DIMACS Technical Report 99-32
June 1999

Natural Language Understanding with the Generality Feedback

by

Boris Galitsky[1]

[1]DIMACS Center, Rutgers University, 96 Frelinghuysen Road, Piscataway, NJ 08854-8018.
(galitsky@dimacs.rutgers.edu)

DIMACS is a partnership of Rutgers University, Princeton University, AT&T Labs-Research, Bell Labs, Telcordia Technologies (formerly Bellcore) and NEC Research Institute.

DIMACS is an NSF Science and Technology center, funded under contract STC-91-19999; and also receives support from the New jersey Commission on Science and Technology.

ABSTRACT

This study addresses the problem of improving the quality of natural language (NL) understanding by means of the involvement of the additional criterion of correct formal representation of the input inquiry. This criterion is based on the logical compatibility between the input inquiry, translated into the formal language, and the domain, encoded in the same language. Inquiry generality is measured as a normalized number of object tuples, which deliver the translation satisfaction for a given domain. Computational experiments in various domains show that the certain diapason of generality indicates the proper translation. Specific heuristics are developed for the transformation of the translation formula to improve the value of the generality criterion.

Usually, if the generality is too high (there is a large number of the tuples of objects) then the semantic analyzer has likely ignored some syntactic constraints for the translation formula. On the contrary, when the generality is too low (there is no object tuples, satisfying the translation), then the semantic analyzer has likely introduced too many syntactic constraints, and some of them have to be eliminated.

Generality criterion helps in the frequent situations, where the morphological and syntactic processing is insufficient for the construction of the proper inquiry translation into the formal language. Our approach is based on the experimentally verified fact that the user poses inquiries with the reasonable generality. This knowledge helps us to improve the semantic analysis to function in the flexible and expandable domains, where the lexical information can be limited.

Compatibility feedback by means of generality control poses the specific requirements to the formal language of inquiry translation and domain knowledge representation. The semantic processor is implemented as a logical program with metalanguage support to present the complex semantic rules. Suggested approach allowed us to build the NL understanding system with high inquiry complexity, involving up to 3-4 concepts.

Introduction

It is well known that the domain knowledge representation strongly affects the quality of natural language understanding. In this study we demonstrate the natural language (NL) understanding in the **narrow, but highly logically complicated problem domain** (PD) and verify the top level of complexity of the NL inquiries our system can handle. In such an environment, the compatibility feedback becomes essential to control the semantic analysis.

Evidently, the problem of NL understanding poses the highest level requirements to the expressiveness of the formal language, which is the basis for both NL inquiry translation and knowledge representation of domain in our design. If the formal language is sufficiently expressive, the criteria of the proper inquiry understanding (translation from NL into this formal language) can be set as the generality control of the translation formula. The existence of the **best generality of a translation formula** has rather computational than analytical nature. Compatibility of a translation formula with domain representation implements the feedback in the understanding procedure. It is the way to control the generality of NL translation as the key criterion to compensate the disagreements in the syntactic pre-processing. We use the first-order language, extended with abstract metapredicates (their arguments range over arbitrary well-formed formulas) and call it ML.

We follow along the lines of traditional requirements for ML (Woods, 1979a, Warren & Pereira, 1982):

- ML must be capable of representing precisely, formally and unambiguously any information presented by an inquiry.
- ML should facilitate the canonic translation from the syntax representation language.
- It should facilitate subsequent application of reasoning in the course of the process of semantic analysis.

The idea of the feedback to an earlier processing step is helpful in various understanding problems, in particular, under the scene analysis. Return to a previous processing step has the following scenario. Uniformly processed image (e.g. edge detection) is followed by the formation of features and the attempt to satisfy a rule (definition) of a scene object. When this rule is partially satisfied, the image is subject to the repetitive processing in accordance to the obtained hypothesis, though with a low likelihood. Such scenario leads to the robust solutions of the understanding problems of various nature (NL, image and speech, see(…)).

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.
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 intensive use of metaprogramming technique in the implementation of the semantic analyzer, we call it metalanguage-supported natural language system (MSNL).

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 these additional information does not influence 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.

Our approach is illustrated by the three strongly distinguishing domains. The first is the chemical plant control; it includes apparatuses, reagent flows, there parameters and the algorithms of process control. The second domain is the economical and political relationships of the countries, various interactions, interests, strategies, etc.; the questions includes comparative analysis such as relations to each other, influences on each other, common and distinguishing properties of countries, organizations, leaders and events. The third domain is based on various stories, anecdotes and jokes, which are formalized involving the concept of formal scenario (see Galitsky, 1998).

1. Posing NL understanding problem in a limited domain

The key feature of NL understanding system we intend to achieve is the **flexibility** degree in respect to the **different PD** (Galitsky, 1992). The current study investigates the NL system, which can be adjusted to the particular PD by a customer, without the help of the system designer. It is performed by defining a new fact, a new object and a new concept in NL. A new concept is expressed via the old ones, already known by 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 of MSNL 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 (compare, for example, with Miikkulainen 1994).

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. Translation from NL to ML

Let the input inquiry be an ordered set of words (or the parsing tree of a sentence) W. In particular case, MSA gives us the triples *subject-predicate-object* or the couples *subject - related to - object*. Though MSA assigns the semantic roles to the words based on the general knowledge of the corresponding meanings, semantic processing reassigns the attribute *object* and *feature* based on the specific knowledge of the strictly limited PD.

Semantic processing builds the mappings from the subsets of W and Cartesian products of W into the symbols and syntactic relations of our language ML correspondingly.

(1) $Features \subseteq M \rightarrow$ set of predicate constants $\{p_k\}$.

(2) $Objects \subseteq M \rightarrow$ set of object constants $\{a_s\}$.

(3) Syntactic links between *features* $W \times W \rightarrow$ set of equalities $(X_{su} = X_{tv})$ for the pairs of arguments X_{su}, X_{tv} of the predicates p_s , p_t and the set of expressions $\boldsymbol{p_s}(\ldots, p_t, \ldots)$ if $\boldsymbol{p_s}$ is a metapredicate.

(4) Syntactic links between *features* and *objects* $W \times W \rightarrow$ set of equalities $(X_{su} = a_s)$.

k,s,u,t,v are the indexes($\in$ Z). In addition, we have the following ("inquiry oriented") mappings:

(5) "and", "or", "no", "neither, only, minimal, etc. $\rightarrow$ set of propositional connectives.

(6) "which", "how", "what", "when", "where", etc. $\rightarrow$ the key (accent) non-instantiated arguments of the predicates from (1).

