with language ambiguities and it seems more promising to incorporate both syntactic and semantic NLP techniques rather than statistical machinery. Several current semantic NLP-based technologies are able to provide the framework that approximates the complex problem of answering questions from large collections of texts (Katz 1990, Tarau et al 1999, Ciravegna et al 1999, Harman 1996).

In this paper, we develop the technique of semantic headers (SH) for the weakly formalized vertical domains. To satisfy the asking customer in a vertical domain, the textual answer needs to be direct (single-step Q/A), accurate, and contain a significant portion of relevant data (usually, above one-third). SH technique must possess the full flexibility of answer generation depending on the question generality and involvement of multiple domains. We illustrate the technique of semantic headers based on the series of domains of various nature, where the customers get advises on the variety of issues from the *personal finance* to *literature search*. The majority of examples we present are for the *tax* engine, which has been developed as a pioneer one in the suite of financial domains, including *loans, mortgages, insurance, real estate, banking, investment, legal,* etc. The essential features of Tax Q/A domain are large-scale vertical domain capabilities, logical complexity and high accuracy requirements. In addition, reasoning about knowledge, time and action is required to properly analyze the questions.

In our earlier papers (Galitsky, 1998, Galitsky, 1999, Galitsky 2000) we have described the political domain, tax engine and querying works of literature based on mental states of heroes. Representation of Tax knowledge, necessary for the general audience, is much less structured than that of political domain and requires much richer set of entities than the mental states domain. Furthermore, the structure of links between these entities is rather complex in the Tax domain; it serves as a critical milestone of both the system scalability and language expressiveness/knowledge representation capabilities.

Ideal Q/A systems should have a good dialog understanding, rich knowledge bases and quality text mining methods. For the tax domain, it needs to incorporate common sense reasoning methods and use good approximations of the general knowledge about the business domain. Our system, where both semantic analysis and knowledge representation is implemented using the logical (meta-) programming technique, is well suited for these features. Conversely, the majority of approaches approximate Q/A with the combinations of information retrieval (IR) and information extraction (IE) techniques. However, current IE systems make these combinations impractical solutions for delivering answers for open-domain questions, due to the dependency of IE systems on domain knowledge.

Recently, new systems for open-domain text Q/A have been developed, providing new insights into techniques useful for Q/A. Given a query, these IR systems allow us to locate documents that might contain the pertinent information, but most of them leave it to the user to extract the useful information from a list of documents. The extraction of the information of interest is the object of IE systems, provided that the information is to be mapped into a predefined, target representation, known as template, often imposed by interfaces to databases. In the tax domain, information is unstructured by its nature and database querying approach is hardly appropriate. Semantic headers play the role of these templates, encoded in highly expressive predicate language.

This work follows the strong raise of interest to NLP tools of automation of customer support systems. Increase of the system performance, answering customer's questions is one of the essential tasks of the system of automatic customer support. Joint efforts of the automatic question-answering agents, Q/A personnel of *Tax Assistance* companies and knowledge engineers to develop and employ the domains will become more efficient if the customers can extend the domain independently with their specific knowledge. After a user got accustomed with the NL customer support system and learned the proper style of forming the questions, he/she can teach the system with the own experience. SH technique is intended to allow the automatic addition of new knowledge to Q/A domain, convenient to the customers to employ.

Developing SH technique requires application of such modern approaches of logical AI as reasoning about agent knowledge, situation calculus, nonmonotonic reasoning, reasoning in the conditions of high uncertainty and inconsistency, etc., which does not frequently find software implementations. At the same time, SH is based on the traditional pattern-recognition technique such as classification graph, its visualization and modification, accompanied by the domain-dependent code generation.

It is well known that in AI systems relatively modest raise of accuracy in respect to a basic approach can be only reached by the significant time-intensive efforts. This is the case for Q/A; our experience with more than 15 domains with contribution of more than 30 knowledge engineers showed that each answer needs a thorough analysis to be included and assigned with headers. On the stage of initial design, semantic header approach was intended to be much more time-consuming under manual annotation than an automatic IR system. At the same time, SH approach is supposed to be less effort consuming than a full-range NLP system, including detailed morphological and syntactic analysis, when it does not affect the Q/A accuracy.

1.1 What Semantic Headers are?

The technique of semantic headers (SH) is intended to resolve a problem of conversion an abstract textual document into Q/A form. There are two opposite 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 natural language and it cannot be satisfactorily represented without it.

The first approach brings the advantage of the understanding, based on the full-range reasoning capabilities, but sacrifices the necessity of NL generation, which introduces additional deviation of the answer semantic. Attempts to provide an exact answer, inherent to the fully formalization approach, narrow down the overlap between a query and existing answers. (Under SH, query understanding is posed as a recognition problem: how to find the most relevant answer for a given query, even if there is no direct answer).

The SH technique is an intermediate one in respect to the degree of knowledge formalization. SH approach is intended for the NLP system, which is the combination of information extraction and text understanding systems. In information extraction, only a fraction of text is relevant; information is represented by a predefined, fully formalized, relatively simple (in respect to the original text) structure (e.g. logical expressions or a database). Only the data, which can be explicitly mentioned in a potential query, occurs in semantic headers. The rest of information, which would unlikely occur in a question but can potentially form the relevant answer, does not have to be formalized.

Pattern recognition methodology has the significant influence on how we treat the answers. When we say that a document (or its fragment) serves as an answer to a question we mean that for this question (related to a domains) either (or) holds:

The fragments contains the information that is the subject of this question;

The fragment is marginally related to the question, but the other fragments have even lower correlation with the question (assuming the question indeed related to the given domain).

1.2 Introductory example

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. Your card will not be charged. Sellers must also have credit card verification. Items listed in the Adult-*

Only category are not included in the New Items page or the Hot Items section, and currently, are not available by any title search. "

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 to adjust it to the potential questions; instead, we consider all the possible questions this paragraph can serve as an answer to by enumerating the semantic. Building the semantic headers of a textual document is based on the posing of query understanding problem as the recognition of the best pattern (document, answer). For example, if there is a question such that the paragraph above is a more appropriate answer than any other paragraph from the whole domain, then the paragraph above should serve as an (part of the) answer or at least a part of it.

There are two ways to achieve the delivering of the proper answer: either to encode the appropriate SH explicitly or to expect the system to find the closest SH with the relevant answer by its own machinery. Evidently, the former one is preferable, but the latter one is employed in the situations of real-world uncertainty and incompleteness.

Evidently, knowledge of the other paragraphs and the semantic model of the whole domain are required to build the set of semantic headers for the given paragraph. What kind of questions can this paragraph serve as an answer to?

1. *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.
2. *What kind of auctions sells adult-only items? How to avoid adult-rated products for my son? How do you sell adult items?* These are the similar questions, but the class of auctions is specified implicitly, via the key attribute *adult-only*. Having this question, the customer expects to receive something similar to the previous question.
3. *When does a buyer need a credit card on file? Who needs to have a credit card on file? Why does a seller need credit card verification?* These are more specific questions about what kind of auction 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 an 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 related 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 answer.

1.3 SH technique in the perspective of information extraction and retrieval systems

The SH technique is intended for the NLP system, which is combination of information extraction and text understanding systems. In information extraction, only a fraction of the text is relevant; information is represented by a predefined, fully formalized, relatively simple (in respect to the original text) structure (e.g. logical expressions or a database). Besides, the writer's intentions and meaning peculiarities do not play the key role under information extraction.

In contrast to the information extraction systems, the goal is to interpret the whole sense of the query; the target query representation is design to circumscribe the full range of complexity of the language. In addition, it is essential to recognize the specific meanings and writer goals. In our case, information extraction happens for the sole purpose of building the structure to match against the query understanding expression. SH technique combines the information extraction and query understanding components such that the results of the former are the basis for the question answering, implemented by the latter component.

SH can be referred as a pragmatic technique of NLP with an **accent to semantics**. In terms of domain coding efficiency, it occupies an intermediate position between the extended keyword search (automated, low time-consuming) and full-range NLP (manual introduction of domain-specific morphological, syntactic, semantic and domain structure rules). In terms of automation of the domain creation, SH is appropriate for both automatic and manual annotation. However, commercially valuable are Q/A domains, manually created by knowledge engineers or automatically derived from these domains, having the language and semantic structure fixed.

The SH technique can be compared with the simpler and more traditional approach when the textual data is transferred into the **answer database** (fully structured data representation). SH is a generalization of Q/A system with knowledge representation in the form of a relational database of answers. In terms of expressiveness of the knowledge representation language, we take advantage of first-order predicate one, extended with the meta-predicates (the meta-variables range over well-formed formulas). Besides, SHs themselves are represented in terms of extended first-order language. Note that a fixed SH semi-structured domain can be rewritten via (very sparse) relational database, but even insignificant addition of data will cause the major modification of the database relations. In addition, vertical Q/A domains frequently require combination of splitting an answer into many more specific ones or merging them while domain design with dynamic sequencing of the stored answers to prepare the total one, delivered to a user. This feature seems to be hard to implement using the database approach. Besides, the conceptual relations between the parts of answers and between the whole answers, reflecting the domain specificity, are inexpressible in a database way of representation.

The nature of the **misunderstanding** and not knowing the proper answer under full-structured (database representation) and semi-structured (SH) Q/A is different. Besides, the role of information extraction for keyword search and for SH technique is distinguishing: the latter required semantic processing even at the step of annotation.

When knowledge is represented in the form of a database, meaning approximation technique is not applicable. Slight deviation in the formal representation of the query may lead to the dramatic change in the set of records, selected as an answer. If the query is misunderstood, response will include extra objects and/or wrong objects. Receiving a wrong set of records, a customer could accept them as a reasonable answer.

A typical example of such domain would be the search of a product via its parameters like price, weight, color, etc. There is a series of commercial products distinguishing in the SQL representation complexity and quality of semantic/pragmatic handling of logical connectives (see, for example, Microsoft English Query, ELFsoftware, EasyAsk, etc.).

Absence of answers for the database and SH domains may happen for both wrong understanding and lack of knowledge, but in SH domain the system rather provides a partially relevant answer than confesses in not knowing at al (at least, in the single-agent settings). In the database domain, lack of understanding and lack of knowledge is practically indistinguishable for a customer. Taking into account that the language of SH domain is richer than that of the database one, derived from the names of columns and their values, we conclude that SH Q/A system is potentially much more robust relatively to domain (language) complexity.

SH technology obey the **monotonicity property**: if there is a substantial deviation of understanding, the proper answer will still be provided if there is no other answer, semantically closer to the deviated meaning. In SH domains, wrong answers are actually unrelated ones, easily detectible. In the other words, SH system cannot provide a wrong information, in the worst case an answer can be non-relevant, but it happens not as frequently and not in as high degree of wrong meaning than under the keyword search, without employing semantic machinery.

When the **keyword** approach applied to a fully structured domain, it is a trivial field search. In the semi-structured Q/A domains, statistical machinery of information extraction is not capable of capturing the semantic degree of similarity of the set of keywords with the answer. Therefore, it is easy and frequent to get completely unrelated answers.

There are the following components of **meaning deviation** under SH technique:

- Deviation while annotation. The nature and value of this deviation is different while textual and formal language representation of the essential idea of the original text answer. The total annotation deviation is additive for these sub-components, taking into account whether these operations are performed manually or automatically.
- Deviation of meaning while building translation formula (representation of a question)
- Choice of the answer, which is not the closest one, while matching translation formula(TF) with SH.

Each of these components has the specific approximation strategy to decrease the deviation. See (Galitsky 1999) for the second one and section 3 below for the first and third respectively.

There is a tight connection between the problem of building hyperlinks between the documents with the classification of documents for Q/A in respect to IR. It is known, that the most difficult part of the automatic construction of a hypertext is building up the links that connect semantically related documents (Agosti et al 1996, Harman 1996 , Salton et al, 1996). It has been demonstrated that IR technology can be applied to the construction of content link between the parts of documents.

1.4 Approximation via formal headers: semantic versus syntactic ones

We pose the NL understanding problem as building a map between the set of all possible questions and the document set (compare with). This map is constructed, based on the semantic representation of a query, which follows the syntactic analysis.

Generally speaking, there are two ways to **match** the processed **query** with the **document**:

1. Matching the syntactic representation of NL query with the syntactic representation of the template questions (syntactic header). The score is estimated based on the number of common keywords in the query and in the template, taking into account the synonyms and semantic difference between the keyword (local semantic information).
2. Matching the semantic representation of a query with semantic representation of the document content (semantic header of a document). Each document has to be manually or automatically processed to create a list of predicates, representing the basic entities, presented in this document. SH of a document will contain this list of predicates with arguments, instantiated in as high degree as possible to reflect the content of this document. On the other hand, the semantic header should be general enough (arguments uninstantiated to the degree), such that the lower the specificity of a question, the larger the set of corresponding answers. In addition, the closer the meaning of two documents are, the formally closer the corresponding SHs should be.

We intend to show that the second approach has to be implemented instead of the first one to achieve the accurate NL understanding (reducible to classification task in case of SH).

What is the difference between matching the list of keywords of a question and an answer and matching the respective first-order expressions? This issue is addressed when the logical language is introduced as a one to represent formal and informal worlds. The origin of a first-order language is division of description terms into entities and their attributes.

1.5 Entity and its attribute (object)

It is important for semantic model to distinguish the roles of an entity and an object. Though the distinction has the fuzzy boundary and varies from one to another approach to NLP, we present the most essential differences.

1. Typically, the number of entities is much lower than the number of objects in a vertical domain.
2. To relate a query to a subdomain, it is sufficient to analyze the query words, which refer to entities than to objects.
3. In the first order logic, the simplest representation of the NL semantics, entities are represented by predicates and objects are represented by their arguments.
4. The arguments of a predicate have semantic types (e.g. person, country, city, product). The structure of all semantic types is a tree with directed nodes, which denote the relations of semantic types inclusion (e.g. person → leader of … → president).
5. To express the semantic link between predicates, we unify their arguments, which have compatible semantic type.
6. Typically, there are reasonable sub-domains of vertical domains where the following approximation holds: predicates and metapredicates correspond to verbs and objects/attributes correspond to nouns.

There is certain background that the first order language and its extension with metapredicates corresponds to both syntax and semantics of NL; in particular, the grammatical principle of a single subject-predicate relation (Redey 1999). Any syntactical relation such as complement forming, other than copulative conjunction, coordination, is repeatedly derivable from the subject-predicate relation. There exists a grammatical model of the NL sentence structure describing the non-co-ordinate relations among its constituents as predicative, objective, determinative, attributive etc. ones. Grammatical constituents corresponding to these relation types are not distinct in logical sense: the roles of these constituents can be interchanged (what was an object can be subject further on), hence all other differences can be neglected, in other words the functional sentence model can be based on one single generalized subject-predicate relation.

The peculiarities of entity/object division introduce the general framework for construction of the **language** both for semantic representation of a **query** and for semantic headers of an **answer**. To match a semantic representation of a query with a semantic header, a naïve approach would be to calculate the weighted sum of the number of different predicates and the number of different corresponding arguments for the predicates, which occur in both query and template. However, experiments show that more robust Q/A can be achieved by the straight unification of the translation formula with SH, traditional for logic programming (See , for example, Dahl 1999, Antoniou, 1997 and section 3 for more details).

This approach is required in addition to syntactic matching of normalized NL query with NL templates.

For example (fig. 1.4), the set of semantic headers for a document can be

{*format(HDparameters), upgrade(HDparameters, HDname), download(Where, SoftwareName)*}. At the same time, the translation formula may look like

upgrade(2G, robotics), download(Where, SoftwareName).

To perform the semantic matching, we need the reasoning step *install(…) :- download(…)* in addition to direct matching *predicate* → *predicate*, followed by *argument* → *argument,* which is indeed the unification in traditional logical programming.

NL understanding problem can be reduced to classification problem (Fig.1), if the cardinality of document set is much lower than that of the set of all possible semantic representation of input queries. The latter set of all semantic representations, which depends on the problem domain coverage by the particular semantic model, is in its turn much smaller than the original set of all possible queries in NL. The decision, whether a given domain with the particular semantic model falls into classification or understanding model approach for Q/A, should be made given the ratio

# of possible semantic representations : # of all documents,
(but not the ratio *# of possible NL questions : # all documents*). If the ratio is low, than the Q/A domain is subject to classification. If it is of the magnitude of 100-1000, then the NL understanding approach should come into play, where an answer is obtained as a value of predicate's parameter.

In contrast to the syntactic match extended by synonyms analysis, semantic one possesses the flexibility of matching multiple entities in a query. If one uses the approach of syntactic templates only, there should be a specific template for each single entity and for each pair of entities, if we enforce the system limitation to be not more than two entities per query. If the number of templates rises too fast, than the confidence level distribution over templates becomes too flat and the accuracy of classification significantly decreases.

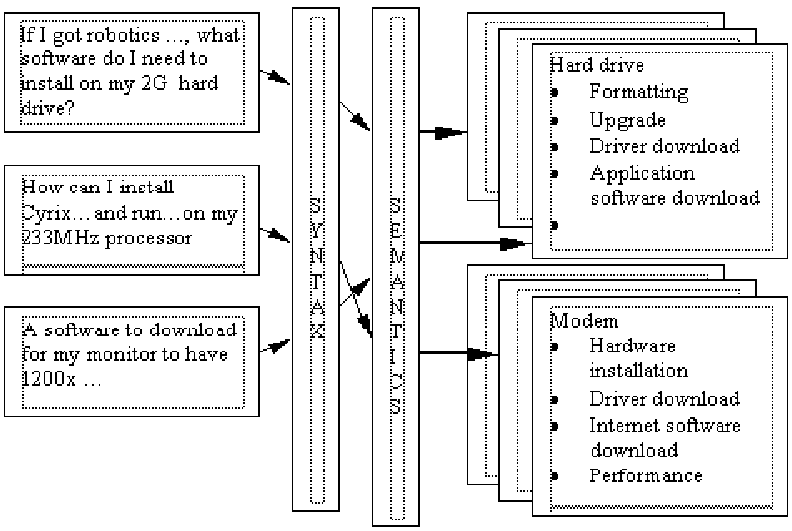

Fig 1.4: NL Q/A is posed as a problem of matching a question with an answer from the priory prepared set. Input query is to be assigned to a set of classes (documents). Assignment is performed based on the matching of semantic headers of the document with the semantic representation of the query, subject to the syntax and semantic analysis (from left to right). Note that the questions (on the left) are arbitrary, and the answers are structured in the certain way. Therefore, the matching needs to take into account the way the answers are organized, involving reasoning and semantic means.

2 Technique of semantic headers

2.1 Vertical domains for question answering

The intention of matching translation and representation languages requires the specific insight. For example, a better translation for the sentence "Mike wants to own a car" would be the formula
want(mike, ∃Car own(mike, Car)) in comparison with just
want(mike, own(mike, Car)),
where we put the special attention on the existential quantifier over the variable for an object *Car*. The scope of this existential quantifier lies within the metapredicate *want*. Its second argument (metavariable) ranges over the abstract well-formed formulas. In our example, this formula is *∃Car own(mike, Car)).*

The naming conventions of the PROLOG extension towards the **metalanguage support** (MS) are in use in this paper (See Galitsky, 1998 for more information). Variables are capitalized, existential quantifiers are assumed by default. Metavariables range over formulas, the interpretation of a metapredicate is the syntactic constraints for these formulas. Because of the frequent use of metaprogramming technique in the implementation of the semantic analyzer, we call it metalanguage-supported natural language system.

We illustrate that the translation quality is rather determined by the domain knowledge representation than by the inquiry representation, taken separately. In the case, where our domain includes the enumeration of all Mike's intentions (what he *wants*) and all the available cars, default existential quantification, inherent to the PROLOG Horn clauses, are sufficient to achieve the compatibility, getting the possible instances of the variable Car. The metapredicate *want,* whose semantics is implemented as a syntactic analysis of its second argument, does not necessarily have to involve the quantifier in its satisfaction procedure (the search through all cars). Furthermore, we could build even more detailed translation of this sentence, involving temporal features, saying that the intention occurred before than the fact of ownership holds,

before(want(mike, ∃Car own(mike, Car)), own(mike, Car)), or even applying Mike's mental state such as

know(mike, before(want(mike, ∃Car own(mike, Car)), own(mike, Car))). It is clear, that this additional information does not affect the compatibility with the domain representation and thus can be ignored. To conclude, we state that the language **expressiveness** of the NL **inquiry representation language** should **match** that of the **domain representation** one. Besides, the expressiveness of this formal language is determined by the capabilities of the semantic analyzer. Meeting these conditions allows one to implement the compatibility feedback and to assure proper handling of logically complex inquiries in the flexible and expandable domains.

Generality criterion is important for the information extraction component as well as for the question answering one. The architecture for generality control is traditional in the sense of "cascading", separation of processing into several stages. The earlier (syntactic) stages, which recognize the smaller linguistic objects and are based on domain-independent rules, are not subject to the generality control. The later (semantic) stages take the linguistic objects as input and build the domain-specific patterns, which are controlled based on the generality criteria.

2.2 Posing NL understanding problem in a vertical domain

The key feature of NL understanding system we intend to achieve to function in SH domain is the **flexibility** degree in respect to the **different domains** (Galitsky, 1998). The current study investigates the Q/A, which can potentially be adjusted to the particular by a customer, without the help of the system designer. It is performed by defining a new fact, a new object and a new entity via NL. A new concept is expressed via the old ones, already known to the system. It is evident, that the more degree NL understanding is based on the advanced reasoning rather than on the detailed morphological and syntactical analysis, the stronger flexibility features in respect to PD we expect to obtain.

The understanding process under SH technique is divided into two stages: **translation** into formal representation (reduced traditional step) and its **adaptation** towards the maximum compatibility with the PD representation. We skip the traditional morphological and syntactic analysis (MSA) and focus on the semantic one, requiring rather expressive formal language and reasoning capabilities. We describe the construction of all the possible translations given the MSA results and present the reasoning mechanism, which leads to obtaining the highest likelihood translation).

We proceed to the definitions of some concepts, specific to our semantic processing. An argument of a predicate can be instantiated (substituted by an object) or non-instantiated (not substituted). We call the **instantiation state** of *n*-ary predicate a binary function, which maps each predicate argument into the set *{instantiated, non-instantiated}*. There is a subset of the set of all 2^n instantiation states when this predicate always fails (independently on the constants, substituted to its non-instantiated arguments). We call this subset the **failure instantiated set** of a predicate in a logical program. For example, if a predicate were represented by a table of a database (a set of predicates is represented by a relational database), then any instantiation state of this predicate would lead to the satisfaction. In such a situation, each instantiation state corresponds to the specific SQL query. The binary predicate *more*, which takes two numbers and succeeds if the first argument is less than the second, has the failure instantiation states *<instantiated, non-instantiated>*, *<non-instantiated, non-instantiated>* and *<non-instantiated, instantiated>*.

We introduce the set of all **semantic types** of the objects. This notion is analogous to the data types in the conventional programming language; type hierarchy, derived types and casting are valid. A domain can be characterized by the family of all semantic types necessary to present it. Each argument of each predicate is assigned by the semantic type from the set of all semantic types. The **pattern** of a predicate is the enumeration of semantic types of its variables; a predicate can have multiple patterns. We distinguish the specific semantic type for **metavariables**, it can be instantiated by an arbitrary formula. We reduce our semantic model to a single predicate expression, which is substituted to a metavariable. For example, the phrase "Iraq wants to attack Kuwait" is translated into *want(iraq, attack(iraq, kuwait, Year)).* The pattern of the predicate *want* is *want(country, abstract)* and of the predicate *attack* is *attack(country, country, year).* Note that various patterns of a predicate can have distinguishing number of arguments (arity).

The domain is called **narrow** one if it could be represented by the set of predicates (the key concepts) such that each of them has just one sequence of semantic types for their arguments. In the other words, each predicate has just one meaning in the narrow domain. To express the superposition of the meanings (the context of a predicate, yielded by another predicate), it is sufficient for two predicates to have a common variable. There is a possibility of multiple superpositions (meanings) of a set of predicates within the narrow domain.

Belonging to a narrow domain can be determined by the presence of the set of predicates. Functioning in the set of overlapping narrow domains is possible; pre-recognition is required, which carries out the above procedure. Besides, a narrow domain can be represented as a set of all semantic types for participating predicates.

The domain is called **small** one if there is more then one meaning for a predicate, but they can be ordered in terms of the number of constraints (generality) in the predicate definition. Two definitions of the same concept can be ordered, if there is a set of terms in one of them that does not occur in the other. For example, consider the pair of definitions for the concept "Who informs Whom"

inform(Who, Whom, What) :-
want(Who, know(Whom, What)),
believe(Who, not know(Whom, What)),
believe(Who, want(Whom, know(Whom, What))) and
inform(Who, Whom, What) :-
want(Who, know(Whom, What)),
believe(Who, not know(Whom, What)).

The first definition specifies that *Who* wants *Whom* to know, believes that *Whom* does not know, and believes that *Whom* wants to know *What*. The second one does not involve the third constraint (*Who* might believe that *Whom* does not want to know *What*). Therefore, the second definition is more general (*Who* is allowed to have or not to have a certain belief). Note that each predicate has a single pattern within a small domain.

We call the domain **pattern-resolvable** one, if there are multiple patterns for the predicates, but it is always possible to choose the proper pattern based on the arguments, occurring in the inquiry and on the satisfiability verification. For any instantiation state for a predicate, obtained for an inquiry, there is the pattern that matches this instantiation state better than any other pattern in the sense of the number of substituted arguments. The choice of the proper pattern requires the feedback loop in the understanding procedure and therefore is inherent to our system architecture. A domain, which is not pattern-resolvable, is subject to understanding errors at least because of the choice of a wrong pattern.

2.3 Semantic types of arguments

How to choose semantic roles for a predicate? To estimate the set of words that forms the totality of specific meanings for an entity in a domain, one needs to observe all the answers whose SHs involve this entity. SH approach requires obtaining as many meanings by means of adding the attributes to an entity as necessary to distinguish given set of answers. SH approach is based on as detailed analysis as required by the given domain in contrast to traditional linguistic analysis systems, which usually do their best for building the full parse tree.

The choice of semantic types for a predicates varies from the fully-formalized domain to SH domain. The former one usually allows to choose the semantic types covering the uniform set of objects or attributes. For example, the geographic domain uses the semantic types of *xcity, xcountry, xcapital, xpresident,* etc.; each of them ranges over the similar kind of objects. In the fully formalized domain, the semantic types serve to express the semantic link between a pair of predicates: two predicates are linked if they have the common variable. Only variables of the same semantic types can be set equal for a pair of predicates.

For example, to represent the query from the geographical domain "Which capital does the country with president named Blair have?" we involve two predicates *capital(xcountry, xcity)* and *president(xcountry, xperson)* in the translation formula *capital(C, Xcity)*, *president(C, blair).* The semantic link between these predicates is expressed via the common variable over countries *xcountry.*

In SH domains, **semantic types range over all attributes required to separate the answers**. Semantically linked predicates are rather substituted into each other in accordance to the intuitive "importance" parameter and/or the classification than are assigned a common variable. For example, expression "investment in stock" gives us the translation formula *invest(stock(_)).* Both entities are predicates: *invest(xinvest), stock(xamoney*), semantic type *xinvest* includes *stock* as a value to specify the substitution. Both these entities can be in use separately, however, to form the meaning for a particular answer, one does not need additional explicit variables. So in SH domains we just substitute the latter predicate into the former one. This choice is based on the "one-to-many" assertion: one can *invest* in *stock, mutual_fund, IRA, 401k*, etc. Notice that the opposite choice can be motivated by a set of actions applicable to the options: to *invest*, to *sell*, to *buy*, to *analyze*, etc. The reader may take into account the fact that more specific meanings can be expressed by substituting an attribute into the internal predicate. For example, query "How efficient can be investment in stocks" is represented using the attribute *efficient* (which belongs to semantic type *xamoney)*: *invest(stock(efficient)).* The name of semantic type *xamoney* may seem to be unrelated to the term *efficient*; other semantic types can be efficient or inefficient as well. Families of semantic headers for the given set of answers are formed based on the necessity to separate these answers; therefore *efficient* could potentially be a value for the totality of semantic types. However, in a given domain it is the case only for a limited number of semantic types.

How many arguments should a predicate have? If we have a reasonable query, where more than one words needs to be substituted into a predicate, then its pattern should include semantic types for both these words (constants or predicates). For example, when we have a query "Can I deduct expenses on my house repair if I am a head of household" we conclude that the predicate *deduct* needs to have at least two arguments which will range over the "deduction items" and the "category of deducting individual". Additional argument is required to distinguish the "How to report…" , "What to report …", "When to obtain", etc. kind of questions, which we call the **focus of question**. Finally, we derive the following pattern *pattern(deduct(xdeduct, xcategory, xquest, xscore)).* The last argument is reserved for the number of answer and serves as a means to link two predicates.

A good example of the choice of arguments appears in the child psychology domain. The basic predicates are the psychological/behavioral states. The first argument of these predicates is substituted by the predicates for reaction of a teaching agent (e.g. a parent) to that state. It is very important to separate the **states** of the agents (teachers, parents and their children) from the reactions (actions), which are advised to follow these states. The state predicates The query "Should I punish my son
In the financial domains, the division of predicates into state and action ones is not essential, the former can be an attribute of the latter and vice versa. "Should I appeal an audit decision if it requires to pay more tax than I owe?" can be represented as audit (appeal(), tax(pay(), more

How many semantic types does a domain need? Usually, a knowledge engineer chooses the basic entities of a domain and derives the semantic types to separate answers relying on these entities. The motivation for the basic entities to have the distinguishing semantic types is that if a sentence include two or more of these predicates and a common attribute, a conflict will arise which predicate should be substituted. Additional predicates acquire new semantic types to avoid this kind of conflicts.

Semantic type ([list of objects of this type])	**Predicates with the argument of that type**
xamoney([ira, home, protect, safe, creditor, back, IRS, retirement, fee, home_equity, mortgage, personal, cash, non_cash, property, carryover, recover, distribution, income, capital_gain, self_employment, donate, charity, growth, file, discharge, allocate, uncollect, report]).	money, loan, lien, charity_contribution, cost, compensation, commission, lump_sum, foreign, stock, bond, CD, MMA, bankruptcy , funds , mutual_fund , saving , tip, alimony
xlegal([period,audit,collection,passive,involve,bankruptcy,resolve,notice,appeal,convert,eira,ira,tax_credit,advance_tax_credit,taxpayer,totalization,separation,eligible,education,clearance,depart,lost,adjusted_basis])	limitations, dispute, tax_court, proof, right , agreement, bluff, request, spy, institution, record
xatax([work,foreign,alien,convert,benefit,ira, resident,avoid,decrease, capital_gain,partnership, owe,over_withhold,social_security,medicare,form, estimate, sole_proprietor,contractor,wage,bank,report, calculate,business, interest,dividend,claim,test,home,non_deductible,abusive, collection]).	tax, levy, affedavit, self_employment_tax, preparer, dependent , income_tax xcategory, property_tax,transfer_tax, stamp_tax
xatransaction([deduct,deduction,property_tax,form,additional,return,pay, auditor,audit,collect,collection, bankruptcy, examination, accept,spouse, cash,problem,alien,income]).	force, challenge, appeal, company, prevent, offer_in_compromise, disclose, relief, extension, complete, winning, resolve, depart, defer
xaproperty([refinance,end,early,build,temporary,condominium, gift,divorce,inheritance,death,improvement,repair,selling_price,price,foreclosure, repossession,abandonment,debt,transfer,spouse, less,more,cooperative, previous,depreciation,remainder_interest, columbia,mcc,pay,first,base, estimate,community,personal]).	home, property
xmental([keep,record,tip,report,return,annuity,time,ssn,itin,clearance, qualify,medical,education,use]).	need, purpose, discourage
xasell([stock,property,hobby,stamp,coin,collection,home,long_term, short_term,car,computer,boat,medical_insurance,ira]).	sell, trade, buy, installment_sell
xafile([disclosure,april15,on_time,safe,fast,electronic,tele,credit_card]).	File
xaira([open,flexibility,candidate,collateral,non_deductible,college,penalize, roth,annuity,group,spouse, marriage,mix,inheritance,death,widow, simple_plan, traditional,more,medical_insurance]).	ira,educational_ira, rollover_ira
xatest([citizenship,relationship,gross_income,ira,support,single,head_of_household,marriage,widow,exclusion,deduct,deduction, distance,time,alien,substantial_presence,green_card,bona_fide_residence,physical_presence]).	test
xmoneyaction([more,points,higher_education,periodic,tax,owe, later,property_tax,monthly,	pay, expense, refund, fee

medical_insurance,on_time,delay,debt,medical,care, accident, health,unable,advance_tax_credit,sick, child_care,child_support, disaster,settlement,advance,penalty,credit_card,alimony,spouse, separate_maintenance,self_employment_tax,move,hobby,sell,business, non_deductible,other,double,commission,mortgage_credit, mortgage_interest,expect,long]).	
xlimit([travel,tax_credit,withdraw,deduction,deduct,gross_income,capital_loss,require,exclusion,mortgage_credit,distribution]).	limit
xdeduct([ground_rent,penalty,mortgage_interest,gain,loss,stamp_tax,transfer_tax,expense,refund, property_tax,contribution,earned_income,foreign, IRA,income,claim,bad_debt,business,interest,points, itemize,standard, money,gambling_loss,pay,medical,personal,mortgage_credit,home,tax, association,eira,charity_contribution,unreimbursed_employee_expense, alimony,child_support,invest,more,receipt,cancelled_check,income_tax,charity,property,dependent,allow,marriage,decrease,amount, miscellaneous, sell,withdraw,distribution,maximize]).	deduct
xincome([sell,installment_sell,trade,home,withdraw,bad_debt,determine,long_term,decrease,hobby, short_term,future,self_employment,net,distribution, current,commission,foreign,tax_deferred,nontaxable, award, property, less,illegal_income,debt,gift,meal,lodging,expense,severance_pay,lump_sum,benefit,pay, wage,winning,lottery,tip,alimony,farming,business, sole_proprietor,director,contractor,church,rent,fishing, partnership, interest,dividend,carrier,official]).	capital_gain, capital_loss, gain, loss, taxable_earned_income, earned_income, gross_income, taxable_income, self_employment_income
xdebt([more,money,irs,installment,child_support,worthless,business, nonbusiness]).	owe, debt, bad_debt
xdoc([joint,audit_proof,on_time,delay,paper,alien,nonresident,resident,electronic,tele,child,advance_tax_credit,installment_sell,sent,track,credit_card,explanation,without,lost,respond,incorrect,penalty,new,apply]).	return, letter, receipt , affidavit, notice, mcc, clearance
xcontrol([request,recieve,notice,record,date,mail,decision,abusive,against,before,form]).	audit, examination
xcredit([learning,dependent_care,claim,more,divide,forward,unused,refinance,amount,ira]).	credit,mortgage_credit, hope_credit,
xchild([qualifying,age,interest,dividend,income,minor,one,two,claim]).	child
xage([contribution,distribution,withdraw,eira,ira,under]).	age
xbusiness([sole,report,start,choose,defend,protect,penalize,contractor,subcontractor,milage,hobby,categorize,plan,home,income]).	business, self_employee
xirs([approve,recommend,download,calculate,contact,dispute,question,e_mail,address]).	Irs
xusetest([period,home,disability,condemn,destroy,marriage,divo	use_test, ownership_test

rce,spouse,death,business,rent]).	
xconvert([money,funds,ira,non_deductible,deductible,business,eira,stock]).	Convert
xwork([risk,income,invest,contribution,find,contact,report,visit,recategorize, minister,newspaper,tools,use, independent]).	work, activity, doctor, director, hobby, farming, church, fishing, partnership, carrier, official, contractor, sole_proprietor
xpension([other,plan,type,tsa,civil_service,federal,government,simple_plan,plan_401k,keogh_plan, tax_deferred, commersial, private, qualifying, simplified_method, general_rule, non_qualifying, use]).	pension, annuity, retirement
xalien([dual_status,both,resident,start,status,nonresident]).	Alien
xpenalty([free,tax,marriage,cancel,charge,mortgage]).	Penalty
xinvest([safe,cash,risk,defer,eira,fund,money,stock,ira]).	invest, contribution, investment_income
xbenefit([sick,unemployment,union,local,convert,ira,child,blindness]).	Benefit
xtaxcredit([education,child,foreign,claim,rule,buy,mortgage_credit,recieve, change]).	tax_credit, advance_tax_credit

Table 2.3: We use the extensive example of the entities and their attributes to show that each semantic type contains the words with high variety of meanings and different parts of speech. It is evident that if one would want to divide each semantic type into intuitively uniform components and create additional arguments correspondingly than the instantiation would be very sparse and the coding inefficient.

For example, the predicate *file(xafile, xquest)* is the basis of derivation of the following 8 meanings in accordance to the values of semantic type xafile: *disclosure,april15,on_time,safe,fast,electronic,tele,credit_card.* Note that these attributes are far from being uniform: they include the legal actions, specific dates, adverbs, ways of filing and payment. A natural approach to semantic types of an action like *file* would be

1. agent, performing filing;
2. subject of filing (document);
3. temporal attribute of filing (on_time, delay, etc);
4. general attribute of filing (additional information on filing)

However, this approach is inefficient for encoding the SHs of a real domain. The number of documents, which are supposed to be distinguished based on the attributes of a predicate, is rather limited. Our experiments of the building of SH domains show that less than a thousand documents are sufficient to cover a vertical domain for an average user; therefore, the number of attributes to separate the answers for a predicate is far below a hundred. Division of attributes in accordance to semantically uniform subsets would make the domain encoding rather inefficient. So we merge the proposed above semantic types into *xafile*, specific for the particular predicate. Furthermore, we add the specific semantic type *xquest* to distinguish between answers for "what is…", "how to calculate…", "how to respond…", "where to report…", "when to report…", etc.