We give the examples of the mappings (1)-(4). Note that the word combinations are firstly mapped into the pairs of words and than to ML:

(1) "follow" $\rightarrow$ *flow(Reagent, ApparatusFrom, ApparatusTo);*

"absorber2" $\rightarrow$ *absorber2;*

(2) "flow to cool" $\rightarrow$ "flow","cool" $\rightarrow$ *(Reagent= ReagentCool), flow(Reagent, AppFrom, AppTo), cool(ReagentCool, AppCool, TempChange);*

(3) "cool in a heatexchanger" $\rightarrow$ "cool","heatexchanger" $\rightarrow$ *cool(ReagentCool, AppCool, TempChange), AppCool=heatexchanger.*

This kind of processing is traditional; it converts the syntactic representation into semantic one (see, for example Woods, 1979a, Woods, 1979b, Grosz, 1977). The main distinguishing property of our approach is the usage of metapredicates to express the NL concepts with the higher level of abstraction, then the first-order predicates. We present the framework of semantic mapping, which allows one to consider substitution of first-order variables and metavariables in the uniform basis.

Below is the procedure of deriving an element of a series of formula translations $\{T_j\}$, $W \rightarrow T$ in the language ML. Underlined transformations can be performed or not; correspondingly, the series of translations is generated. This way we cover the possible semantic ambiguity of the inquiry.

1. We build the starting translation as a formal conjunction of predicates from (1): $p_1(X_{11}, \ldots, X_{1n}) \& \ldots \& p_k(X_{1k}, \ldots, X_{kn})$. Predicates initially occur in the same order as the corresponding words from W occur in the NL inquiry (or in the MSA resultant tree).
2. If p_s and p_t are connected by the connective "or", then <u>p_t is moved behind p_s. Disjunction is inserted between p_s and p_t</u>. If there is a negation, found by MSA to be related to the word - proimage of p_q, then <u>*not* is inserted in the translation before p_s</u>.
3. If p_q is found to point to the question target (syntactically linked with the component (6) below, then <u>p_q is moved to the rightmost position</u> in the translation. One of the variables of p_q is considered to be the inquiry *target*. For example, "What pressure" $\rightarrow$ *pressure(Value,...), Value* is the inquiry target.
4. If there is some object constant a_s of the type τ_s such that there is no predicate p_k in the translation T_j having an argument of this type, then the term $(X_{su} = a_s)$, followed by the new *predicate* $p_s(X_{s1}, \ldots,$

X_{su} ,…,X_{sn}), is added to the translation, X_{su} is of the type τ_s.

5. If p_s and p_t are found by MSA to be mutually dependent, and they are of the same type, then there exists at least one variable X_{su} from p_s and X_{tv} from p_t such that $X_{su} = X_{tv}$. This equality should be inserted before p_s or p_t , whichever occurs earlier in T_j. We obtain the representation

   $\{p_1(X_{11}, \ldots,X_{1n}) \&\ldots\&(X_{su} = X_{tv})\& p_s(X_{s1},\ldots, X_{su},\ldots,X_{sn})\&\ldots\& p_t(X_{t1},\ldots,X_{tv},\ldots,X_{tn}) \& p_k(X_{1k}, \ldots,X_{kn})\}$

   If p_s and p_t are mutually dependent and $\boldsymbol{p_s}$ is a metapredicate, then p_t (and its preceding equality terms) are assigned to be equal to X_{su}. The translation looks like

   $\{p_1(X_{11}, \ldots,X_{1n}) \&\ldots\&(X_{su} = \boldsymbol{p_t}(X_{t1},\ldots,X_{tn}))\& p_s(X_{s1},\ldots, X_{su},\ldots,X_{sn})\&\ldots\& p_t(X_{t1},\ldots,X_{tn}) \& p_k(X_{1k}, \ldots,X_{kn})\}$

   There is no type verification for the substitution of a formula into a metapredicate

6. If p_s and a_t are found by MSA to be dependent, and they are of the same type, then there exists at least one variable X_{su} from p_s such that $X_{su} = a_t$. This equality should be inserted before p_s, where X_{su} firstly appears in the translation. We derive

   $\{p_1(X_{11},\ldots,X_{1n})\&\ldots\&(X_{su}=a_t)\&\ldots\&(X_{su} = X_{tv})\&p_s(X_{s1},\ldots, X_{su},\ldots,X_{sn}) \&\ldots\&p_t(X_{t1},\ldots, X_{tv},\ldots,X_{tn}) \& p_k(X_{1k}, \ldots,X_{kn})\}$

   The complicated situation appears in case of antisymmetric predicates. We introduce the following convention to their instantiation states: the first antisymmetric argument (possibly) gets the value from the left predicate, the second argument derives the value and (possibly) passes it to the right. There is a strong semantic correlation: these arguments are correspondingly **active** and **passive**. We use the prepositions from W to determine, which link to form: with the first or with the second antisymmetric argument. For example, "follows from absorber3" $\rightarrow$ *flow(Reagent, absorber3, ToApparatus)*, "follow to heat_exchanger1" $\rightarrow$ *follow(Reagent,FromApparatus, heat_exchanger1)*, "influence on pump2" $\rightarrow$ *influence(Reagent, Apparatus, pump2).*

7. Each predicate in the translation is verified not to be in the *failure instantiation state*. It could be achieved by varying the order of predicates in the translation. Any MSA result (subgraph), leading to the *failure instantiation state* of a predicate in the translation, should be disregarded.

This is the end of the briefly presented translation procedure. In addition to requirements of 5) and 6), we demand the satisfiability of the given stand-alone predicate with the linked or substituted argument, if other arguments are non-instantiated, or non-satisfiability of this predicate with all non-instantiated variables.

Participation of the metapredicates significantly increases the language expressiveness and precision of the optimal translation search. We can use a simple combinatorial estimate of the language expressiveness as the ratio between the number of "meanings" (properly constructed formulas) and the number of constants. This ratio increases when we introduce the metapredicates in the translation. Furthermore, there is the series of concepts (including mental states and actions), which cannot be expressed without the involvement of metapredicates.

Note, that the decision concerning the assignment of a particular word to a predicate or metapredicate occurs within a narrow PD only.

We verify the resultant translations in respect to compatibility with the domain representation (the logical program of PD).