Let us consider the critical cases of the ratios between the number of semantic types and the number of predicates. Since we do not need to have the same semantic types for different predicates, each predicate can be assigned by a unique semantic type. The more semantic types we choose for domain, the easier it would be to avoid conflicts, substituting a constant into either predicate if a pair of them occurs in a query. At the same time, high number of semantic types for a domain leads to the repetition of the same value for multiple semantic types, which implies the worsening of the structure of classification tree(section 4). The exhaustive search of argument substitution becomes less efficient.

If we minimize the number of semantic types and try to achieve a situation when an atom occurs only in a single semantic type, the number of substitution conflicts will dramatically raise. Our experiments of the creation of the suite of financial domains show that it is impossible to avoid the repetition of an atom in a vertical SH domain.

2.3 Family of semantic headers for an answer

A semantic header is intended to express the semantic link between a sequence (pair, triple) of entities, inherent to a given portion of information in an answer rather than a specific way of asking about it. So the number of semantic headers is supposed to be significantly less than the number of ways of asking about this answer. In other words, the following assertion holds

# of semantic headers < # of ways of asking with the same superficial semantics < # of syntactically different ways of asking.

Construction of the family of SH for an answer combines the following goals:

to represent the answer content from the perspective of potential querying

to provide the accessibility to an answer for as much possible ways of asking as possible.

Let us consider the following example for the answer #50400 (Fig 2.4).

pay(taks(social_security,_,_),50400):-do(50400).
taks(social_security,_,50400):-do(50400).
This is the answer about "social security tax", in particular, about "paying" it.
information(social_security,50400):-do(50400),do(10240).
This SH is less specific and involves another answer. So if one is interested in the social security issues in general, both answers #50400 and #10240 are relevant. If the question just addresses social security in connection with the tax issues, then #50400 is supposed to be sufficient.
retirement(railroad,50400):-do(50400).
This is another topic the answer raises.
wage(withheld(X,_),50400):- nonvar(X), X=taks(Y,_,_),
  nonvar(Y), Y=retirement(railroad,_), do(50400).
"withholding tax from wage for retirement of railroad employees" is the third topic of the answer #50400. The sequence of constraints is represented by the simplified formula
wage(withhold(tax(retirement(railroad))))) which cannot serve as a SH, because it should neither match, for example, with TF *wage(withhold(_))* nor with TF *wage(withhold(tax(_))),* because the other unrelated answers will be initiated in addition. Therefore, built-in predicates *nonvar* is required to filter out the unrelated answers.
withheld(taks(X,_,_),50400):-nonvar(X),
  (X=retirement(railroad,_); X=social_security), do(50400),!.
This is the similar way of asking about the same as above, but without mentioning "wage". If this SH is satisfied, the search for other possibly related answers is terminated (by cut).
excess(credit(_,_),50400):-do(50400).
This is the third item this answer addresses.

Table 2.3: enumeration of SHs for the answer #50400.

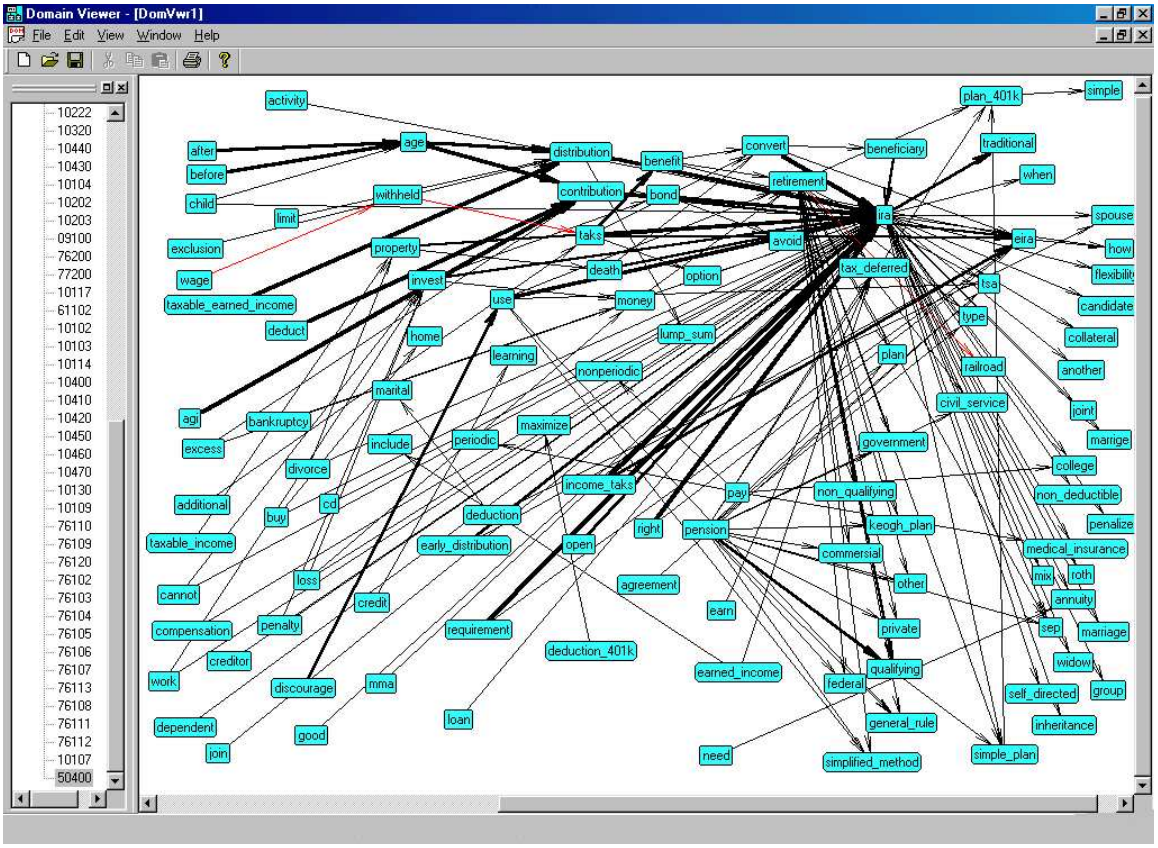

Fig 2.3: The domain visualization by DomViewer, the Windows software for knowledge engineers, experts and domain testers. On the right, classification graph is displayed; the nodes are assigned with predicates and edges with their mutual substitution to form semantic headers. The thicker the edge, the more times the different paths for the whole domain (here, IRA) contain this edge. In other words, the edge thickness visualizes how many times the corresponding subformula occurs in the totality of SH for the whole domain. Evidently, the majority of edges convert to the predicate *ira(…)*.

The toolkit automatically distributes the nodes throughout the rectangle to minimize the number of edge intersections.

On the left, the answers are enumerated. The paths for given answer are shown in red. In particular, answer # 50400 has the paths *wage-withheld-tax*, *ira-railroad and excess-credit* (not shown). Judging on these entities, one can reconstruct the meaning of the answer, encoded using SHs. The actual answer is as follows:

"Most employers must withhold social security tax from your wages. Certain government employers (some federal, state, and local governments) do not have to withhold social security tax. If you work for a railroad employer, that employer must withhold… You can claim the excess as a credit against your income tax…"

Frequently, encoding an answer by the family of semantic headers looses the logical links between various issues, addresses in this answer. Since the expression of interconnections between the paragraphs in an answer is outside of formalization scope, the members of SH family for an answer are not always formally correlated.

2.4 Semantic headers for an entity

How can an entity be represented by a family of SH? Which way do SH express the spectrum of meanings for an entity, significant for given domain? It occurs by means of attributes, substituted into the predicate for this entity and by substitution into the other predicates. Various denotations of a predicate are expressed via combinations with various parts of speech. Usually, it is insufficient to use the attributes of the same part of speech for an entity, even for a very narrow domain.

The following needs to be taken into account to distinguish answers involving the same entity. One needs as many SH containing an entity as required to separate these answers. If there is a single answer, introducing or explaining an entity, as well as semantically linking it to the other entities, then a single SH is sufficient for a domain. There is a series of mechanisms of building more specific SHs when multiple answers are characterized by the same entity:

- Attributes of the main entity, inherent to an answer.
- Logical links to other entities, which vary from one to another answer within a highlighted manifold of answers.
- Question words, which allow to distinguish, for example, “where” and “when” answers (“Where to send tax returns?” / “When are tax refunds sent back?”. As we have mentioned, the special semantic type *xquest* is introduced to separate this kind of questions.
- Defining one SH via another SH, adding/elimination entities or even substituting the foreign ones.
- Using the procedure “Clarify” to encourage the customer to replace/add the attribute, known for the system to obtain more specific answer instead of the general one.

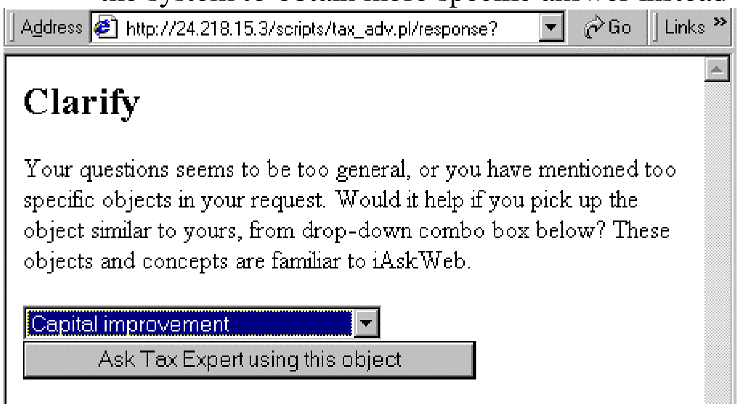

Fig. 2.4: A customer is suggested to clarify the situation, where the system understood the general entity in a query, but is unable to perceive a parameter for this entity. When a customer asked whether it is possible to deduct a roof repair, the system understood that the question was about a deduction of some object, unknown for it. Therefore, the system is suggesting the list of objects which are supposed to be relevant to what has been asked.

gain(X,_,_):-var(X), clarify([home, remainder_interest,
trade,transfer, installement_sale]), !.

If a query contains gain without parameters (or with unknown parameters), the customer is suggested to clarify (to choose from the drop-down combo box).

gain(sell(P,_),_,77114):- nonvar(P), sell(P,77114).

The line above reduces one SH to another. Document about the gain on a sale is the same document, containing the general information about sales. Argument of the predicate *sale* should be instantiated.

gain(sell(home(X,_,_),_),how(calculate,_),82103):-var(X), do(82103).
gain(sell(home(X,_,_),_),how(calculate,_),82107):- nonvar(X), home(X,_,82107).

If we speak about calculation of the gain for a sale of home, the situation depends on the attribute of this home. If there is no attribute (var(X)), then the answer “To figure the gain (or loss) on the sale of your main home…” is initiated. If there is a specific home attribute (a word, forming a specific meaning together with the entity *home*) such as *remainder_interest, improvement, inheritance, temporary, condominium, selling_price*, the answer is different. In contract to the previous SH above, the *gain-sell-home is* reduced to *home* and not to *sell*; in addition, instantiated argument of *home* is required to obtain the answer about “Foreclosure or repossession / Ordinary income”.

gain(sell(home(_,_,_),_),_,82116).

SH above is required when we build the translation formula with two or more conjunctive members with the common answer id (here, 82116). This SH is design to be matched against one of these members; this match will not occur if another answer id will be generated (verified) by the previous (following) conjunctive member(in the translation formula).

For example, the query “how to estimate the gain from sale of my house if I have used it for business” is represented as

gain(sell(home(_,_,_),_),_,Id), business(use(home(_,_,_),_),_Id);
SH for the second conjunctive member contains the reference to the answer (do(82116)) and the first conjunctive member introduces the constraint via the answer id (SH above).
gain(sell(home(_,_,_),_),when,82121):- do(82121).
gain(sell(home(_,_,_),_),how(report,_),82122):- do(82122), do(82125).
gain(sell(home(_,_,_),_),example,82125):- do(82125).
SHs above introduce the difference between the questions “when to do something” and “how to report something”.
gain(sell(home(X,_,_),_),_,82128):-nonvar(X),X=remainder_interest, do(82128). The particular case of the family of SH above for remainder_interest.
gain(trade(home(_,_,_),_),how(calculate,_),82106):-do(82106).
We have different answers for *sale* and *trade*, though their occurrence is symmetric.
gain(home(X,_,_),_,82107):- nonvar(X), home(X,_,82107).
“Gain for a home” is the same as “gain for a sale of home”, if the predicate *home* is instantiated.
gain(home(transfer,_,_),_,82108):-do(82108).
gain(home(spouse,_),_,82108):-do(82108).
SH above is employed when the *home* is associated with *spouse*. Intuitively, it is hard to imagine the meaning for the pair *home-spouse*, which is indeed “transferring home to a spouse”. Since multiple synonyms may occur instead of *transferring*, we would like not to add an entity or attribute for this kind of action. It is motivated by the fact that we currently do not need to distinguish *to transfer*, *to gain, to change ownership* to separate answers about gain.
gain(future,_,45400):-do(45400).

Table 2.4.1 Representation of an entity gain. See the examples of the queries for each answer in the Table 2.4.2 below.

82103	How to figure gain (loss) on sale of my home?
82106	How to figure gain (loss) when I trade home?
82107	How to figure gain in the case of home foreclosure (repossession)?
82107	How to figure gain if I abandon my home?
82108	How to figure gain(loss) when I transfer home to spouse?
82116	How about ownership (use) test if I used my home for business (rental) purpose?
82121	Do I need to report gain (on my tax return) if I sold my house?
82122	How do I report gain on sale of my house?
82123	What is an installment sale?
82124	How to fill in tax return when I had installment sale of home?
82125	Give an example of reporting gain on selling a home?
77114	What should I do if I sell stamps from my collection?
45400	How can classify current losses to produce future gains?

Table 2.4.2: Examples of the queries involving the concept *gain* for each answer in the Table 2.4.1above.

2.5 Processing of synonyms

To present the multiword/synonyms processing sequence, we apply two following replacements (using the predicate *abbrev*):

abbrev([restricted,access], restricted_access).
abbrev([adult,rated], adult).
abbrev([adult, only], adult).
abbrev([on, file], on_file).

These are just trivial replacements of a multiword. Second and third replacement may not be required, if the system can just ignore the words *rated* and *only* in a particular domain. However, there is a possible conflict replacing the multiword component. In a companion domain, *Internet order processing*, the word *access* might be replaced by the word *login.* To handle this possibility while working in the combination of two domains, the following (post) replacement is required:

abbrev_post([restricted,login], restricted_access).

Note that word substitution cannot handle the full spectrum of synonyms, including the semantic ones. For example, the following questions have the same meaning in our domain

What do you sell? and *What can I buy?* Note that modality and temporal information does not affect the meaning. The predicate representation *sell(what)=buy(what),* however it is evidently not true that *sell(Attribute)=buy(Attribute)* for any *Attribute*. We cannot use *abbrev([sell], buy)* replacement to express the fact that the queries above are similar.

2.6 Combining the meanings of entities

Throughout the paper, we frequently use the term "meaning of an entity (keyword, atom)". In this section, we present its definition in the SH environment and compare it with the speculations in linguistic semantics, where the "meaning" by itself can be approached just approximately, but the combination of meanings to form a new one is explored in details.

Evidently, construction of semantic header from the atoms is combination of meaning of the corresponding words. It is one of the fundamental problem in linguistics, how to form a meaning of NL expression combining the meaning of its lexical units. In contrast to traditional linguistic and semantic studies (see, for example, Boguslavsky, 1996, Partee, 1990), addressing the issues of semantic sphere, the theory of SH provides the strict background for this issue because the meaning of an atom is relative and the role an atom is solely to separate the answers. The meaning in SH theory has a "relative to domain" nature: the meaning of an atom is a set of answers this atom is in use to separate from the other answers. The meaning of a predicate (for an entity) is expresses via the set of all semantic headers including this predicate and only this set. Roughly, a meaning of a predicate is the set of these answers. A subject constant does not have a separate meaning by itself, it forms a meaning when it is substituted into particular predicate, reducing the set of answers the corresponding subformula enters into the SHs for.

We usually merge different parts of speech with the same root to a single entity predicate. To compensate the loosing of part of speech information, we employ the logical means to better analyze the context of the query. In other words, sacrificing knowledge about the form of the word, we rather establish its meaning by taking into account the other significant words, building the translation. The rules, how to involve the other syntactically dependent words in the determination of the meaning (translation) of the given one must be very flexible, therefore the default logic comes into play (section 7). For example, we represent the meaning of words "tax" and "taxable" by the same entity *tax*, because the is no such other entity (attribute) *E* in our domains such that *tax(E)* and *taxable(E)* (or *E(tax)* and *E(taxable)*) would be assigned to the different answers.

Sometimes it is important to form distinguishing predicates for the different parts of speech with the same root, for example, *deduct* and *deduction* in the tax domain, where these words are sometimes associated with the different answers. The entity of *deduct-deduction* is so important for the tax domain, that there are multiple situations with *deduct* as (an action of an involved agent) and *deduction* (as a complex relation between *expenses*), which must be separated. At the same time, for the rest of answers *deduct* and *deductions* are fully synonymous, so the corresponding pairs of SHs are required. In the process of developing an arbitrary domain, a knowledge engineer may need to split a predicate (entity) in accordance to different parts of speech when a new set of answers requires separation. Note that *deductible →deduction* rather than *deductible→deduct*, because semantically in our domain *possibly can be deducted → deduction* is closer to a relation then to an action: usually *deductible* is related to *expense*.

To derive a meaning, expressed by a translation formula, we combine up to four entities (the number of attributes is limited in a lesser degree). Usually, a sentence with higher number of entities can be split into two consecutive queries, and a user is encouraged to do so. To handle the unreasonably complex sentences with more than four entities, we would need to extend the number of arguments for the predicates, which leads to rather inefficient domain coding. Obtained predicate expression with more than four levels may mean that some syntactically separate entity is mistakenly involved, though the system may still perform robustly if some additional instantiation of argument does not affect the choice of the answer. Indeed, building semantic headers, independent on the eventual instantiation of arguments, is a good style of domain design.

N	Expression from a query	Fragments of syntactic representation for selected words	Translation formula
1	Decrease itemized deduction	decrease – deduction - itemize	deduction(decrease, itemize)
2	Deduct business expenses	deduct-expense- business	expense(deduct(_), business))
3	Itemize expenses	expense-itemize	expense(_,itemize)
3	Reduce business expenses	decrease- business- expense	expense(decrease, business)
4	Decrease tax deduction	decrease-deduction-tax	tax(deduction(decrease))
5	Decrease tax	decrease – tax	tax(decrease)
6	Deduct tax from income	deduct - tax, deduct - income	tax(deduct(income(_),_))
7	Deduct employee expenses from income	deduct –employee- expense, deduct – income	expenses(deduct(), employee), deduct(income)
8	Taxable income	tax – income	tax(income)
9	Deductible income	deduct- income	deduct(income,_)
10	Deduct property tax from income tax	deduct – tax - property, deduct – tax - income	deduct(tax(property), tax(income(_)))
11	How expensive is (to file) tax deduction	tax-deduction-expense	tax(deduction(expense,_))
12	Increase income because of increased deduction	income-increase, deduction-increase	deduction(increase, income(increase))
13	Deduction of income tax	deduction-tax-income	tax(deduction(income))
14	Increase income by reducing expenses	increase-income, decrease-expense	income(increase), expense(reduce)
15	Increase income from property	increase - income, reduce - expense	income(increase), expense(reduce)

Table 2.6: deriving the meanings from the combination of entities and their attributes. Analysis of the typical combinations of meaning of entities helps to choose the proper semantic types and interdependence between the predicates. For example, to choose whether to substitute *deduct* into *tax* or vice versa, one needs to consider the meaning of "to deduct" as "to subtract"; therefore the predicate deduct needs to have active and passive pair of arguments. We conclude, that *tax(deduct(_,_))* is a proper choice of predicate interdependence. For the predicate *expense* we choose two argument with semantic types of
Note that this table is simplified in terms of semantic types in respect to table 2.3 for the totality of semantic types for the tax domain.

As an example, in the table 2.6 below we present the expression in NL, combined from the set of words *deduction, itemize, decrease, increase, reduce, business, expense, tax, income, property* and their variations. A family of meanings can be expressed in this basis; syntactic relationship between this words helps to build translation formula (TF). Based on the set of words, we could derive much larger set of combinations using various forms of these words. However, to separate the documents, it is sufficient for SH technique to use just the normalized forms of words.

Note that each subset of our set represents the unique meaning in terms of potential answer separation; it is true for the majority of domains and their entities. Therefore, in most cases, the meaning of NL expression can be represented just by an unordered enumeration of entities (keywords) in a vertical SH domain. Encoding of these meanings by the predicate formulas serve the deductive and matching than representation capabilities of the overall NLP system. The situations with multiple meanings for an unordered set of atoms are subject to separate considerations, in particular, section 2.3 about meanings for an entity and section 7, which presents the applications of default rules. In general, meta-programming technique is required to build multiple SHs with the same set of atoms.

In the left column of table 2.6, we present the expressions, consisting from the set of words above and reasonable in our domain. In the middle column, one can see the clips of respective syntactic structures (pairs or triads of *predicate-object* relation chains). Each row contains one or two of these chains. Synonym substitution is applied, for example, in this domain it is sufficient to suppose that *reduce* → *decrease* to separate the answers.

The right columns presents the predicate language encoding of these expressions, where some of the words from our set are mapped into predicates, and the rest of words – into their arguments, which in turn can be predicates as well. For simplicity, some arguments are not shown.

Note that the order of the (normalized) words in the middle column follows the syntactic dependence between these words. However, the occurrence of the corresponding atoms in the right column (translation formula) is determined by the dependence between entities for the whole domain. The latter is implied by the structure of a domain: how the entities contribute to the recognition of documents (answers) for this domain. Representation of predicates is more complex in a real-world domain: additional semantic type *xquest* is sometimes required to distinguish answers (see section 2.3).

For our analysis, revealing of the parts of speech is required to obtain the link between the words; further coding loses this information as far as the context has been established. In that respect our approach can be referred as a reduction of the "real" syntactic analysis to keyword extraction. However, advanced deductive means are involved to compensate the loss of syntactic information: rules of semantic processing take advantage about knowledge of interconnection between the words in a query. Therefore, SH approach lies between the full syntactic analysis and pure keyword extraction in terms of the query processing, but requires rather sophisticated semantic means to derive the translation formula.

2.7 Use of context for construction of the translation formula

The problem of eliminating the secondary constraints can be naturally addressed in the bounds of the logical theory of contexts (Lenat & Guha 1990, McCarthy, 1993), more specifically – multidimentional context space. We would expect the set of entities for a query to fell into 2 categories:

1) Entities, pointing to the current context;
2) Entities, whose meanings are evaluated in that context.

It is clear, that these categories overlap in general case. The following pragmatic question arises, can one eliminate the entities of the first, but not the second category after the context has been determined without affecting the translation semantics? As computational evidence shows, it is not the case.

For the Tax domain, there is a traditional division of topics into enumerated contexts (so called "Teletax topics"). However such division, natural from the educational prospective, hardly appropriate in terms of the **uniqueness** of the **meanings** in each context. The companion requirement for the context as a means to verify the formula validity poses the different principles for context creation as well.

Experience, accumulated while constructing the set of contexts for the general world knowledge (Lenat & Guha 1990, McCarthy, 1993, Buvac, 1996) is found to be hardly appropriate for a narrow domain, where the general dimension of the domain space is significantly reduced and modified in accordance to the domain-specific features.

The formal context for the NL domain needs to be constructed based on the set of all semantic headers for the set of documents within this context. In this particular case, the semantic headers can be viewed as the essential facts, extracted from these documents.

How can we define the notion of **context**, when the setting of domains and subdomains has been established? **Domains** are distinguished based on informal, pre-formalized knowledge about their content, in spite of the fact that they can be identified based by a set of entities. On the contrary, contexts are the post-formalization divisions of domain knowledge, based on the formal expressions in the domain-specific language. It is very important for practical applications to relate context to a domain; the logical definition of context by means of lifting operator is insufficient for NL processing before the query translation has completed.

The **context dimension** is then defined as the set of predicates, irrelevant to the totality of other predicates. Context dimension is naturally revealed based on the semantic types; two distinguishing dimensions cannot have **inclusive** semantic types. Besides, some context dimension can be associated with some order on the values of their arguments.

In particular, highlighting of mental and non-mental (physical) dimensions brings in the powerful method of describing multiagent interactions in various physical worlds. The deductive means of mental and physical dimensions are very different; this fact strongly affects the reasoning machinery of multiagent system (Galitsky, 1999b). Division into order-based dimensions (temporal vs spatial, points vs fragments, etc.) is always necessary for correct domain representation. The following is the additional criteria for the utility of a dimension:

1. Successful separation of mutually-irrelevant portions of knowledge from each other.
2. Possibility to determine the intersections and unions of the dimension's fragments.
3. Division of large portions of knowledge into portions in accordance to the dimension's fragments.
4. Points and fragments (especially critical ones) of a dimension correspond to semantically sound and familiar real-world entities and considerations.

What does it mean, that a constant belongs to a semantic type? It means, that an entity together with this object (parameter) comprises the specific meaning with the document, presenting this meaning. In other words, this constant concretizes the manifold of meanings for this entity.

There is a variety of parts of speech for each semantic type. It is caused by the fact that different meanings are expressed via distinguishing syntactic structures. Note that there is significant overlap between the constants for various semantic types; therefore construction of the context dimension based on the semantic types should involve narrowing down or merging the overlapping semantic types.

From the knowledge representation language viewpoint, for each predicate the constants (both objects and predicates) of its semantic types are substituted into this predicate. This is the way to introduce the hierarchical dependence between the predicates. The semantic types and their constants have to be defined to disallow mutual substitution for a pair of predicates: if one can be substituted into another, vice versa should not be possible.

Natural question is, can one use the content of a book (enumeration of chapters and sections) as a source to extract the terms to represent and classify a domain? The answer is "No". Expressing a topic by a sentence does not necessarily introduce the keywords for this topic. Frequently, the way the content is built is hardly correlated with the terms the topic knowledge is represented.

3. Matching translation formula with SH

Matching the translation formula (TF) with SH is the key step of Q/A, where the recognized meaning of a question is compared with the pre-encoded meaning of an answer (see fig. 3.1). Match of TF against SH implements the following functionality:

- Answer generality/specificity control. Since we match a predicate formula and not the list of keywords, we take advantage of the formula unification. For instantiated TF, the system gives the specific answer, determined by the instantiation arguments; for uninstantiated TF, the system gives all answers for the TF predicates.

- Invariant properties in respect to morphological and syntactical specificity of query phrasing.
- Working in the non-uniformly encoded domain: some answers are assigned with the more complex SHs (areas with higher completeness within a domain) and other answers are assigned with a single predicate formulas (lower completeness sub-domains).
- SH have the full capability to express content. Some predicates may be synonyms in the context of a specific answer, but have a distinguishing meanings and must be separated to properly handle the other answers (for example, *cost()* and *expenses()* are synonyms in the context of *education*, but have the different sense when one speaks about *deduction*).

If the is no such SH that given TF can be matched against, than TF can be subject to **attenuation**, which obeys the monotonicity properties (see section 5.4) and can lead to approximately related answer in case there is no exact one or the proper SH is absent.

Use of meta-predicates (multi-level predicate expressions) in TF and SH is not only implied by the language expressiveness requirements. Since TF~SH match is based on the satisfaction call to TF (and not the direct unification) in logical programming, we rather match the meta-predicate expression *p(q(s))* than the conjunction *(p(q), q(s))*. The latter would be matched with two SHs assigned, generally speaking, with two different answers, which is not what we usually want. Therefore we define *p* to range over arbitrary formulas, which specify its meaning (or introduces the context). Note that the satisfaction call to *(p(q), q(s))* will first search to match *p(q),*and then *q(s)*; matching results can depend on the order. TF~SH match implementation based on built-in satisfaction call is more computationally efficient and convenient than the unification with the set of expression we would need to explicitly program.

In this section, we address the attenuation algorithm while matching TF against SH. There are two ways to achieve the situation when each possible TF can be matched:

SHs are designed to include all well-written formulas of the given complexity (ignoring *var()/nonvar()* constrains, see section 2.4), and there is no approximation of TF. In this case the problem of separation of answers arises because TF for general questions would cause too many answers.

SHs are intact, and TF should be sequentially approximated by attenuated TFs till unification occurs.

Formula attenuation procedure is defined as syntactic modification of formula, implying the minimal semantic impact on its meaning. The following transformations are allowed (in the order of the increase of semantic influence):

- resorting of terms in a formula;
- elimination of conjunctive terms in a formula.

For each of the terms, the following transformations are to be applied:

- elimination of outer term (for example, *approx((…, p(q(a)),…) , (…, q(a),…))*).
- addition of outermost atom (for example, *approx((…, p(q(a)),…) , (…, q(a),…))*).
- replacement of constants (attributes) (for example, *approx((…, p(q(a)),…) , (…, p(q(b)),…))*).
- uninstantiation of inner constant (turning into a variable, for example, *approx((…, p(q(a)),…) , (…, p(q(A)),…))*). It makes difference with 2) for SH clauses, using the meta-programming tricks (see section 3.2).

These transformations are obtained empirically based on the experiments in the domains with statistically significant numbers of SHs. However, the attenuation procedure obeys the monotonicity property, which can be formulated based on the numerical estimate of the transformation (distance between formulas).

The distance (non-symmetric, see section 5.2) between TF and SH is defined as a number of transformations, applied to TF to achieve unification with SH deriving SH^{approx} (by definition, $d(TF, SH)=d(SH, SH^{approx})$. Note, that approximation of TF can match with multiple SH, and the following **monotonicity property** holds.

For any TF, if one obtains two approximations TF1 and TF_2 such that

$d(TF1, SH1) \prec d(TF_2, SH_2)$, then $cardinality(\{SH1: TF1 \sim SH1\}) \prec cardinality(\{SH2: TF_2 \sim SH_2\})$.
Cardinality here is the number of elements in the set (of SHs). The distance can be estimated with the higher relevancy employing the technique presented in the section 5.

Note the difference between an abstract attenuation of a formula (Galitsky, 1998) and attenuation towards the target (SH), we use in Q/A system. The latter uses, generally speaking, less operations, but involves the addition of outermost atom, in contrast to the former type of attenuation

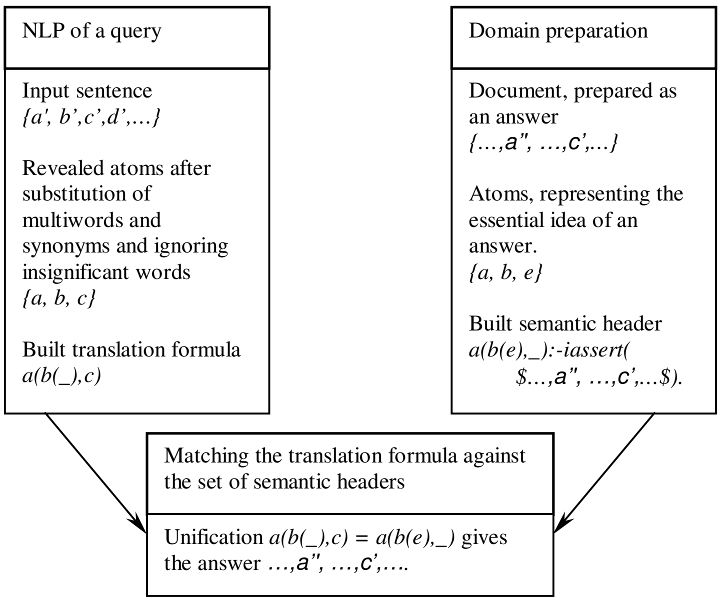

Fig 3.1 The information flow under SH technique. The input query is subject to NLP (on the left). The document is subject to the procedure of SH assignment, performed while preparation of Q/A domain. The essence of finding an answer is the unification of translation formula with all SHs of the domain. However, unification of the standard PROLOG is insufficiently flexible to search for semantically close headers. Extension of this unification is presented at Fig.7.2.

If it is necessary to attenuate a TF towards SH, the predicate importance data improves the accuracy of search for the closest SH. Attenuation monotonicity property implies that the stronger attenuation towards SH corresponds to the higher number of matched SH. To provide the more focused answer, one would prefer to attenuate the original TF minimally. However, since the resultant answers are expected to be just the approximately relevant ones, knowledge engineers find the compromise solution concerning the approximate match, when they design the domain SHs.

We sketch the Prolog implementations of some attenuation steps. Meta-programming technique is illustrated: for attenuation, a clause needs to be represented by a list, then some list modifications are applied, followed by the conversion back into a clause to verify its satisfaction.

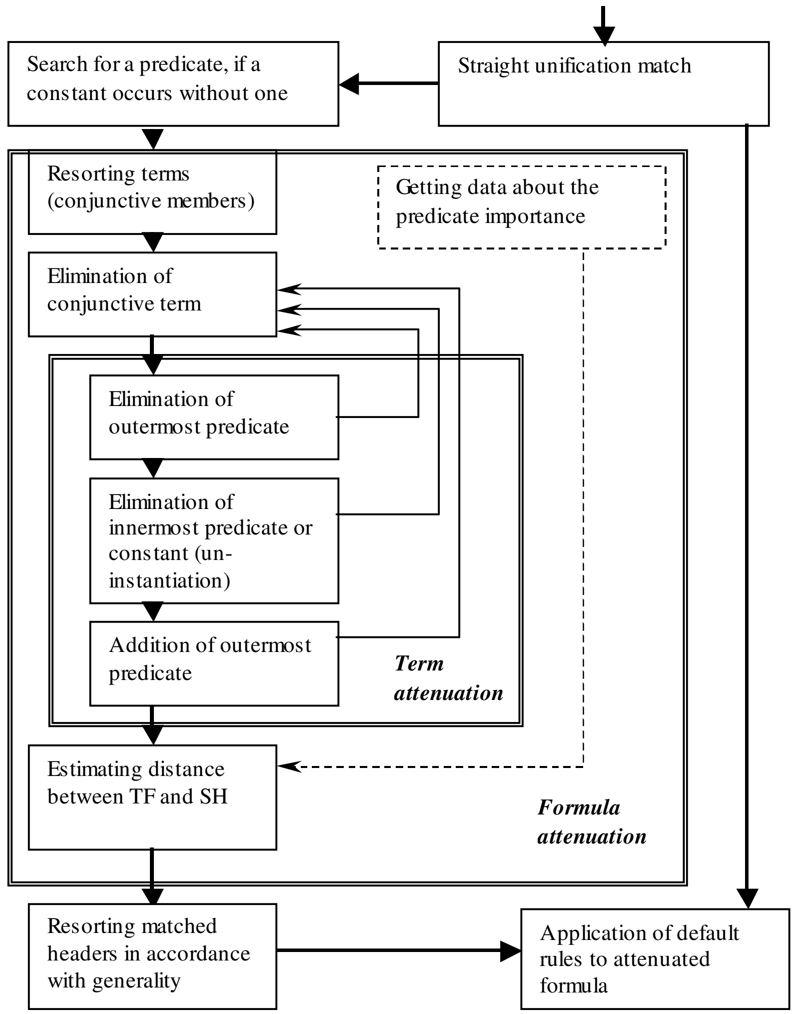

Fig. 3.2: The extended match of the translation formula with semantic headers. In case of failure of the direct call to the translation formula, it is subject to sequential attenuation (the next step is applied if the current one does not deliver the match of TF against SH). Attenuated unification match is followed by application of default rules, which are intended to eliminate the conjunctive members in spite of the fact that they may be satisfied (section 7.2).

Attenuation of formulas

Straight (no modification)

```
modyTranslFormula(QueElim,Ques):- var_list(Que,QueElim), var_var(Que,QueCheck), call(QueCheck), Ques=QueElim,!.
```

Reversing of terms

```
modyTranslFormula(QueElim,Ques):- nrev(QueElim, QueR), var_list(Que,QueR), var_var(Que,QueCheck), call(QueCheck), Ques=QueR ,!.
```

Attenuating each term

```
modyTranslFormula(QueElim,Ques):-  len(QueElim,L),L>0,
 findall( NewTerm, ( member(Term, QueElim), var_const(TermV, Term),
 [!((call(TermV),NewTerm=Term ); ( [! modyTranslTerm([Term],[NewTerm])!]
  )) !]), QuesD), QuesD\=[], remove_duplicates(QuesD, Ques), len(Ques, L).
```

Attenuation of terms

Eliminating outer atom

```
modyTranslTerm(QueElim,Ques):-
                QueElim=[Que], (Que=..[Extern, FullC|_]),
          var_const(FullV, FullC), call(FullV),
           Ques=[FullC].
```

Substituting attribute

```
modyTranslTerm(QueElim,Ques):- QueElim=[Que], (Que=..[Pred, FirstArg |Args]),
  pattern(Full), Full=..[Pred|ArgsFL],
  clause_list(ArgsF, ArgsFL), change_char('x','_', ppp(ArgsF), ppp(ArgsVC)),
  clause_list(ArgsVC, ArgsV), FullV=..[Pred | ArgsV],
  findall( ArgsV, call(FullV), ArgsVs),
  findall(FArg, member( ([FArg, _,_]), ArgsVs), Vs),
  Vs=[First|_], Qu=..[Pred, First| Args], Ques=[Qu].
```

Eliminate upper and enforcing uninstantiation of inner variable

```
modyTranslTerm(QueElim,Ques):-
  QueElim=[Que], (Que=..[Extern, RestOfFull|_]),
  (RestOfFull=..[Pred, FirstArg |Args]),
   pattern(Full), Full=..[Pred|ArgsFL],
    clause_list(ArgsF, ArgsFL), change_char('x','_', ppp(ArgsF), ppp(ArgsVC)) ,
    clause_list(ArgsVC, ArgsV), FullV=..[Pred | ArgsV],
    findall( (ArgsV), call( FullV), ArgsVs),
    findall(FArg, member( ([FArg, _,_]), ArgsVs), Vs),
    Vs=[First|_], Qu=..[Pred, First| Args], QuesV=[Qu], var_const(QuesV, Ques).
```

Addition of outer predicate

```
modyTranslTerm(QueElim,Ques):-
  QueElim=[Que], (Que=..[Pred| OldArgs]),
  pattern(ExternFull),
  ExternFull=..[ExternPred, Arg|ExternArgs], VRun=..[Arg, Values],
  call(VRun), member(Pred, Values),  NewExternC=..[ExternPred,Que|ExternArgs],
  var_const(NewExternVC, NewExternC),
  change_char('x','_', NewExternVC,NewExternV),
  var_var(NewExternV,NewExternVV),call(NewExternVV),
  var_const(NewExternV,NewExternVCC).
```

Table 3.0: the Prolog implementation of TF attenuation. We comment on the **service** predicates (as an opposite to the domain representation ones) , which implement the attenuation. The service predicate *change_char* replaces a character to covert the implementation-specific representation of a variable into the representation of a canonical constant. *findall* is a built-in Prolog metapredicate, which obtains the list of values (the third argument) for an expression (the second argument) given its parameter (the first argument). *Member* verifies or yields a member of a list. Operator =.. converts a term into a list.