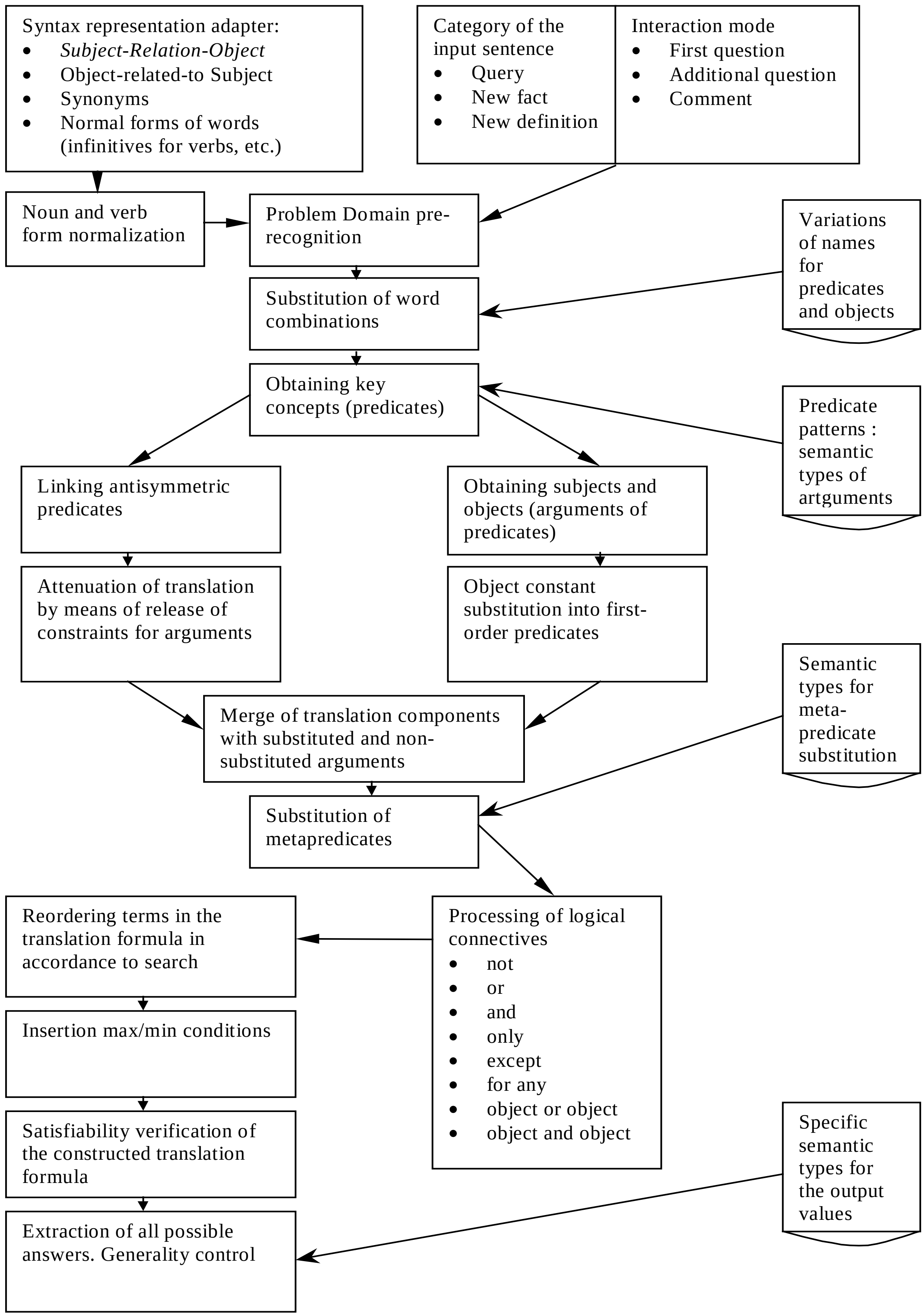

Fig.1 The general scheme of the semantic processor with generality control for functioning in the pattern-resolvable domains (see the Table 1 for the description of units, pp.9-12 and 15). Four units, storing the data for semantic representation, are on the right.

Unit #	**Unit name**	**Unit function and implementation** (see http://www.dimacs.rutgers.edu/~galitsky/NL/unit.html for details)
1	Syntax representation adapter	Converting the parsing tree from the implementation-specific representation into the canonical representation to extract the predicate-argument and predicate-predicate links
2	Category of input sentence	By default, the input sentence is an inquiry. It can have an arbitrary structure, providing easy NL interaction, in contrast to the domain extension, which introduces some constraints for the occurrence of new words in the definition of a new fact, a new object or a new concept. The latter is defined via the old ones, known to the system; processing of the defining (right) part of the definition is similar to that of the inquiry.
3	Interaction mode: • First question • Additional question • Comment	MSNL allows the user to divide an inquiry into parts, the objects of interest can be referred in the sequential queries with the additional constraints or released constraints. Besides, if the system gives an unsatisfactory response, and the user believes the system could have the inquiry-related knowledge, then he/she is allowed to reformulate the question. This repetitive question can contain the references to the facts and to the objects from the failed question. If the user is not satisfied with the system's knowledge concerning the particular fact, he can input another fact he wishes the system to know, using the objects, mentioned in the inquiry.
4	Noun and verb form normalisation	MSNL is invariant in respect to a particular natural language and does not store the forms of nouns, verbs, etc. unless it is critical for the functioning in a particular domain. In that case each form for noun or a verb is assigned the meaning, specific for this domain and stored within its representation. This property of MSNL makes the semantic analysis component independent on the particular domain representation.
5	Problem Domain pre-recognition	MSNL can function in the multiple narrow domains at the same time. Each domain is assigned a specific set of relations or objects. Since MSA is assumed independent on the particular narrow domain, pre-recognition occurs after MSA, where the specific set of terms for each domain is matched with the input inquiry. After an appropriate domain is revealed, it is fully loaded to the system, including semantic patterns for each predicate and all objects for each semantic type.
6	Substitution of word combinations	Besides the parsing tree, MSA delivers the list of general synonyms to MSNL. The quality of understanding raises dramatically, when the synonym substitution occurs either after the semantic analysis of context or after the problem domain pre-recognition. MSNL analyses synonyms for objects, taking into account their semantic types, and synonyms for predicates, involving the corresponding metapredicates, etc. Therefore, the domain representation (subject to the on-line update by a user) contains within itself the lists of word combinations with the analogous meanings.
7	Obtaining key concepts (predicates)	The normalized words of the sentence are matched for the set of all predicates for the domain. If a sentence includes the predicates from the other domains, we conclude that they do not contain information (no new knowledge, no additional constraints or links) within the given semantic model for the pre-recognized domain. If it is not the case, then we conclude that our semantic model does not cover this sentence. Such a situation should not be considered as a misunderstanding. Either the user has to be notified that he/she is out of