3.1 Introduction to meta-programming technique

We present four basic service predicates, which are required to implement the semantic processing of translation formula.

Predicate	**Comments**	**Example**
var_const	Converts all the occurrences of the variables in an expression into the <u>canonical constants</u> such that for each pair of occurrences of the same variables the resultant corresponding constants are equal. A formula is subject to syntactic transformation, when all arguments are actual or canonical constants.	var_const(p(V1,V2,V1), p(v1,v2,v1))
var_var	Converts all the occurrences of the variables in an expression into the <u>variables with other names</u> such that for each pair of occurrences of the same variables the resultant corresponding constants are equal. This predicate is required to make a copy of a formula to check its satisfiability, not affecting the values of the	var_var(p(V1,V2,V1), p(U1,U2,U1)), *call(p(U1,U2,U1))*

	variables of the original formula.	
clause_list	Converts a clause into a list	clause_list((p(V1),q(V2,V1)), [p(V1), q(V2,V1)]
change_char	Replaces a character of a string representation of a formula. Particular in use to replace the initial character of a constant into '_' and back to convert it into variable (canonical constant).	change_char('x','_', p(xV1), p(_V1)). xV1-constant, _V1-variable.

Table 3.2: service predicates of clause/list conversion.

Transition from the strict to attenuated match is one of the issues of domain development strategy. On the **debug** stage, the approximate match is turned off such that the system can be verified to provide the exact answers for all questions, designed a priory. Knowledge engineers achieve proper answering of all questions by the exact match. The extended match serves as a basis for the **release** version, when an unexpected question is handled with the full power of approximation machinery.

4. Classification graph of Q/A domain

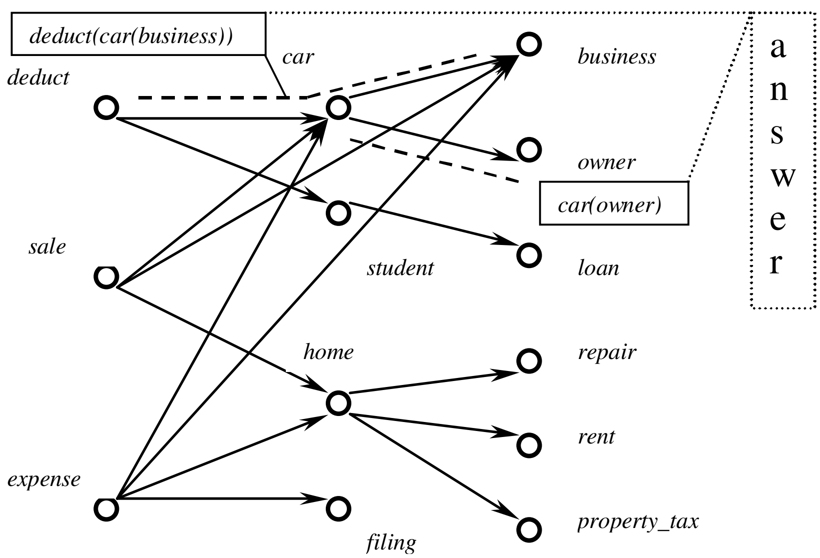

Fig.4.1 Classification graph for the fragment of a financial domain. The essential entities are chosen to divide the set of all answers *D* into three classes: about *deduction*, about *sale* and about *expenses*. The following subclasses are revealed: *car, student, home, filing;* three of these subclasses are further divided into sub-subclasses *{business, owner},{ loan},{repair, rent, property_tax}* correspondingly.
The nodes of the graph are assigned by a predicate; there is a single node for each predicate. The directed edge denotes the substitution of one predicate into another (arrow side). Semantic header is associated with a sub-graph of classification graph; in particular, for unary predicates, SH is associated with a path of the classification graph. For example, SH *deduct(car(business()))* is associated with the path *deduct-car-business* of the classification graph (shown as a bold dotted line). The answer is assigned by two SHs: *deduct(car(business()))* and *car(business()).*

The classification of a real question-answering domain is far from ideal: there are common sub-units for two different classification units of the graph (Fig.4.1). In the other words, the classification graph is rather far from a tree (if to ignore the direction of edges). In particular, there are common subclasses (*car, student*) for the classes *deduct* and *sale.* Furthermore, there are links between the nodes for classes and sub-subclasses. There is a common sub-subclass *business* for the class *sale* and subclass *car.*

Note that the division of entities into class-determining ones reflects neither parts of speech nor general meaning of the words. Inversely, the position of an entity *p* in the classification tree is determined by the semantic role *p* plays to separate given subset of documents (*p* occurs in SH) from the rest of documents.

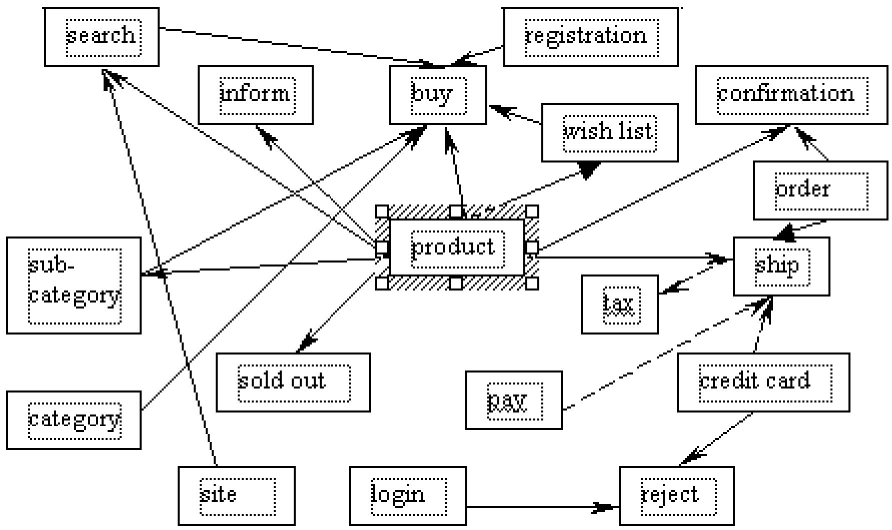

Fig 4.2: The graph of domain predicates (e-commerce retailer domain). The arrow link means that there exists a semantic header, where one predicate is substituted into another predicate. For example, the semantic header *login(reject(_,_),_)* is associated with an answer to the question *"What should I do if my login is rejected?"*. Note that since there are multiple semantic headers for an answer, it can be more than one node associated with this answer. For example, three SH *pay(ship(_,_),_), credit_card(ship(free,_),_) and ship(tax*(_,_),_) are associated with the answer *"Shipping is always free"* for the questions: *"How to pay for delivery?"*, "*Can I use credit card to pay for shipping?"* and *"Is tax included in shipping?"*

Fig 4.2: The graph of domain predicates (e-commerce retailer domain). The arrow link means that there exists a semantic header, where one predicate is substituted into another predicate. For example, the semantic header *login(reject(_,_),_)* is associated with an answer to the question *"What should I do if my login is rejected?"*. Note that since there are multiple semantic headers for an answer, it can be more than one node associated with this answer. For example, three SH *pay(ship(_,_),_), credit_card(ship(free,_),_) and ship(tax*(_,_),_) are associated with the answer *"Shipping is always free"* for the questions: *"How to pay for delivery?"*, "*Can I use credit card to pay for shipping?"* and *"Is tax included in shipping?"*

We address the important issue, whether the predicates for a verb or a noun should be at a higher level in a domain classification. It actually depends on the ratio *other_nouns : other_verbs*.
where *other_nouns ={V | noun(V)} and other_verbs ={N | verb(N)}*. It is rather natural, that the number of predicates is lower than the number of the object or predicates the former ones range over. In other words, the deeper the level of classification, the larger the number of nodes for the classification graph. For example, if a domain contains a few actions with products and thousands of these products, the actions will range over products. On the other hand, if a manifold of operations can be applied to a small number of products, the situation is opposite. The predicates or objects for actions will be substituted into the predicates for product names.

Semantic header technique introduces the hierarchical structure on the portions of data for a domain. The hierarchical structure of the domain is expressed via the set of links between a predicate and its argument. Let us pose a problem of the document classification based on the semantic headers.
To divide the totality of all documents of a domain into classes, we choose the class-determining predicates. Their arguments will be the other predicates and objects which determine subclasses. Those subclasses, which are assigned by predicates are split into sub-subclasses, and so forth. The terminal classification units (those, which are not further split into subunits) are assigned by the objects (not the predicates).

Achieving certain classification criteria determines the division of the set of all terms (keywords) into predicates and objects, as well as the hierarchy within the set of all predicates. The universal criteria of a classification of an arbitrary nature is the similarity of its classification graph to a tree. In other words, different classes should have as few subclasses as possible, different subclasses should have the minimum number of sub-subclasses, etc.

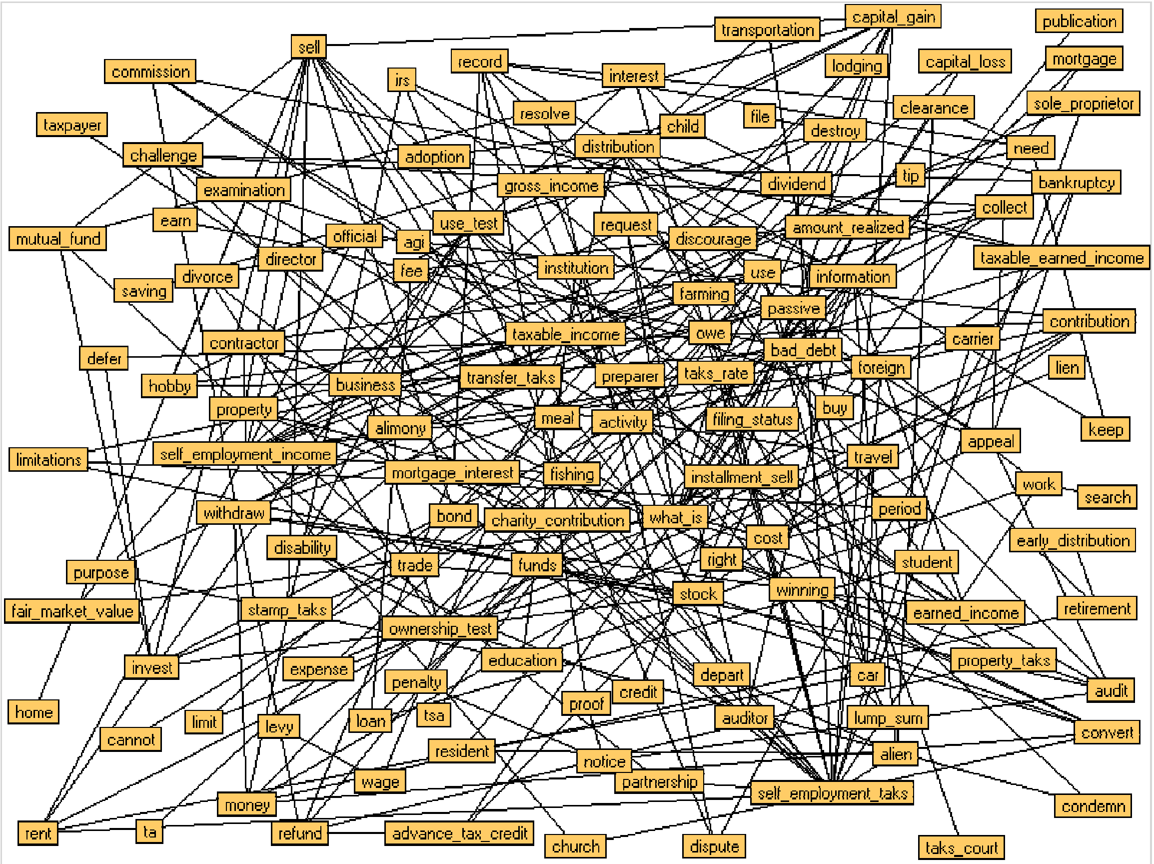

Fig 4.4: Visualization of the domain (Tax). The entities are randomly distributed to show the complexity of interconnections in accordance to existing semantic headers. The edge **audit-appeal** visualizes the semantic header
…(audit (appeal(…,…),…),…. Revealing the structure of a domain is important when we pose Q/A problem as a classification of a set of documents. Some predicates are assigned the by documents directly, arguments of the higher-level predicates range over the other predicates (shown above) to refer to a document.

In the semantic header approach, the classification criteria match the question answering quality: the better the structure of classification, the "cleaner" is the answer (in the sense of a fewer documents from a foreign sub-unit for an answer). As we mentioned earlier, the SH technique is a reduction of NL understanding problem to the pattern recognition one; however, there is no geometrical relationships between classes defined. To prepare a domain for Q/A, based on the answer recognition, the set of answer documents is subject to hierarchical classification. A classification of an arbitrary nature is based on the chosen set of features of objects. In our case, we choose the most essential entities of the documents to divide them into classes.

For example, we divide the Internet retailer domain into documents about products, about order processing, about shipping, and about navigating the site. It occurs based on the predicates *product(_), order(_), ship(_), site(_).* Evidently, other divisions are possible; they are based on an alternative choice of the essential entities. Each class can be divided into subclass, which is determined by a predicate or an object. For example, the documents about shipping are divided into documents about returns, about the geography of shipping, about length of shipping, about the duration of shipping and about the price of shipping. The semantic headers for these documents are, in particular, *ship(return(_)), ship(overseas), ship(duration), ship(pay(_)).* Subclasses are subject to the further division into sub-subclasses, if they are determined by a predicate (not by an object). The subclass of documents about shipping is further divided into sub-subclasses.

5. SH Space

5.1 Question-answering as set-theoretic mapping

Let we have a set of *keywords* K. Each document D_i from the set is assigned by a family K_i of ordered sequences of keywords. We highlight the set of **predicates** P based on the following principles:
Other elements of $K_p \subseteq K$ can be related to the predicate $p \in P$. The set of all documents D_p, assigned by the sequences with p, can be split into the collection of non-overlapping subsets of D_p, each of which is assigned by the elements of K_p. Any sequence of keywords for a document contains at least one predicate. Elements (atoms) of K, which are not predicates, are **constants**.

In logical terms, elements of K_p are substituted instead of the arguments of this predicate p. We call the **semantic header** the well-written formula, which contains the set of atoms from the element of the family K_i . Therefore, the corresponding family of these formulas is assigned to a document D_i.

The set of all arguments A of predicates from P is split into the subsets, which are characterized by **semantic type** S. Each argument of each predicate is assigned by a semantic type $s \in S$. Therefore, each n-ary predicate p is assigned by the n-tuple $s_1^p,\dots, s_n^p$ of semantic types of its arguments. Only those elements of K can be substituted into the arguments of predicate p, which belong to semantic type of its arguments. Each atom belongs to zero, one or more semantic types, so the assignment of semantic types is a many-valued function.

Introduction of semantic types divides the set of all keywords (atoms) into overlapping subsets of keywords of the same semantic types. Therefore, for each predicate p, the following holds for each argument of n: $K_p \subseteq Ks_n^p$. The set of all atoms, which can be substituted into the argument n of predicate p is a subset of all atoms of corresponding semantic type of this arguments. This statement holds because there are, generally speaking, other predicates with arguments of the same semantic type $s=s_n^p$.

The following **limitation for semantic types** affects the definition of a well-written semantic header.
A predicate cannot be substituted into its arguments.
If a predicate p can be substituted into predicate q, then vice versa is not allowed: predicate q cannot be substituted into predicate p.
We draw the **domain classification graph** G based on the hierarchy of predicate substitution. The *classes* of our classification are determined by the predicates, which do not belong to any semantic types (cannot be substituted into other predicates). Each class contains the subset of all the documents D. Division of these classes into subclasses occurs based on the predicates or constants, which are substituted in the class-determining predicates. Subclasses, determined by the predicates, are in turn divided into sub-subclasses, and so on.

Our graph G is directed and includes converging edges, generally speaking. The semantic header $p(\dots,q(\dots),\dots)$ introduces the directed edge pq between the nodes G_p and G_q . Each terminal node G_t is assigned a subset of documents D_t . Note that the edges pq and rq converge if the predicate q can be substituted into both p and r (both headers $p(\dots,q(\dots),\dots)$ and $r(\dots,q(\dots),\dots)$ exist in our domain).

The following problems can be posed in terms of presented above domain definition.

- How to define the difference (distance) between the documents in terms of classification graph such that this definition correlates with the difference in the **meaning** of these documents, expressed via semantic headers.
- How to perform the search of **optimal** classification graph (to minimize the number of converging nodes), varying division into semantic types and obeying separation between the documents.

In the framework of introduced above terms, the strategy of building the Q/A domains includes the following steps:

1. Choosing the set K;
2. Formation of the family of SH proimages for each document (answer);
3. Division of K into predicates and constants;

4. Choice of semantic types of predicates;
5. Building SH from their proimages;
6. Construction of classification graph.

5.2 Geometry of meaning for Q/A domain

Usually, it is possible to reveal a collection of sub-domains within a domain, more homogeneous with respect to Q/A than the whole domain. In accordance to our domain definition via patterns, we cannot formally divide the whole domain into (disjoint) components. However, each of these sub-domains is analyzed in accordance to specific rules. The higher uniformity within a sub-domain simplifies its encoding, adjustments of patterns, types, abbreviations, synonyms, default rules, etc. In this subsection, we intend to present these observations based on geometrical intuition.

The measure of importance above helps us to understand the domain structure and the role of each predicate in addition to such predicate properties as the degree of embedding into SHs and the level on the hierarchical classification. However, the latter parameters are less appropriate for numerical analysis.

We proceed towards the definitions for distance between questions, between TF and SH, between SH and SH, between answers and between predicates.

Lexicographic distance between SH is symmetric difference between the sets of atoms, which are the proimages of these semantic headers. For a better accuracy, the boolean representation of occurrence can be multiplied by a normalized importance parameter (see setion 5.3). Besides, we add an expression taking into account the fact whether these SH are assigned to the same answer.