		domain bounds, or the domain representation should be reevaluated. Frequently the latter is impossible because of internal contradictions or ambiguities, inappropriate within a pattern-resolvable domain in accordance to its definition. We note that the problem of covering a large problem domain by the family of the narrow ones can be resolved on the level of predicates, objects do not have to be involved in the domain pre-recognition. Furthermore, the metapredicates of mental states and actions are invariant in respect to narrow domains.
8	Linking antisymmetric predicates	Note that MSNL is the rule-based semantic processing system. Analysis of the antisymmetric predicates is one of the most complex subtasks towards the translation construction, involving the propositions and unary predicates, which link the symmetric predicates together. We demonstrate the three following possibilities of the links between the couples X_{s1},X_{s2} and X_{t1},X_{t2} of arguments for the pair of antisymmetric predicates p_s and p_t : $p_s(X_{s1},X_{s2},\ldots,X_{sn})\&\ldots\& p_t(X_{t1},X_{t2},\ldots,X_{tn})$. We call the last expression **translation template.** These cases are: 1) $X_{s1}=X_{t1}$; 2) $X_{s2}=X_{t1}$; 3) $X_{s2}=X_{t2}$; We present the examples from the political domain: 1) Which country started war before weapon agreement with Iraq? $war(X_1, X_{s2},\ldots)\&\ldots\& agreement(X_1, X_{t2},\ldots) \& X_{t2}=iraq$. 2) What is the export to the country which launched the air-to-air missile? $export(X_{s1}, X_{s2},\ldots)\&\ldots\& launch(X_{s2}, X_{t2},\ldots)$. 3) Which country exports to the country that receives import from China? $export(X_{s1}, X_{s2},\ldots)\&\ldots\& launch(X_{t1}, X_{s2},\ldots), X_{t1} = china$. There are specific rules, which take the translation template, verify some constraints of the propositions, related to these predicates, the other (non-antisymmetric) predicates, related to both of them, to conclude which case to involve. Note that the analysis of argument substitution occurs independently.
9	Obtaining subjects and objects (arguments of the predicates)	As only the predicates has been revealed, MSNL generates the list of all objects for all semantic types for all arguments. This list is then matched against the input sentence to **extract** the **object references** from this sentence.
10	Object constant substitution into first-order predicates	After all the objects for all semantic types involved are found, MSNL determines the belonging of the object to the particular predicate among the set of all predicates, which have the semantic type for this object. There is a series of rules determining the belonging preference in case it has not been established by MSA. Additional **substitution control** is required in addition to *object-predicate-subject* triples, revealed by MSA. The more arguments are substituted by the objects, the **less** the **translation generality**. Therefore, substitution unit is the first one in the feedback loop of the translation satisfaction. The substitution occurs into the translation template (uninstantiated translation formula), where each argument is unique (there are no arguments with the same value).
11	Attenuation of translation by	The template formula is the most constraint one (least general). Before the attempts to satisfy the translation in the pre-recognized domain, MSNL applies

	means of the release of the constraints for arguments	the series of heuristic rules to attenuate the links between the predicates. MSNL transforms the formula, making two arguments of the same semantic type for two predicates to turn into different value (**diversify** these arguments). If MSNL returns to this unit because of the satisfaction failure, this transformation will be applied repeatedly until each combination of arguments will be diversified. The following constraint stops the diversification transformation: for any predicate its argument should exist, which is the same as some other argument (of the same semantic type) of the other predicate.
12	Merge of translation components with substituted and non-substituted arguments	The unique feature of our approach is that the analysis of the links (mutual context) of the predicates is implemented completely separately from the determination of the argument substitution. Each of these operations has the specific semantic rules, and each of them is intended to correct MSA. After MSNL completes these operations, the formal **unification** procedure is initiated: *Translation_with_linked_predicates* = *Translation_with_substituted_arguments* The unification procedure merges the results of predicate linking and argument substitution. If a subject is related to more than one object or more than one predicate (part of speech), this fact is reflected by unification procedure. If one argument is substituted, and the other argument of the other predicate has the same value, then this other argument gets substituted as well. Direct semantic rule, which would call the same result, could be inconsistent with the semantic rules for predicate linking and/or argument substitution. Splitting of the formula translation procedure into these procedures provides the clarity and consistence of the semantic rules for each of these procedures.
13	Substitution of metapredicates	Using metapredicates in the formation of translation is the essential capability of MSNL. The metavariables of metapredicates have the specific **abstract semantic type** to be able to range over arbitrary formulas. These formulas are constructed in the units, following the current unit. The metapredicate processing starts with the substitution of one predicate into the abstract semantic type of the other predicate (that is indeed a metapredicate). It occurs in the argument substitution unit in accordance to the same semantic rules as with the objects of the regular semantic types. The only difference between the substitution into predicates and into metapredicates is the following. MSA helps in much lesser degree, because the syntactic relations between the words are weaker correlated for the metapredicates than for the first-order predicates. This unit replaces the abstract argument of a metapredicate, instantiated by the symbol of a predicate (just the predicate name), by the whole term (this predicate with its arguments) in accordance to the example *(want(Smb, own), own(Smb, car))* $\rightarrow$ *want(Smb, own(Smb, car)).*
14	Reordering terms in the translation formula in accordance to search	This unit performs the task of the term reordering to achieve the proper **instantiation state** (See …). If there is a pair of predicates with the common arguments such that one of them **yields a value** (and verifies a constraint) and the other one **verifies a constraint** (does not yield a value), then the former predicate should be followed by the latter predicate in the translation. It is necessary to assure that the formula will be satisfied if it can be potentially satisfied by a set of objects (and formulas, if the translation includes metapredicates). In other words, starting from this unit, procedural semantics

		should be taken into account to provide the proper satisfaction process of the translation formula. For example, the formula *less(Temper, 70), heat(acid, heatexchanger1, Temper)* is converted into *heat(acid, heatexchanger1, Temper), less(Temper, 70).* The first predicate yields the value of *Temper*, and the second one verifies the constraint (The temperature should be less than 70). Note that the input translation would always fail because the predicate *less* with an uninstantiated argument (first or second) always fails.
15	Processing of logical connectives • not • or • and • only • except • for any • object or object • object and object	MSNL analyses logical (propositional) connectives and constraints at the finalizing step of building translation formula. Processing of the logical connectives is the additional illustration of the fact that a very superficial involvement of the logical connectives in the MSA is followed by the thorough evaluation by MSNL. Logical connectives are subject to much more constructive analysis when the translation formula has been built (without the connectives). We mention, that default connectives in the translation formula are conjunctions. The semantic model contains the scope of propositional connectives for predicates and for their arguments (e.g. *import of oil or iron, import and export of oil, capital is not Washington, not an agreement*). If a connective links two predicates, corresponding symbol is inserted between them in the resultant translation. If two objects are logically linked, then the translation includes the duplicate of the corresponding predicate with the same arguments, except for these objects. To handle the concept only, we add an additional expression in the inquiry translation. It finds all the values of the argument under the scope of only and verifies that this list contains the object, specified in the inquiry or any single object.
16	Insertion of max/min conditions	The semantic analyzer extracts the requirements for search of an object possessing maximum or minimum of a numerical value. The expression, which has to be added into the inquiry translation to implement the min/max condition, has the procedural nature and is not explicitly mapped into by the input inquiry. The following expression is added to the end of the translation *[! findall(Xnum, (MaxObjectFormula) , Xs) !], Xs\==[], quicksort(Xs,Sorts), last(Output, Sorts)* to implement the *maximal* condition.
17	Satisfiability verification for the constructed translation formula	MSNL runs the resultant translation against the domain knowledge representation. Depending on the instantiation state, each term of the translation is matched against the corresponding fact or clause head in the logical program of the domain knowledge representation. This match either finds the values for the arguments or verifies the compatibility of values the translation formula contained prior to the current satisfiability procedure. The ability to answer the ”find an object” questions versus “yes-no” ones is built-in for our system architecture (and PROLOG system itself).

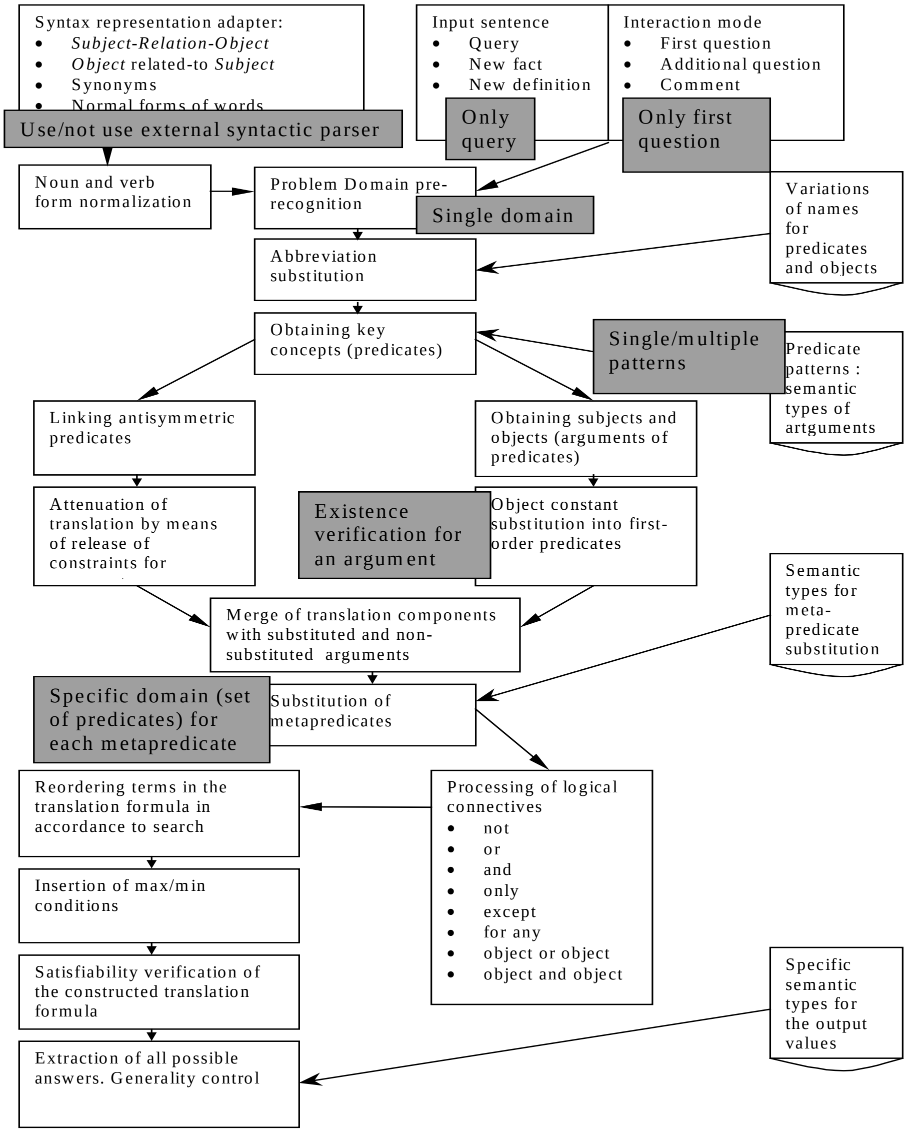

Fig.2 Control parameters of MSNL system. They provide the domain flexibility, domain modification and domain extension features. Semantic processing unit is designed to be maximally independent from the syntactic and knowledge representation ones. The control parameters are chosen so that their modification provides the domain flexibility features. To run the semantic processing in a new domain, the following operations are required:

1) The semantic model of the new domain is determined, as well as the relations and overlapping with the other domains. Verification of the uniqueness of the new domain in the sense of the set of patterns, the set of semantic types and the superposition of existing domains has to be performed
2) The new domain must be characterized as being narrow, small or pattern-resolvable one. Otherwise, either the new domain must be splitted or some limitations should be enforced for the inquiry complexity.
3) The control parameters of the semantic processing unit should be adjusted for the new domain.

Therefore, neither specific lexical information nor semantic peculiarities affect the adaptation to a new domain. These invariant properties are provided by generality control, reducing the domain adaptation by modifying the parameters depicted above.

Semantic processing system can accept the syntactic analysis results in various formats as well as an original NL inquiry; generality criteria affect the internal syntactic pre-processing and domain pre-recognition. Semantic system functions in the various modes of interaction:

1) A user can ask question, add a new fact or define a new concept in NL;
2) A question can be answered in one iterations or with multiple iterations, if a user is knowledgeable of releasing or strengthening of the inquiry conditions.