$A = proimage(SH_1)\text{-}proimage(SH_2);$

$D(SH_1, SH_2)=\sum^{a\in A} occurrence(a)*\ importance(a) +\ hold(\exists answer : (SH_1 \rightarrow answer\ \&\ SH_2 \rightarrow answer)).$

To take into account variables, we add the property $d(a(b), a(c)) > d(\ a(b), a(\_))$. Note that the symmetric distance between SH and TF is the same as the distance between SHs except that the second term is eliminated.

Syntactic distance between SH is the weighed sum of the transformations, converting one SH into another. These transformations are the same as defined in section 3 to match TF with SH. Each transformation is assign a domain-dependent score.

Atomic distance between answers is defined as the symmetric difference between the atoms for SHs of the first answer and that of the second answer:

$AA = \bigcup proimage(SH_1 | \ SH_1 \rightarrow answer_1) - \bigcup proimage(SH_2 | \ SH_2 \rightarrow answer_2);$

$D(answer_1, answer_2)=\sum^{a\in AA} occurrence(a)*\ importance(a).$

Peripheral distance between answers. One can express the distance between the answers as a distance between a pair of points in a space, where these points are intuitively "centers of mass" of respective SHs. Intuitively, the measure of meaning deviation between two answers is based on the averaging of difference in the meanings of their respective SHs. Therefore the distance between answers is invariant with respect to the number of SHs for answers and formula complexity. Distance between answers is rather dependent on SHs, encoding different parts of answers, than on ones for various ways of saying about the same.

$D(answer_1, answer_2)= < D(SH_1, SH_2\ ) | \ SH_1 \rightarrow answer_1\ ,\ SH_2 \rightarrow answer_2) >;$

Peripheral distance between answers is based on either definition of the distance between SHs.

Assigned distance between SHs is based on the averaging over distances between assigned answers.

Distance between atoms is the distance between SHs computed via averaging over SHs pairs these predicates correspondingly occur in.

Classification graph is an important tool to estimate differences between atoms, SHs and answers as the graph distance between the nodes, paths and families of paths correspondingly.

Classification graph distance between atoms is the length of path (including edges of either directions) between the corresponding nodes.

Classification graph distance between SHs is the length of shortest path (including edges of either directions) between the corresponding paths.

Classification graph distance between answers is the minimum length of the path (including edges of either directions) between the corresponding paths from one family to that of another family of paths for these answers.

The distances between semantic objects observe the symmetry and reflexivity properties, but do not obey the triangle property. Multiple definitions of the distances assume application of iterations to dynamically obtain the values of distance expression parameters.

Defined above parameterized expressions for distance between such semantic objects as a predicates, SHs and answers assist in the following functionality:

1. Improved match TF≈SH. When the system cannot find an exact match, it searches for the best one based on the minimal distance.
2. Analysis of geometrical properties of a domain is helpful to determine the information coverage, completeness and uniformity properties. Based on the space of semantic headers one can reveal a least filled area. This would mean that more answers can easily fit (be reasonably separated) within this area. Knowledge of less dense areas in the SH feature space helps a knowledge engineer to mine for a new information, appropriate and subject to convenient encoding within a domain. Hence, construction of the feature space has similar role to that of under pattern recognition.
3. Distance between the predicates is helpful when one makes decisions whether a couple of words are synonyms or have to be expressed by distinguishing atoms.

Note that we have obtained the distances between semantic objects as parameterized expressions. Parameters are obtained in the course of machine learning and are domain-specific.

Operating with distances between atoms, SHs and answers builds the bridge with probabilistic document retrieval methods (Boros et al 1998, Ciravegna et al 1999). Evidently, the latter possesses higher efficiency sacrificing the resultant q/a quality for a narrow domain. Obtaining statistical information about a domain based on the SH technique is very valuable, because each component of textual data is subject to thorough manual analysis. Accumulating statistical data on the semantic features of q/a domain such as correlation between the entities and their links to the answers is expected to be quite helpful for automatic text analysis for the larger domains.

In the probabilistic document retrieval technique (Gey & Chen, 1998) relevance of a document *D* to a query *Q* is given by an expression

$relevance(D, Q)=-3.51+(1/\sqrt{N+1})\Phi+0.09N;$

$\Phi=37.4\ \Sigma\,(qtf_i/(ql + 35\ ))+0.33\Sigma log(dtf_i/(dl+80))-0.194\Sigma\ log(ctf_i/\ cf\ )$

where N is the number of terms common to both query and document,

qtf_i / dtf_i is the occurrence frequency within a query/document of the ith match term,

ctf_i is the occurrence frequency within a collection of the ith match term,

ql/dl/cd is the length of query/document/collection, i.e. the number of terms in a query/document / number of occurrences of all terms in a test collection.

The essence of this approach is to apply Bayesian inference to develop predictive equations for probability relevance where training data is available from past queries and document collections. Logistic regression has become a standard technique for discovering the degree to which causal factors result in reaction incidence (Hosmer and Lemeshow, 89). In the document retrieval there is an inverse problem, how to predict the relevance of a document to a query given the evidence of occurrence of query words and their statistical parameters in documents (answers).

Application of logical analysis of data is based on the transformation of any judged document into a boolean vector with one component for each distinct classification term (Boros et al1998). Text classification by bootstrapping with keywords is a basis of the approach, which does not require a priory judged set of documents (McCallum&Nigam 1999). Instead, it uses a small set of keywords per class, a class hierarchy and a large number of easily accessible unlabeled documents. The keywords are used to assign approximate labels to the unlabeled documents via matching the terms. The preliminary labels are then subject to bootstrapping process that learns a naïve Bayes classifier using maximization of expectation and hierarchical shrinkage.

5.3 The measure of importance for domain entities

distribution	780	sep	222	loan	150	wage	76
tax	726	deduct	222	invest	150	rollover_ira	76
contribution	670	widow	150	inheritance	150	rmd	76
age	670	tax_deferred	150	flexibility	150	railroad	76
convert	490	simplified_metho	150	earned_inc	150	periodic	76
requirement	426	self_directed	150	discourage	150	penalize	76
income_taks	426	roth	150	death	150	other	76
educational ira	426	qualifying	150	collateral	150	option	76
early_distribution	360	property	150	child	150	nonperiodi	76

Table 5.3: Entities and their importance

When coding a domain, it is necessary to distinguish the role of each predicate in the domain representation. When we extend the match procedure between TF and SH, elimination/addition of a predicate into a formula needs to take into account the relative importance of a predicate.

To estimate the importance, we compute the symmetric difference of occurrence of the predicates in SH family of the pairs of answers and then perform averaging over the set of all pairs. Intuitively, if a predicate is associated with a small amount of answers or occurs in SHs for almost every answer, than the role of this predicate in answer classification (and, therefore, in the domain representation) is not very significant. Conversely, if an occurrence of a predicate in a SH family dramatically affects the assigned answers, this predicate is rather important for q/a domain.

Hence, for each predicate we calculate the number of pairs of answers, where SH family for one answer contains this predicate and that of the second answer does not. The results are shown in the Table 5.3, where each predicate is assigned an importance index.

As we see in Table. 5.3, the most important are entities *ira* and their generalizations/synonyms *retirement/pension.* It means that if we analyze a query for the occurrence of each of these words, we will filter out approximately the half of the set of answers as inappropriate. If we use the terms from the lower left part of the table (minimum importance), the absence of these predicates would mean that we have crossed out just an insignificant portion of the set of answers. Our definition of importance is based on the computational efficiency of the division of the answer set during the solving of the classification problem.

5.4 SH as approximation means

One of the important features of SH approach is that it approximates the meaning of answer to make efficient and plausible its match with the question via TF. We use the term *approximation* to show the reduction from the fully formalized answer to its essential content, represented by means of SH. We mean the *fully formalized answer* as the potential capability of representing each sentence and interlinks between the terms for the whole answer in our formal language.

More precisely, **approximation** is the **transformation of a set of formulas** (SHs) for an answer, **ignoring a given predicate**. This transformation follows the intuitive notion of declaring this predicate unknown. The approximation transformation therefore eliminates each occurrence of this predicate in the set of SHs for an answer.

Syntactically, if the predicate to be eliminated is the innermost in a formula, it is replaced by a variable; more than a single occurrence is replaced by the same variable.

For the expression $p(q(t(\dots))$, we obtain $p(Q, t(\dots))$, so the arity of p is decreased by one.

The new variable is the same for each occurrence of q in a formula.

We see that approximation transformation modifies the patterns of the predicates, which range over the ones being eliminated. To avoid growing of the predicates' arities, we merge the newly added arguments and correspondingly merge the semantic types. For the comments on advantages and disadvantages of these operations on semantic types see section

To sum up, we write the definition and properties of our approximation:

Def0: $approx_q(q(\ldots))$ gives an empty formula (the formula is eliminated from the set);

Def1: $approx_q(p(q(\ldots)))= p(Q);$

Def2: $approx_q(p(q(t(\ldots))))= p(Q,t(\ldots));$

Generally speaking, approximation is not commutative: $approx_q(approx_p(\varphi)) \neq approx_p(approx_q(\varphi))$.

Ideally, one would want to have the following feature for the approximation: if a TF matches with SH, then this TF will match with any proimage of SH under approximation:

Approximation monotonicity: $(TF \approx SH) \Rightarrow (\forall p (TF \approx SH_1, approx_{\ p} (SH_1, SH)).$

Since this does not always hold, we do need to apply approximation to the fully formalized answer representation: when we build it, the focus is rather on more complete representation of the content than on the representation, appropriate for q/a. Therefore the formal approximations is intended to convert the former into the latter. From q/a perspective, approximation of the set of SH is actually an improvement of it.

As a property, complimentary to approximation monotonicity, we consider the **separation monotonicity** property:

$(((TF \approx SH) \ \& \ \neg(TF_1 \approx SH)) \Rightarrow$

$\quad (\forall p (TF \approx SH_1 \ \& \ approx_p(SH_1, SH)\& \ \neg(TF_1 \approx SH_1)\& \ approx_p(SH_1, SH))).$

This property states that if two semantic headers are separated by any translation formula, then all the approximation proimages are separated by this translation formula as well.

Furthermore, extended match TF poses additional constraints on the distance conservation under approximation:

Distance monotonicity property:

$((d(TF , SH) < d(TF_1, SH)) \Rightarrow$

$\quad (\forall p (approx_p(SH_1, SH)\& (d(TF , SH_1) < d(TF_1, SH_1)))$

5.5 Approximation subgraph

We have introduced the approximation operation on the set of SH for a domain. Since the classification graph fully represents the domain structure (interconnection between its entities), it seems natural to define the domain approximation as a graph reduction operation.

The approximation operation is applied to a **node**: it is identified with all nodes it has a directed edge to. This operation is not allowed to distort the original paths except ones, consisting solely from eliminating edges. Only answers assigned to the latter edges can become inaccessible as a result of approximation.

$gapprox^n\!: \ (n, G) \rightarrow (n', G^n), \ n \rightarrow n',$

$\forall n_1 : edge(n_1, \ n) \ n_1 \rightarrow n',$

$\forall path_1 \forall path_2 \ (\ gapprox^n(path_1)=gapprox^n \ (path_2 \ ) \Rightarrow \exists \ n_1 \exists \ n_2 \ path_1 = edge(n_1, n)\& \ path_2 = edge(n_2, n))$

Graph monotonicity property:

$$\begin{array}{ccc} \{SHs\} & \rightarrow & G \\ \downarrow & & \downarrow \\ \{SHs^{approx}\} & \rightarrow & G\text{-}subgraph \end{array}$$

The diagram above is commutative: for any domain classification graph and its approximation subgraph there is a corresponding pair of the SH sets, one of which is an approximation of the other. The same approximation subgraph can be obtained by both ways:

first building the classification graph and then its approximation subgraph;
first deriving approximation SH set and then building its classification graph.

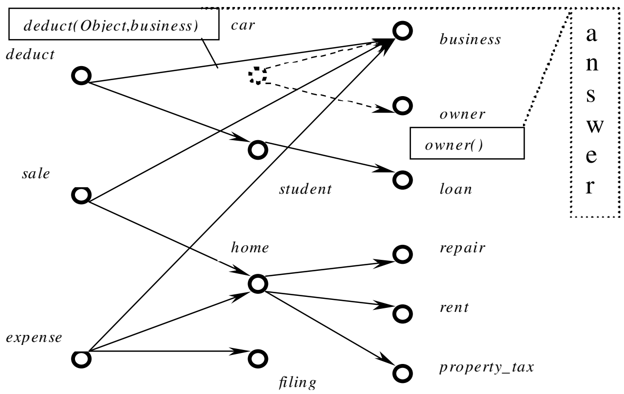

Fig.5.5: The domain classification graph is obtained as a reduction by means of eliminating the predicate *car()*.The node *car* of the original graph (Fig.4.1) is merged with either node it has a directed edge to (here, for example, the node for the constant *business.* The ends of all edges, directed to the node to be eliminated, are transferred to obey the original paths. For example, the path *sale-car-owner* is reduced to the single-edge path *sale(W, business).* The new argument appears to represent the unknown predicate we just have eliminated (see approximation rules in the section 5.5).

If retraction of a node eliminates the single-edge path assigned with an answer, this answer (possibly, ambiguously) becomes assigned to another path ending in the merged node. Note that the node elimination operation can lead to non-deterministic merge of answers.

5.6 Modifying SH domain to achieve optimality

In this section, we consider the issues of domain optimality and achieving it by means of modification of semantic header. Informally, the less non-adequate information an average query delivers, the better the q/a system. Basically, the quality of domain representation can be measured by the minimal number of semantic headers, delivering the related answers. The following should be stated to clarify the **domain quality criteria**:

We fix the overall domain knowledge and vary access to it. For SH domains, we keep the totality of textual documents, but can vary their division into answers to provide the most accurate ones.

Furthermore, we fix the formalization degree: all the facts of the under-threshold importance are ignored; only the combinations of important facts to form SHs are varied. In addition, we fix the TF-SH matching machinery.

The definition of optimality criteria is based on the totality of formalized (not NL) queries, which is defined as the set of well-written formulas in the fixed query language (derived from the original set of keywords *K*, see section 5.1). Note that we distinguish instantiated and non-instantiated formulas since they can form distinguishing SHs.

Generally speaking, increasing the answer size would improve the probability of providing the correct answer but decrease the answer precision: the lower portion of an answer becomes relevant. We call the corresponding parameters answer **accessibility** *a* and **precision** *p.* In addition, separation criterion *s* characterizes the difference between the answers, expressed via SHs: how many queries deliver both answers

The accessibility of an answer is defined as a number of queries delivering this answer normalized to the total number of queries. The domain accessibility is then derived by averaging through the accessibilities of all answers: $a=<\#q_a/\#q>$. This criterion tends to increase the number of SHs per answer and is in conflict with the separation criteria. The latter one is defined for a domain as a portion of “ambiguous” queries, which are matched with more than one SH from different answers. Note that frequently the drop of separation is a desired property for general questions. Numerically, $s = 1- \#ambiguous\ q$.

The precision of answer for a query is based on the number q_a of other queries that delivers this answer. Generally speaking, precision p is determined by the number of SHs per answer (the more issues an answer contains, the more SH it will be assigned and therefore the less precise it will be). For a domain, precision definition is based on the averaging through all queries: $p = 1 / (1 + \langle q_a \rangle)$. The reader can verify the critical values when we have a single answer per individual query $(p=1, a \to 0, s=1)$ or a single answer $(p \to 0, a \to 1, s \to 0)$. Now we can define the domain optimality as $pas \to max$. To compare two domains when the distinguishing portions of q can be matched, one needs to introduce additional criteria of domain completeness $c= \#q_{match}/\#q$. Then $pasc \to max$ will serve as optimality criteria, if one is allowed to add new SHs.

Therefore, one uses *pas*-optimality to estimate the modification of a domain, taken separately, and *pasc*-optimality to compare the quality of different domains.

The following domain modifications affect the optimality criteria *pas*:

1. Dividing an answer a_{div} into two answers a_1 and a_2, based on the division of the SH family into two sub-families. The domain precision will increase because the average number of queries per answer decreases. Averaged a decreases by $((a_{div}/\#answer)- (a_{div}/(\#answer+1))$. s can decrease if there are queries which were addressing SH corresponding to a_1 and a_2
2. Merging of answers has the inverse effect on the values of optimality criteria.
3. Merging or splitting SHs for an answer does not affect the optimality.
4. Adding SH to an answer will directly affect c criteria, decrease p and s criteria and increase a.
5. Adding more specific new answer to an existing (general) answer, splitting existing SHs, affects optimality similarly to 1), except that p increases stronger.
6. Replacement of an existing (more general) answer by a new specific answer, merging and/or eliminating existing SHs increases p stronger than 5), increases s and decreases a stronger than 1).

Operations, inverse to 5) and 6) affects the optimality correspondingly.

Hence, each of the operations 1), 2), 4)-7) could increase or decrease *pas*, so one needs to explicitly estimate the parameters to choose domain modification properly. Optimality criterion can be in conflict with some additional domain-specific requirements (for example, how the answers should be formed), so the resultant optimality can have more complex expression or cannot be estimated numerically.

We proceed to the operations, accompanying the achieving the domain optimality.

Operations while **adding new answers**:

1. Introduction of a predicate
2. Introduction of a new constant
3. Adding a constant to a semantic type

Operations while **domain reevaluating**. If a term is not required to distinguish between the current set of answers, this term is eliminated. There are two ways of elimination with respect to information loss:

1. eliminating a predicate by approximation
2. eliminating a predicate by expressing it via the other predicates in all SH it occurs

Solving of an **equations with respect to a predicate**. As we stated in the section 2.6, the *meaning* of a predicate is its capability to separate answers based on the questions, having the families of SH for each answer fixed. For knowledge representation in a Q/A domain, **expression** of a predicate means the following: what words should replace the given one to conserve the capability of accessing the same number of answers. We need to obtain the expression formula for a predicates to reconstruct the set of words, which need to occur in a query instead of the word for a predicate being hypothetically eliminated. Required machinery is a subject of our future study.

6 Increasing the system robustness

6.1 Interactive clarification

When the question is too general, or the query entity involves the unknown parameter, the system comes back to the user and asks to find the object from the drop-down combo box, which is the most relevant to the mentioned one (see fig 2.4). For example, when one asks "how to deduct my Toyota" the system builds the uninstantiated translation *deduct(_,_)* because it does not know the models of cars. However, the system will suggest the combo box, which includes the deduction items and cars in particular. The user is then supposed to choose a parameter, known for the system and resubmit the question. By default, if a question is translated as an uninstantiated predicate of a common-usage term (e.g. *deduct, cost, expense, home,* etc.), it is assumed as a "what is" kind of questions.

6.2 Multiagent domain representation

To decrease the conflicts between the ambiguous entities when a domain grows up in its size, knowledge is distributed between Q/A agents. These agents share the same domain environment (set of predicates and values for their semantic types), but the domain information is addressed from the various prospective. For the tax domain, the first agent knows the general data about taxes based on the "TeleTax" information, the second agent knows how to file forms, and the third agent obtains the longer answers for the specific topics. The distributed architecture allows us to produce the better coverage of the domain, though the same information can be stored from two different viewpoints. Therefore, we improve the domain coverage, sacrificing the increase of the total amount of data. The customer gets the first answer from the agent, which has obtained more specific (usually, a single) answer. The other agents are initiated upon the request. Distributed architecture helps the customer to obtain more complete answer without loss of its accuracy.