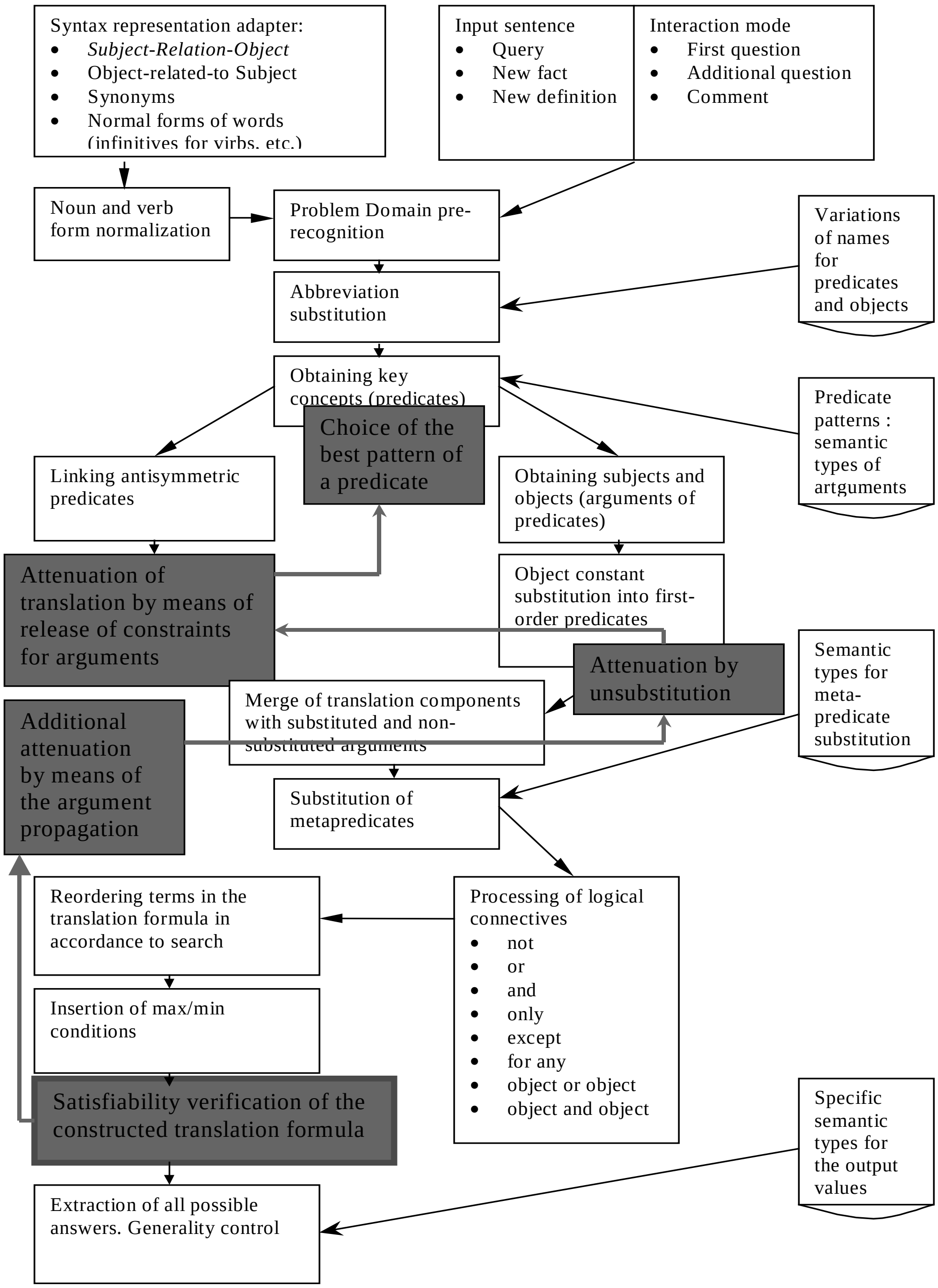

Figure 3. Feedback (backtracking) towards the achieving the translation formula with optimal generality. To achieve the optimum, the system firstly performs attenuation by means of the argument propagation, turning two equal arguments of the pair of predicates into the different values such that the other predicates are not affected. Then the system returns to the argument substitution unit and unsubstitutes an argument, assuming that the corresponding object was wrongly related to a predicate by MSA. Returning to the earlier processing units, the system releases some syntactic constraints for the translation formula. This operation is followed by the reevaluation of the pattern choice for the pattern-resolvable domains.

18	Extraction of all possible answers. Generality control. Interface with human and software agents	In this unit, MSNL assumes that the translation formula has been confirmed. Then MSNL searches for all object tuples. The previous unit of the satisfiability verification has accepted current inquiry generality. Therefore if there are too many or too few (none) tuples of objects, then it would not mean that the understanding is wrong but that additional strengthening / release of constraints is required in the interactive mode. There are three types of the system output: 1) Names of objects (parameter values). 2) Generated series of tables of objects and relations with the cell embedding in accordance to occurring metapredicates. This form of output format matches the traditional semi-structured data representation (for example, XML language, see www.xml.com). See Figure 5 for a generated data table. 3) Paragraphs of text, corresponding to the formulas (semantic headers) of the knowledge representation

Table 1: the units of the semantic processing system (from Fig.1-3).

The linear order is introduced on the set of all translations based on the compatibility *degree* with the problem domain. First, each variable is assigned by its weight for the score calculation: leftmost variable has the weight of 1/2, and the rightmost variable (the inquiry focus) weight is 1. If the call to a translation succeeds, its score is calculated as the weight sum of the number of constants, satisfying each variable. If the call to a translation fails, than it is subject to metapredicate ***attenuate*** (see Galitsky 1998), and the score is calculated for the attenuated formula as a negative weight sum.

The computational experiments show that **the best inquiry understanding corresponds to the translation with the positive score, approaching zero.**

There are following motivations for the presented understanding criterion. We take advantage of the fact that typically the inquiry author expects to obtain rather limited set of data. If it is not the case, the understanding system switches to the interactive mode. We base the generality rating of inquiry translation on the comparison of the generated data set with the data set, highlighted by the user in case he knows all the facts in the PD. This is a domain-invariant way to reduce the NL ambiguity under the interactive question answering.

Suggested approach is fruitful for the *interactive inquiry* system. As only the system makes decision about the optimality of the particular translation, it outputs the resultant data together with the comments concerning the perceived level of generality. A user can then explicitly raise or drop that level, reformulating the question in more general or concrete terms. Finally, the system translates the reformulated inquiry and finds the intersection of the translation sets for original and reformulated inquiries in the positive vicinity of zero score. The inquiry understanding practice showed the fast convergence to the proper translation (3 or less iterations) under the interactive inquiring.

The generality control approach achieves the **high flexibility and independence on PD** under processing of **definitions,** using which a user could significantly extend a domain. The translation procedure transfers the complexity focus from the linguistic analysis to the reasoning one, reducing the task of understanding the definitions to the analysis of the defining part as an inquiry.

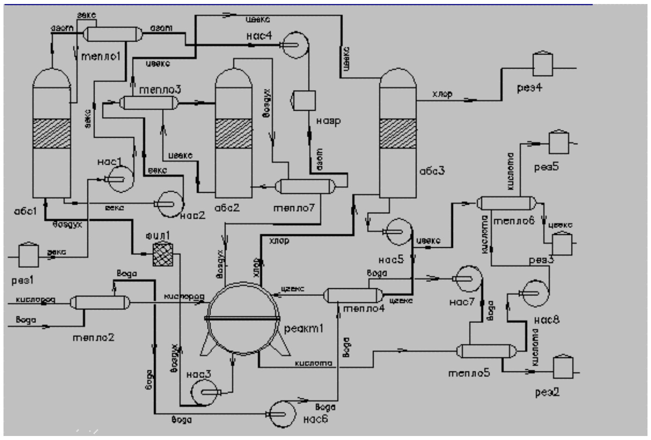

Figure 4. The scheme of the control process. The questions of the control operators involve various parameters of apparatuses, reagent flows, etc. In addition, operator personnel can ask about the interaction between apparatuses and their mutual influence. New process parameters, as well as the new control algorithms (operation sequences), can be entered via NL

3. Examples of inquiries and definitions

3.1 The process control domain

We control the process, involving apparatus, reactors, heat exchangers, reservoirs, pumps, etc., which are characterized by the reagent flows, temperatures, pressures, etc. The definitions of the apparatuses and flows of the process are presented in the form of clauses at http://dimacs.rutgers.edu/~galitsky/MS/t.ari

We present the NL sentences and their translation into ML.

“How is the temperature of reagents in the reaction vessel positively influenced if the acid flows to the absorber and is warmed by less then 50 degrees”

question:- Reagent=acid,
warm(Reagent, AppWarm, TempChange), TempChange<50, AppWarm=AppTo,
flow(ReagentFlow, AppFrom, AppTo),
absorber(AppTo), ReagentFlow=Reagent, ReagentInflTo=ReagentFlow, AppInflTo=AppFrom,
influence(ReagentInflTo, AppInflTo, ReagentOn, AppOn), reaction_vessel (AppOn),
ReagentOn=ReagentTemp,AppOn=AppTemp,
temperature(ReagentTemp,AppTemp,T).

We proceed to the definition of the concept “to engage a pump". “A pump, from which a reagent flows to the absorber2, is **engaged**, if the results of the reaction must be cooled.”

engage(Reagent,App) :- pump(App), AppFrom=App,
flow(Reagent, AppFrom, AppTo), AppTo=absorber2,
resolve(ReagentIn, ReagentOut, AppReact), ReagentIn=Reagent, ReagentCool=ReagentOut,
cool(ReagentCool, AppCool, TempChange), AppReact\=AppCool.

We translate the right part of the definition as an inquiry: "What reagent can flow from a pump to absorber 2, if the results of the reaction are cooled".

Not only *App*, but also *Reagent* is transferred to the newly constructed predicate. The introduction of *ReagentOut* would make the predicate ***engage*** not general enough to cover all the situations, where it is necessary to start a pump in a particular environment. We note that there are two ways to interpret this definition. The first way is to introduce the concept of engaging by the example of the pump with absorber and the special process. This way changes only the terms of the PD description. It is important to distinguish the semantics of the definitions when one says “It is necessary to engage a pump, when..." instead of saying "to engage a pump means, that...". In other words, awareness what is the reason, and what is the result ("pump is engaged, because…") is necessary both for proper translation of the right part and for the transfer of variables from the right (defining) part to the left part (being defined). The second possible interpretation assumes adding of the new data in PD using a new concept. In that case the new predicate does not bring in new constraints to its variables. The example above shows, how a new control procedure can be introduced, called "to engage a pump".

Now we are able to process an inquiry with newly defined concept: "How many degrees warmer would the acid get in those heat exchangers, which are affected by the **engaged** pump?"

question :- pump(Apt),
engage(Reagent,Apt), ReagentInflTo=Reagent, AppInflTo=Apt,
influence(ReagentInflTo, AppInflTo, ReagentOn, AppOn),
AppWarm=AppOn,ReagentWarm=ReagentOn,
warm(ReagentWarm, AppWarm, TempChange).
TempChange - ?

We finally present the inquiry with metapredicates *want* and *know*. "For what apparatus does a controller want to know that the water temperature does not influence the reaction vessel?

want(controller, Intention),
Intention= know(controller, ControllerKnowledge),
ControllerKnowledge = (
Reagent = water, temperature(Reagent, Apparatus, T), Reagent = ReagentInflTo, Apparatus = AppInflTo,
not influence(ReagentInflTo, AppInflTo, ReagentOn, AppOn), reaction_vessel (AppOn)).
Apparatus - ?

As we mentioned in the introduction, the translation formulas for the inquiries above are approximations, which ignore the details in a degree specified by the PD representation. In particular, we skip the quantifiers, temporal and some deductive relationships to avoid unnecessary complications.

3.2 Questions in the political domain: involving mental attributes of the countries

The question "who informed US about the export of weapon from China to Iraq" is represented as *inform(Xcountry, us, export(china, iraq, weapon, xnum)),* involving the patterns *inform(xcountry, xcountry, xabstr)* and *export(xcountry, xcountry, xproduct, xnum).* Note that the formula *export(china, iraq, weapon, xnum)* is substituted instead of the metavariable (semantic type *xabstr*) of the predicate *inform.* See Figure 5 for more examples of the questions.

4. Related systems and conclusions

Generality control is found to be an important feature of NL interaction. The compatibility feedback, enforcing the semantic analysis to form the least general (most concrete) but satisfied inquiry translation is required to handle the logically complex situations. They involve the presence of two or more concepts in an inquiry, which has the common context (linked by a common argument of the pair of predicates, representing these concepts). Computational experiments show that such inquiries are handled much more robustly in a narrow domain, than the traditional sequential analysis performs. Coarse combinatorial estimate gives us the elimination of 25 possible errors of MSA for an inquiry with two predicates with 4 arguments within 2 semantic types each. If we statistically analyze the inquiries for all three domains, presented in this paper, we gain more than 30 % of the understanding accuracy, using the generality control via compatibility feedback. Even assuming the MSA performs ideally, we eliminate at least 15 % of errors applying this approach in the situations with the explicit ambiguity that cannot be resolved without human intervention.

Metalanguage-supported natural language system

Ask Question | Clear

About ETAP

Ask Question with ETAP syntax engine | ea

- Where did Iran claim to perform the military exercise ?
- Tell me about the Iranian hostage by Taliban in 1998 ?
- **Which country compets with Taliban for export of oil?**

MSNL Response:

	Country with	Country with	abstr				
compet	**Iran**	taliban		country from	country to	trade	num
			export	Iran	Taliban	oil	unknown

Competition between Iran and the Taliban over Caspian Sea oil pipeline routes.

- Which country wanted no support by Iran of Afghanistan ?
- Do you know if Iran wanted not sending refugees by Taliban ?
- Who supported attacks on Iran ?
- Did Iran claim that it did not kill the refugees?
- What did six plus two want to investigate ?
- Who wanted to investigate who have persecuted Shi'a ?
- When did Iran want a military exercise on the border with Afghanistan ?
- When did Iran claim that Iran attack Taliban ?
- Whom did Taliban ask to help ?
- What did Taliban ask UN about the help to Taliban?
- Whom did Taliban ask about the help to Taliban because of the hostage ?
- When did Iran want a military exercise on the border with Afghanistan ?

Knowledge representation is performed via semantic headers (shown in bold):

war(afghanistan, taliban, civil, 1998):-iassert($The Taliban, a Sunni Islamic revolutionary group in Afghanistan, has been fighting (and winning) a civil war in Afghanistan. By August 1998, they had gained control of all but the fringes of northern Afghanistan. The Taliban are not the recognized government of Afghanistan, and they maintain cordial relations only with Saudi Arabia and Pakistan.$).
win(Taliban, war(afghanistan, taliban, civil, 1998)):-iassert($The Taliban, a Sunni Islamic revolutionary group in Afghanistan, has been fighting (and winning) a civil war in Afghanistan. By August 1998, they had gained control of all but the fringes of northern Afghanistan. The Taliban are not the recognized government of Afghanistan, and they maintain cordial relations only with Saudi Arabia and Pakistan.$).

Figure 5. Web page of the MSNL System, operating in the domain of Iran-Taliban (1998) crisis. (http://www.dimacs.rutgers.edu/~galitsky/NL/tal_dom.html)

To draw the comparison with the functioning of the other NL understanding system in narrow domains, processing the inquiry with a pair of concepts, we mention Ask Jeeves (www.ask.com) and Microsoft English Query (http://microsoft.com/sql/70/whpprs/eqp.htm). The first system needs the template inquiry, explicitly set in advance, to properly determine the context for multiple concepts. The second system performs the translation from NL into SQL, followed by the SQL query launch. Therefore, the understanding quality cannot be controlled before the data is actually delivered to a user. Both these systems would gain the understanding accuracy of the logically complex inquiries using the generality control.