When the customers feel the system would gain by a new portion of tax knowledge, they used the Domain Extension Wizard for tax domain. DEW is called as well when the translation formula cannot be satisfied by semantic headers completely. The partial satisfaction means that just one conjunctive member of the translation formula finds its semantic header; it usually happens when a query involves two distinguishing sub-domains.

6.3 Domain Extension Wizard (in the Tax domain)

Domain extension wizard (DEW) allows a person, knowledgeable in a particular Tax area, to share his/her experience in this domain. A customer can extend the natural language question-answering domain with the new advice, recommendation or intriguing story and to specify the major ways of asking about it. This toolkit processes the **combination** of the answer (the textual representation of advice with possible URL links to more detailed references) and a set of questions or statements. These questions or statements are supposed to be natural from the prospective of the domain extension author to generate the document as a reasonable answer (association). DEW does not directly support the addition of new facts or entities into the domain representation. However, the input of a set of questions, accompanied by an empty answer will mean that the pure formal facts are added (answers contain the parameters of the predicates). In the same manner, the input of an answer with an empty question window will switch DEW to the option of automatic annotation: formal questions will be generated independently. The (base) domain is ready for extension by DEW when its semantic model is designed: the essential entities and their attributes are specified, and the domain-related synonyms and multiword substitutions are defined.

Enter question(s): In this window, you can input all the questions or statements you think to be the natural ways of querying about the work of literature in the right window. A mark '?' should follow each question. You must keep in mind that not only the questions you specify explicitly can be asked. For each question you specify, there is a family of similar questions, which will be answered using the same formal representation as the explicitly specified questions.	**Enter answer**: Here you can input the text you want to be available to your site customer. You, as a question-answering system developer, have control over how large the portions of text should be. The larger the answer, the higher the number of questions per answer is expected. Correspondingly, the answer generality increases (the customers will get a larger portion of unnecessary information in the responses).
How do I increase itemized deduction? Which expenses can I include in itemized deduction?	To increase your itemized deduction, you need to include …to your expenses…
Add to domain Press this button when you want to add a new portion of text (answer) with a list of questions. You will see the confirmation of the added questions and answers below. When you have extended your domain, it is necessary to build the question-answering application, which implements this extension. Please press the 'Compile domain' button after you have finished inputting the domain extension data.	Compile domain When you press this button, the newly created source file will be appended to the existing domain and the updated source will be compiled to update the domain. You have to add all questions-answer combinations to the domain before you compile it. The demo implementation is based on a single domain; it assumes a single user to extend it at this moment. After the extended domain is compiled, the current domain is turned into the base (original) one.

When the extended domain is compiled, you can see the formal (internal) representation for the questions. For some questions, DEW fails to build any representations. (This means that the developer has used unfamiliar words). Occasionally, DEW will build the same representation for two distinct questions. When DEW is unable to link an entity and its attribute, the semantic header (question translation) may be too general. Finally, a wrong translation can make the domain extension data inaccessible or cause the inclusion of non-related data in iAskWeb's answers. In all these cases, manual editing of the domain representation files (by a knowledge engineer) is required.

Domain extension code:

deduction(itemized, increase(_)):-do201. expenses(include,_):-do201.
do201:-iassert($To increase your itemize deduction, you need to include …to your expenses…$).

Domain is compiled. Ask a question to the updated domain

Ask Now you can ask the questions for the domain extension as well as for the base domain

6.4 Handling situations with the lack of direct answer

In the section 3, we have presented the mechanism of the attenuation of a translation formula when the is no direct answer (no SH can match it). Attenuation of translation formula serves as a particular mechanism of the search for an appropriate answer in case the direct answer is absent in the system (fig. 6.4).

If no entity is found in a query, exact answer is not expected, but a close one can be obtained if the system finds a predicate, which is satisfied, instantiated by a constant from the input query. In particular, *what_is* predicates is verified to be satisfied with the query terms.

If there are too many predicates, which can be potentially satisfied with the attributes from a query, it is very unlikely that a proper one would be chosen. Therefore, it is preferable for the system to give no answer or try a keywords search, especially in the multiagent environment, where another agent may have an answer with higher confidence level.

If no potential constants have been extracted from a query, the internal keyword search is launched. Only the keywords, found in just a few answers are considered adequate, and these answers form an output one. External keyword search is initiated if the Q/A agents have failed to produce an answer

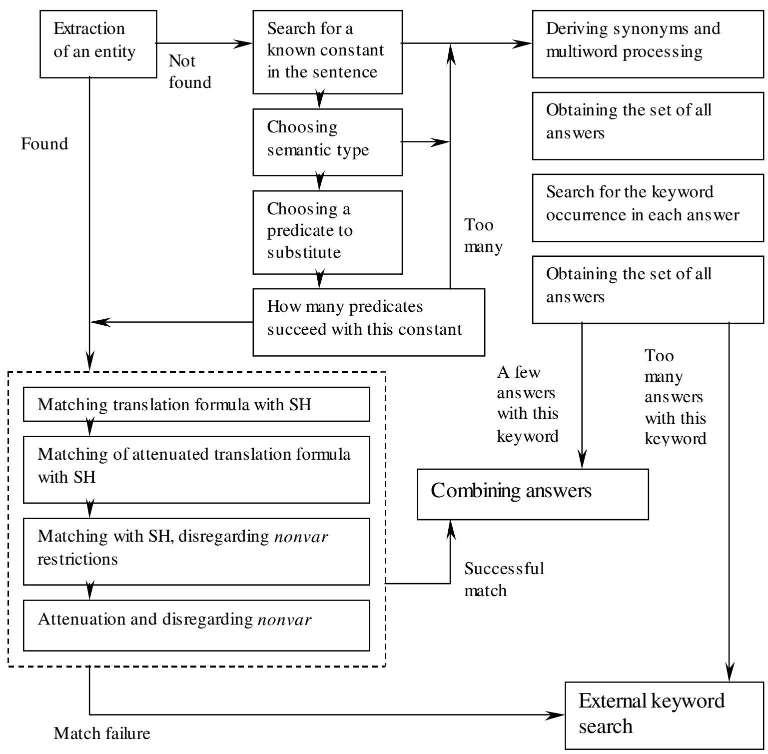

Fig 6.4: How the system handles the situations when there is no direct answer. Attenuation of translation formula (left-bottom block) has been discussed in the section 3. Building translation formula, the system tries to find the absent predicate of no one has been extracted from the input query (top right). If the found predicates generate too many answers, internal keyword search is initiated and the number of relevant answers is analyzed.

Combining answers block outputs the answers produced either based on occurring or hypothetically attached predicates, or based on the internal keyword search. If all of these attempts fail, external keyword search is launched.

6.5 A sample run

We present the protocol of answering the relatively complex query.

[should,i,appeal,if,the,audit,requires,to,pay,more,tax,than,i,owe]
NLP System for Semantic Header Domains.
Copyright(C) B.Galitsky / iAskWeb,Inc. 1988-2000 All rights reserved.
replace_abbrev([should,i,appeal,if,the,audit,requires,to,pay,more,taks,than,i,owe])
synonyms/multiwords
pattern(appeal(xatransaction,xscore)) pattern for *appeal*
superMember(more,appeal,0)
related_to_check(appeal,more) *more* is substituted into *appeal*
pattern(audit(xcontrol,xscore))
pattern(pay(xmoneyaction,xcategory,xscore))
related_to_check(pay,taks) *taks* is substituted into *pay*

pattern(taks(xataks,xcategory,xscore))
related_to_check(taks,owe) *owe* is substituted into *taks*
pattern(owe(xdebt,xscore))
related_to_check(owe,more) *more* is substituted into *owe*
[appeal(more,_33908),audit(_33928,_3393C),pay(taks,_33958,_3396C),taks(owe,_33990,_339AC), owe(more,_339D0)] flat translation formula
metapred_subst([appeal(more,z33908),audit(z33928,z3393C),pay(taks(owe(more,z339D0),z33990,z339AC),z33958,z3396C)]) substitution of metapredicates
elimProd([appeal(more,z33908),audit(z33928,z3393C),pay(taks(owe(more,z339D0),z33990,z339AC), z33958,z3396C)],[appeal(more,z33908),pay(taks(owe(more,z339D0),z33990,z339AC),z33958,z3396C)])
The predicate *audit* is eliminated because it is uninstantiated. It is indeed semantically redundant.
addition of external predicate :
modyTranslTerm([appeal(more,z33908)],[right(appeal(more,z36324),z36344)])
attenuate each term:
modyTranslFormula([appeal(more,z33908),pay(taks(owe(more,z339D0),z33990,z39AC),z33958,z3396C], [right(appeal(more,z36324),z36344),pay(taks(owe(more,z339D0),z33990,z339AC),z33958,z3396C)])
neither straight nor reordered call succeeded

In the protocol above, finding the *patterns* of predicates is followed by the substitution verification, implemented by the service predicate *related_to_check*. Constant substitution is followed by the metapredicate substitution, and the results are subject to elimination of the (most) uninstantiated predicate, if the TF is a conjunction. The resultant TF cannot be satisfied by a SH, so the attenuation is required. The term *appeal(more,z36324)* requires addition of an external predicate*; more* is substituted into *appeal* because it belongs to semantic type *xatransaction.* A knowledge engineer did not predict the combination *right-appeal,* so the system needs to compensate the lack of necessary SH itself. The predicate *right* comes into play to form the satisfied term *right(appeal(more,z36324),z36344)]).* Since the added predicate is semantically relatively neutral, we can expect insignificant meaning deviation. Below we show the answers the resultant TF delivers. We see that the first conjunctive member (which has actually been attenuated) delivered more relevant answer than the second conjunctive member, *pay(taks(owe(more(…)…)…)…).* (Note that the answer components appear in the inverse order.).

Indeed, the first answer is about owing tax, and the second one is about the IRS appeal procedures.

For various reasons, you may not be able to pay your federal individual income tax in full. Do **not** delay filing your tax return because you are unable to pay in full. If your tax return is not filed on time, you may have to pay a late filing penalty, in addition to a late payment penalty, and interest…
It is important not to ignore an IRS notice. If you neglect or refuse to make payment or other arrangements to satisfy your obligation in full, we may take enforced collection action (filing a notice of federal tax lien, serving a notice of levy or offset of a tax refund).
When doubt exists as to whether you owe the liability or whether you have the ability to make full payment on the amount owed, the IRS may be able to settle your unpaid tax balance for less than the amount owed. See Topic 204, *Offers in Compromise*, for more information.
If you would like more information on "your rights as a taxpayer," how to make arrangements to pay your bill, enter into an installment agreement, and to inquire about the implications when you take no action to pay, download Publication 594, *Understanding the Collection Process*, and Publication 1, *Your Rights As a Taxpayer*, or you may request copies by calling 1-800-829-3676.
If you disagree with the results of an audit or examination of your tax return, or if you disagree with certain actions taken by

the IRS to collect tax, you may have the right to an appeal. The Internal Revenue Service provides for an informal appeals process in many cases.
The IRS has an appeals system for people who do not agree with the results of an examination of their tax returns or with other adjustments to their tax liability.
For further information on the appeals process and information on how to stop interest from accruing on any anticipated liability, download Publication 5, *Appeal Rights and Preparation of Protest for Unagreed Cases*, and Publication 556, download *Examination of Returns, Appeal Rights and Claims for Refund*. Publication 1660, *Collection Appeal Rights (for Liens, Levies, and Seizures)* discusses how you can appeal those actions and may be ordered by calling our toll free number at 1-800-829-3676.
Publication 1660
This publication explains …

7. Application of default reasoning for the processing of the translation formula

7.1 Default reasoning for NLP

NLP is one of the unique possibilities to apply the nonmonotonic reasoning. Rather advanced theoretical investigations have been carried out towards building the formal background for nonmonotonic intuition and a series of counter-examples. However, the number of practical applications of nonmonotonic reasoning is far behind the theoretical advances. Representation of rules, which perform matching of translation formulas against the semantic headers, indeed requires implementation of default reasoning.
This approach to nonmonotonic reasoning is one of the most promising one in terms of practical applications from the viewpoint of the series of authors (e.g. Antoniou 1997, Levesque & Pirri, 1999), and directly connected with the NL task of semantic-level disambiguation.

Reasoning with exceptions is the inherent feature of natural language; because the domain-independent semantic analysis skips the exceptions, the matching unit becomes the essential one to handle the domain-specific deviations of the concepts.

An abstract default logic distinguishes between two kinds of knowledge, usual predicate logic formulas (axioms, facts) and "rules of thumb" (defaults). Under SH, the former component is variable (the current query translation) and the latter one is fixed for a given domain. Default theory includes the set of facts which represent certain, but usually incomplete information about the world (query, in our case); and a set of defaults that cause plausible but not necessarily true (because of ambiguity) conclusions. Some of these conclusions have to be revised when more query context information is available.

Let us consider the traditional example from the nonmonotonic reasoning literature.

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

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 matching rule default, we read it as follows. "If the query is about the bird, and it is consistent to assume that there is a word in this query with the meaning of *flying,* then conclude that the query is about flying". In the absence of knowledge to the contrary, which could be brought in by the other words of the query, it is natural to assume that this query is about 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 later we learn that he cannot fly, for example because he is a penguin, then the default becomes inapplicable.

Nonmonotonic reasoning technique helps us to provide the proper question-answering in this situation. Imagine that we have a set of documents answering questions about birds. One document is a general one, it presents information that *birds fly*, and contains the common remarks. There is one specific document, stating that *there is a bird Tweety, which is a penguin, lives at the South Pole, swims, but does not fly*. How should semantic headers, involving the predicates *bird(Name)* and *fly(Name)*, look like?

If there is no processing of the translation formula, then the explicit restriction for the semantic header 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?”; they will cause the same answer that birds fly, unless we mention the *bird Tweety*.

bird(X), X\=Tweety -> respond(birds fly).
fly(X), X\=Tweety -> respond(birds fly).
bird(Tweety) -> respond(there is a bird Tweety, which is a penguin, lives at the South Pole, swims, but does not fly).
fly(Tweety) -> respond(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 SH. Each exceptional object has to present in the semantic headers, which include 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.

7.2 Examples of defaults from the Tax domain

We provide more examples of default rules for the *Tax* domain. Let us highlight a sub-domain, based on the predicates *tax, deduction, income, expense, decrease*, their synonyms, attributes and parameters. Each of these words has multiple meanings; the manifold of meanings for each word is expressed by the joint occurrence with the other words, involving predicate-argument syntactic relationship.
There is a few steps in the NLP for SH, where nonmonotonic reasoning seems to be applicable. We highlight two formal theories, which can contain defaults:
theory of domain representation;
theory of transformation of translation formula.
Application of default rules is essential to use the additional semantic knowledge for proper transformation of translation formula. The essential role of defaults is to eliminate the unnecessary conjunctive members from the translation formula. These members do not bring in additional constraints for an answer, though the corresponding words can reasonably occur in the input query. At the same time, these terms of the translation formula dramatically increase its generality, potentially involving the manifold of unrelated answers.

Application of default rules is motivated by the very different factors than the generality control, though both these tools have the pragmatic nature. The former is means caused by the inertia of human (and human-machine) verbal communication, when a question author mentions the entities, assumed by default in a vertical domain. Default rules, in contrast to generality control machinery, are applied to compensate that inertia to the properly represented formula. Conversely, generality control corrects wrong translation formula, obtained as a result of improper semantic hypothesis.

For example, let us consider the input query “how to file time extension for my tax return?”. Its translation looks as *extension(file(),time), tax(return()).*

It is evident that the latter member can be eliminated without loss of constraints for an answer. The default rule to provide such kind of functionality will look like the following: “If the question is about *tax(…)* and it is not inconsistent to assume that the question is about *extension(…),* then we conclude the question is actually about *extension(…):*

```
tax():  extension()
----------------------
extension()
```

Note that we express the defaults via the predicates of translation formula, rather than via the atoms of an original query, which would look like

“tax”: “extension”
----
extension()
The motivation for the rule above is that the necessity of application of a default rule is caused by the specific semantic of the domain rather than by the incorrect translation into the formal language. Necessity to apply the reduction is caused by the fact that the queries “how to file… of my tax return”, “how to file… of my return”, “how to file… of my tax”, “how to file… of my tax documents”, “how to file… of my documents I have sent to IRS”, are equivalent in our vertical domain with the given set of answers.

To illustrate the importance of the premises of our default rule, let us consider the results of keyword search for “how to file extension of time for tax return”. The most relevant document is about the extension of computer file. Therefore, the default rule for the premise “computer” looks like
“computer”: “extension”, “file”
----
computer_file(extension())

It is clear, that it is not always correct to eliminate the most general predicates like *tax*. For example, in the question “tax on the sale of home” we cannot eliminate the predicate *tax*; nothing needs to be eliminated in this case.

Let us illustrate the application of default rules on the domain classification graph.

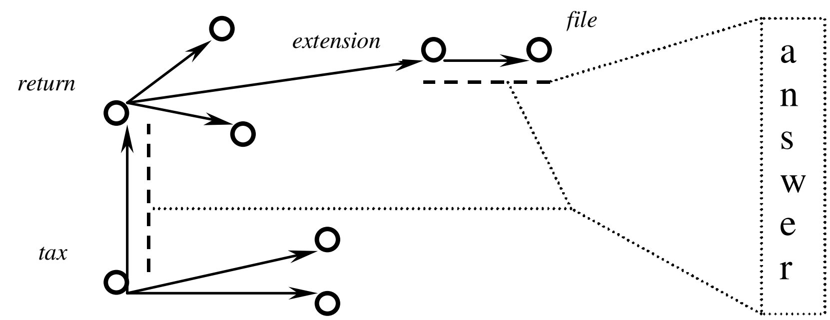

Fig. 7.2: The fragment of the classification graph: illustration for the query “how to file time extension for my tax return?”. Its translation as a conjunction as *extension(file(),time), tax(return())* is appropriate for the application of default rule.

What if we prefer to do without the defaults? The system would generate answers for each conjunctive members: everything about tax return and about filing extension of time. To avoid this situation, the following term should be derived as the hypothetical translation formula: *tax(return(), extension(file(),time))*. The problem with it is that, generally speaking, both *return()* and *extension(file(),time))* are supposed to be substituted into the predicate *tax*. Another solution would be to substitute *extension(file(),time))* into *return(),* which is shown on the graph (the lower path). The strong disadvantage of this approach is that a series of such substitutions (i.e. specific semantic headers) will be required for each way of verbal introduction of “extension of time for…”. It would lead to an explosion of the required number of semantic headers, that dramatically decreases both efficiency and robustness of Q/A system.

The typical situation of applicability of default rules is as follows. The translation formula consists from two conjunctive members, one of which is on the higher classification level (can potentially initiate a high number of answers), and another is on the lower classification level (more specific, can initiate 1-2 answers). Since the former conjunctive member does not introduce constraints for an answer, it should be eliminated: the system should descent through the graph to the lower level.

Application of the default rules is not always required for a conjunctive form. Each conjunctive member is mapped into a path on the graph; if there exists a joint path, which is a union of the corresponding paths of these conjunctive members, then a reasonable (narrow) set of answers will be initiated with the intact translation formula. The similar situation arises when two conjunctive members are linked via the answer number (see section 2.4).

Let us consider two more examples:

1) First order language rule for "Do I need to report gain on sale of home?"

gain(sell(home(_))) : need(report)

----

gain(report)

2) Default rule in the way of keywords for "Please tell me what needs to be written in line 6b when we file Form 1040".

need, write, form, file : line

----

line

Note the difference between the following default rules ([…] denotes the list of predicates, which are conserved by a default rule):

tax, deduct, […]:student, loan *tax, deduct, […]:student, loan*

------------------------------------ *and* -------------------------------------

deduct, student, loan *deduct, student, loan,[…]*

Finally, note that as usual, constructing the system of default rules takes into account the required separation for each entity. A predicate is eliminated from the translation formula by a default rule if its absence / presence affects the answer.

8. Considering special domains (Tax advisor)

8.1 Questions in Tax Domain

Educational expenses incurred during vacation from your job.
How to claim exemption for my children?
What is this year 's deduction for children?
How much is maximum earned income credit for persons with one qualifying child?
What requirements apply to all taxpayers that eligible for the EIC?
Can I claim the EIC if my filing status is head of household?
Can I claim EIC if I have a social security card that says not valid for employment?
I am nonresident alien , when can I claim EIC?
Are there additional requirements to claim EIC if I do not have a child?
Which employers do not withheld social security tax?
Can I claim the excess as a credit against my income tax?
Can I add any social security or rrta tax withheld from my spouse s income?
Am I eligible for credit if I pay someone to look after your dependent (spouse) in my home?

Table 8.1 Sample questions in Tax domain. Q/A in this domain requires processing of up to 4 entities. For example, the highlighted query is represented by the translation formula

filing_status(household(_,_),_,_)&tax_credit(earned_income(claim,_),_,_,_).

Note that just extraction of the keywords *filing, status, head, earned, income, credit, claim* is insufficient to recognize the related answer. One needs to determine that the query logically joins the entities of *filing_status* and *tax_credit* (classes of our classification). The query reveals the subclass *household* of the former class and subclass *earned_income* from the latter one, which is in turn linked to sub-subclass *claim.*

The predicates we used in our Tax domain examples throughout the paper perform the acquisition of the additional arguments (with new semantic types) and a lot of potential ambiguities is introduced. If there is a pair of predicates with the same semantic type, then a parameter can be substituted to either of these predicates, bringing two distinguishing meanings in the translation formula. The whole picture of the domain representation becomes more distorted in real life domain.

8.2 Evaluation

The Tax Advisor was subject to evaluation by about 6 thousand users per month during the season of filing returns for 1999. All the transactions are logged; customers have the options to provide the feedback to the system concerning the particular answer if they are not satisfied with it (too long, non-relevant, partially relevant, etc.). Though the answers usually do not exceed a page of text, the customers had more than 90% satisfaction with the system (Galitsky, 2000).

Usually, the customers try to rephrase the questions in case of system's misunderstanding or absence of a response. Second iteration (rephrasing the answer) was almost always sufficient to obtain the required information. At the beginning of evaluation period, the number of misunderstood question was significantly exceeded by the number of questions when the system did not have the related knowledge. This situation was dramatically reversed later, however the number of misunderstood questions was monotonically decreasing in spite of raise of the overall represented knowledge.

A series of tax return assisting and investment companies has expressed the interest in the tax Q/A system. It can replace human agents, automatically answering tax questions in up to 95% cases. Human agents intervene Q/A process in case of failure of automatic system. There is the other scenario of human-machine interaction: automatic Q/A system accelerates the work of human personnel of technical support.

9. Related works and discussion

We first note that the sole fact that the system does not answer a particular question does not bring in any information about its internals. To extract information about a system, one needs to achieve the variety of reactions to the same (or similar) query, posed in different ways. The most valuable queries are those "on the edge" of correct/incorrect/no answer situations. In other words, to reveal more properties about NLU system one needs to bring it to the "bound of domain" position, when the slight changes in phrasing and topic dramatically affects the answers.

Numerical analysis of the quality of keyword-based NLU showed that the original keyword accuracy of 45% for the computer science papers can be improved by bootstrapping to 66%, which approaches the estimated human accuracy of 72%. In our domains, which are very far from the formal scientific ones, pure keyword approach would deliver less than 20% relevant answers. We believe that its statistical improvement can hardly reach 25% value.

It is very important for Q/A systems to merge the meanings of multiple entities in a query. For example, AskJeeves responds to the question "How can I find a medical insurance in Boston" as enumeration of various services in Boston, which includes neither medical nor insurance services. However, if one asks "Where can I find a medical insurance" AskJeeves gives you general information about medical insurances. In the first example, the keyword *Boston* was extracted and matched, and in the second one, nothing left besides *medical insurance* concept to match. Such kind of functioning shows AskJeeves inability to handle more than one entity in a query unless there is a template question, which contains all entities. The letter would lead to the exponential growth of template in the case of syntactic match.

In spite of the series of efforts in the recent years, the development of computationally feasible semantic theories and, in particular, their linkage to the grammar level by an adequate semantic interface has not reach an expected level of consolidation. Just a low degree of consensus has emerged concerning the formulation of semantic interpretation rule to map the syntactic structures onto semantic representation. The lack of mature methodology has immediate theoretical as well as practical consequences for NLP system design. Rule sets tend to be developed and updated in a less than systematic manner to be adjusted to the complex linguistic data, manifold of description layers, heterogeneous format and finally become non-portable. The latter leads to the dramatic raise of number of rules for a multi-domain Q/A system. As a result, the compatibility of rules and their mutual interactions may get out of explicit control of the knowledge engineers.

Technique of SH is just a single approach to systematize the semantic rules. Two abstraction mechanisms were introduced by (Romacker et al 1999), to increase the linguistic generality, by means of using the configuration within dependency graphs rather than a link to a particular language phenomena. The "minimal" subgraph of a syntactically dependency graph, intended to cover the manifold of semantic observations by instantiation of few and general interpretation schemata. The second abstraction specifies the way how these schemata interact with grammar and domain knowledge. The efficiency of semantic description is further increased by using the inheritance hierarchies of these schemata.

The standard way of deriving a semantic interpretation is to assign each syntactic rule one or more semantic interpretation rules and to determine the meaning of the syntactic head from its constituents (see, for example, Eijck & Moore 1992). There are no constraints on how to build this rule set except by the semantic language itself. There is no specification for such abstraction mechanisms (descending beyond the level of sortal taxonomies for semantic labels (Creary and Pollard 1985) as commonsense reasoning rules and property inheritance, so the number of rules rapidly increases. Introduction of interpretation schemata instead of assigning specific interpretation rules to each grammar item raises the efficiency of domain preparation.

Semantic representation can be treated as a direct mapping from syntactic to conceptual representation (Sondheimer et al 1984, Hirst 1988) where the property inheritance is in active use. Single interpretation rule is attached to each role, therefore, all specifications of semantically relevant attachments of these roles must be present. Property inheritance is in use when the selection of alternative semantic rules is constraint to the one, inherited from the most specific case frame. (Charniak & Goldman 1988) and (Jacobs 1991) already specify semantic rules within the inheritance hierarchies. Since the semantic interpretation rules in these studies are based on syntactic rules and not on abstract structures in the dependency graph, covering multiple syntactic phenomenon, the number of the former rules increases.

Underlying the Q/A design principles, we present the necessity of strict separation of linguistic and conceptual knowledge. Each semantic approach suggests its own interface between them; both knowledge representations should obey the hierarchy principle in accordance to multiple studies. This interface can in turn be divided into two levels:

static, high-level constraints implementing the mapping of syntactic to conceptual roles or the meanings of closed classes of lexemes.

dynamic, search through a knowledge base, parameterized by a limited number of simple schemata.

The development of new tools for structuring of technical data is particularly important in the business area, where on-line textual information from news and service providers has been heavily used. Reports from the Gartner Group (Bair 1998) explicitly mentioned advanced classification systems with semantic technologies as the most strategically relevant element to support effective knowledge management. However, currently available technologies are still more or less based on IR ones, which does not provide adequate accuracy, further affected by Q/A itself, when coping with rich and complex classification structures. In contrast to IR systems, which do not take into account linguistic information, the pattern-matching (PM) can achieve accurate and fast categorization over a large number of classes (Hayes& Weinstein 1991, Jakobs & Rau, 1990). Resource development for PM does not require linguistic expertise and can be done by trained users, but is still weak in respect to linguistic analysis and good separation of document classes, required for SH Q/A.

Information extraction technique is useful for detecting texts relevant to a given single class with results 80-95% for both precision and recall, but displays the poorer performance on a large number of classes, that is the case for our Q/A domains. Under IR domain representation becomes expensive (Grishman 1997), as it requires a large amount of time or linguistically prepared personnel. Natural attempts to separate syntactic from domain-dependent knowledge simplifies the task, but does not solve the problem for a large amounts of answers.

Combination of the PM and IE provides the text classification adaptable to user needs (Ciravegna et al 1999). Pattern-based (shallow) classification performs the coarse text filtering (as a person skimming texts), and IE is oriented to handle some of the classes, where the PM classifications needs to be adjusted. The preprocessor needs to implement the database lookup to add semantic features (e.g. person, organization, location, etc.) as well as auxiliary categories, such as verb-of-speech for possibly multi-word tokens in the database. The database information is provisional and by no means complete. The deep analysis uses the abstract representation of lexical elements via Quasi Logical Forms (Ciravegna & Lavelli, 1999). The parsing is followed by the application of commonsense reasoning and discourse processing. Readjustment to a new domains takes one-three man month.

Note that in contrast to above approaches, SH needs the higher accuracy of text classification, as well as its hierarchical property.

10 Conclusions

It is well known that NL text modeling is a challenge to represent both syntax and semantic; and it is a necessity to represent a semantic, while representing syntax is optional, but may be important for matching in Q/A system in the situations with lack of world knowledge. The problem of correspondence of the approximate formal logical text models with the original natural language texts is addressed and answered by SH technique.

Conformal text representation and its intentional extension have been proven (Redey, 1999) to be a proper superset of classical first-order and intentional logic; meta-programming technique adds the pragmatic role of matching questions and answers. Q/A from the pattern-recognition viewpoint has been introduced (Fain & Rubanov 1996) via posing understanding problem as a choice of adequate reaction (answer) to a stimulus (question). SH technique assists in the transition from a simple answer classification graph to one, reflecting the domain representation language.

SH was developed as a means to combine the best of two worlds: symbolic reasoning and structural match without giving up much of either. SH needs reasoning because the general world knowledge cannot be represented via structural match, in other words, by means of "analogue" (physical) description. Search for the Q/A correspondence needs a structural match, because it is extremely inefficient to express it with the use of syntactic rules of individual unification for each question.

1. Tax domain has been created, which covers sufficient information for the general audience of taxpayers using about 800 and 250 answers for two main sub-domains. The domain includes more than 240 entities and about 1560 and 2280 of their parameters in these sub-domains (constants). 1800 semantic headers were designed to provide an access to these answers.

Building the Tax domain, we discovered that such a limited number of structural units covered uniquely logically complex and poorly formalized domain. It is well known, that the general vocabulary of business communication does not exceed 3-4 thousand words. Note, that it would require 10-100 times higher number of syntactic headers to provide the Q/A access to the same set of answers. We would have to explicitly enumerate combination of keywords for syntactic match instead of using the deductive capabilities of SH approach, which allows the automatic joining entities in a translation formula.

2. Natural language understanding technique to process up to four entities per query is required to perform the Q/A in the tax domain. Advanced semantic analysis with the generality control (Galitsky, 1999a) and rather expressive formal predicate language, extended with metapredicates for knowledge representation and query translation, are sufficient to represent the tax domain.

3. Technique of semantic headers brings in the convenient software implementation environment:
- Efficient and natural way of communication of knowledge engineers with domain experts;
- Robust and smooth capabilities of domain extension;
- Reusability of the domain components, including predicates and their semantic types, for the similarly structured domains.

4. Computational experiments show that the domain extension is possible in a variety of domains with different degree of potential formalization and logical complexity: *Geography and Politics, Military Crisis, Tax Advises, Internet retailer, Baseball, literature search*. Each of these domains has specific reasoning machinery and the distinguishing nature of preparation for Q/A.

5. Application of the semantic header technique to NL Q/A shows the following. There is the superior performance over the knowledge systems, based on just a syntactic matching of NL queries with the prior prepared NL representation of canonical queries, and the knowledge systems based on fully formalized knowledge. Our approach gives the higher precision of answers than the former one because it involves the semantic information in higher degree. Semantic headers 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 not required.

6. Logical approximation while matching the query translation against the semantic headers improves the robustness of search by making it easier to access a related set of document. Technique of semantic headers allows one to compensate certain semantic disagreements between the sample (training) queries and actual one, increasing the robustness of domain extension by a customer.

7. The Tax, Internet retailer, baseball, literature search and other systems have been recently released by iAskWeb Inc. A few portal companies are customizing these domains and preparing them for their clients. The customers use NL to search these domains as well as to extend them with their specific knowledge.

References

1. Tarau,P., De Boschere,K., Dahl,V., and Rochefort, S. (1999) LogiMOO: an Extensible Multi-User Virtual World with Natural Language Control. *Journal of Logic Programming*, 38(3):331-353.
2. Katz, B.(1990), Using English for Indexing and Retrieving, In Winston, P. H. and Shellard, S. A. editors, *Artificial Intelligence at MIT: Expanding Frontiers*, volume 1, MIT Press, Cambridge, MA.
3. Katz, B., Yuret, D., Lin,J. , Felshin, S., Schulman, R., Ilik, A., Ibrahim,A. and Osafo-Kwaako, P. (1999) Integrating Web Resources and Lexicons into a Natural Language Query System, *IEEE International Conference on Multimedia Computing and Systems.*
4. Tarau,P., De Boschere,K., Dahl,V., and Rochefort, S. (1999) LogiMOO: an Extensible Multi-User Virtual World with Natural Language Control. *Journal of Logic Programming*, 38(3):331-353.
5. Galitsky, B. (1999a) Natural Language Understanding with the Generality Feedback. *DIMACS Tech. Report 99-32*.
6. Galitsky, B.(1998) The formal scenario and metalanguage support means to reason with it. *DIMACS Tech. Report #98-27*, Rutgers University.
7. Galitsky, B. (1999b) Analysis of mental attributes for the conflict resolution in multiagent systems AAAI-99 workshop on Agents' Conflicts.
8. Galitsky, B. (1999c) Automatic generation of the multiagent story for the Internet advertisement *SSS-99 Workshop on the Intelligent Agents*, Stanford CA.
9. Galitsky, B. (2000) Technique of semantic headers for answering questions in Tax domain. *Proc IASTED Conf. on Law & Technology*, San Francisco CA.
10. Antoniou, G. (1997) Nonmonotonic reasoning. Chapter 14. MIT Press Cambridge, MA London England.
11. Dahl, V. (1999) The logic of language, in *The Logic Programming Paradigm,* Apt, Marek, Truszczynski, Warrens, eds, Springer-Verlag.
12. Harman, D.K.(Editor) (1996) *The Fourth Text Retrieval Conference (TREC-4)* NIST Special publication 500-236.

13. Agosti, M., Crestani, F., and Melucci, M (1996) Design and inplementation of a tool for the automatic construction of hypertexts for information retrieval. *Information Processing and management*, 32(4):459-476.
14. Salton, G., Singhal, A., Buckley, C., and Mitra M. (1996) Automatic text decomposition using text segments and text themes. *Proc. Seventh ACM Conference on Hypertext*.
15. Gey, F.C., Chen, A.(1998) Phrase Discovery for English and Cross-language Retrieval at TREC-6, *Text Retrieval Conf*, NIST Special Publication.
16. Hosmer, D.W. & Lemeshow, S. (1989) Applied Logistic Regresion *JohnWiley & Sons*, NY.
17. Ng, H-T, Goh, W-b and Low, K-L. (1997) Feature selection, perceptron learning and a usability case study for text categorization. In Belkin, N.J., Narasimhalu, A.D.,Willett, P., eds. Proc 20th Annual Intl ACM SIGIR Conf on IR, Philadelphia, pp67-73.
18. Boros, E., Kantor, P.B., Lee, J.J., Ng, K.B., Zhao, D.(1998) Application of logical analysis of data to TREC6 routing task *Text Retrieval Conf-6* , NIST Special Publication .
19. Fain, V.S. and Rubanov, L.I. (1996) Activity and Understanding *World Scientific*.
20. McCallum, A., Nigam, K., (1999) Text Classification by Bootstrapping with Keywords, EM and Shrinkage. *Technical Report, Just Research* http://www.cs.cmu.edu/~mccallum.
21. Redey, G. (1999) iCTRL: Intensional conformal text representation language *AI 109* 33-70.
22. Boguslavsky , I.M. (1996) The sphere of action for lexical units. Studia philologica, Moscow, (In Russian).
23. Partee, B.H., ter Meulen, A., and Wall R.E. (1990) Mathematical methods in linguistics. Kluwer, Dordrecht.
24. Ciravegna, F, Lavelli, A, Mana,N, Matiasek, J, Gilardoni, L, Mazza, S, Ferraro, M, Black, W, Rinaldi, F, Mowatt, D. (1999) FACILE: Classification texts integrating pattern matching and information extraction IJCAI-99, Stockholm Sweden, 890-896.
25. Bair, J. (1998) Dimensions of KM technology Selection, GartnerGrouppublishing, Oct 1998.
26. Hayes P.J., Weinstein SP (1990) Construe/TIS: A system for content based indexing of a database of new stories. In Proc. Of the 2nd Conf IAAI.
27. Jacobs, P.S., Rau, L.F (1990) SCISOR: extracting information from on-line news. *ACM communications 33(11)*.
28. Grishman , R.(1997) Information extraction: Techniques and challenges. In M.T. Pazienza, ed. *Information extraction: a multidisciplinary approach to an emerging technology*. Springer-Verlag.
29. 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.
30. Romacker, M., Markert, K., Hahn, U., (1999) Lean semantic interpretation In *Proc IJCAI-99*, 868-875.
31. van Eijck, J. & Moore, R.C.(1992), Semantic rules for English. In Hiyan Alshawi, ed. *The core language engine* 83-115 MIT Press.
32. Creary, L.G, & Pollard C.J.(1985) A computational semantics for natural language. In Proc ACM-85 172-179.
33. Sondheimer, N.K., Weischedel R.M., and Bobrow, R.J., (1984) Semangtic interpretation using KL-One. In *COLING-84* 101-107.
34. Hirst, G. (1988) Semantic interpretation and ambiguity *AI* 34(2):131-177.
35. Buvac, S.(1996) Quantificational logic of context. In *Proc 13th AAAI Conf* Menlo Park CA.
36. McCarthy J. (1993) Notes on formalizing context Proc IJCAI –93 Morgan-Kaufmann.
37. Levesque, H.J., Pirri, F.(eds) (1999) Logical foundations for cognitive agents. Springer-Verlag Berlin Heidelberg.