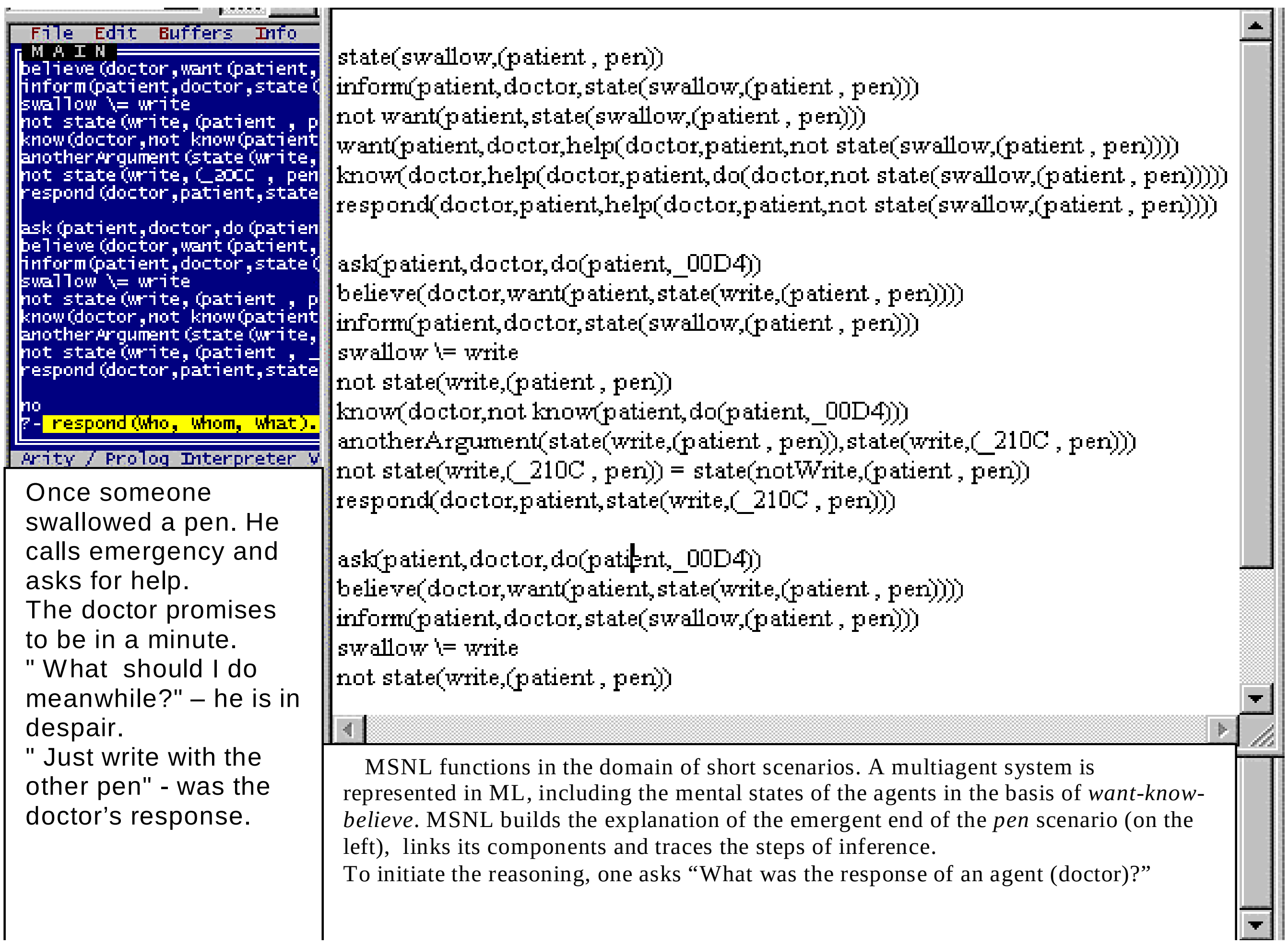

Figure 6. MSNL in the domain of formal scenarios. The domain is extremely compact, but the reasoning, required for question answering, occurs on the edge of inconsistency. Potential applications include the learning aids for autistic kids, see (Leslie & Thaiss,1992) for background information about the autism-specific scenarios.

Presented system is worth comparison with the other keyword-match approach, **latent semantic analysis**, based on the extraction and representation of the contextual-usage meaning of words by statistical computations applied to a large corpus of text (Landauer et. al., 1998). The overall information about the word contexts, in which a given word does and does not occur, provides a set of constraints that determines the similarity of meaning of words to each other. However, this methodology is insufficient for our problems, where the question answering requires more sophisticated reasoning than the substitution of one concept (or context) into another one. For the domains, where NL interaction is keyword-resolvable (inquiries contain just a single logical predicate from our viewpoint), generality control is also helpful for the search improvement.

Our approach to the context management problem (see Lenat, 1998, Buvac, 1996, McCarthy, 1993 for comparison), involving the **narrow – small – pattern-resolvable** chain of the domain criteria, is oriented to the NL understanding. The operations of transition from one domain (context, world) to another is rather concrete, than that of the logical formalisms of context. Two context management issues have to be resolved under question answering:

1) Context pre-recognition.
2) Verification that the determined context will not deliver understanding errors for a given inquiry.

Specific heuristics are developed for the transformation of the translation formula to improve the value of the generality criterion. These heuristics are mostly based on the observations of logical and statistical nature and therefore serve as a specific approximation of the human understanding of NL. Usually, an inquiry, which could be potentially wrongly handled in accordance to our heuristics, allows properly understandable reformulation.

To conclude, we mention the joint work of MSNL as a semantic component with the syntactic analyzer START and the knowledge representation subsystem OMNIBASE, developed in AI Lab at MIT (Katz, 1990; Katz, Yuret et. al., 1999) (www.ai.mit.edu/projects/infolab/gulf_ir.html) for the political domain. Combination of syntactic, semantic and knowledge engineering tools allowed one to answer questions of the high complexity, involving up to four concepts per inquiry.

Acknowledgements

The author is grateful to Lev Rubanov, Vitaly Fain, Igor Boguslavsly (Institute for the Information Transmission Problems), Israel Gelfand and Ilya Muchnik (Rutgers University), Boris Katz, Deniz Yuret and Sue Felshin (MIT) for fruitful discussions and comments on the manuscript.

References

1. Galitsky, B.A.(1998) The formal scenario and metalanguage support means to reason with it. *DIMACS Tech. Report #98-27*, Rutgers University.
2. Leslie, A. and Thaiss, L., (1992) Domain specificity in conceptual development: Neuro-psychological evidence from autism. *Cognition*, 43, 225-251.
3. Miikkulainen, R. (1994) Subsymbolic Natural Language processing *The MIT Press*, Cambridge, MS;London, England.
4. Galitsky, B.A. (1993) Understanding of definitions under NL interaction *proc EastWest AI Conf-1993*, pp.234-240
5. Woods, W.A.(1979a) Transition Network Grammars for NL Analysis *in Readings in NLP* Morgan Kaufmann Publ CA.
6. Woods, W.A. (1979b) Semantics and Quantification in NL Question Answering, *in Readings in NLP* Morgan Kaufmann Publ CA.
7. Grosz, B.I., (1977) The representation and use of focus in a system for understanding dialog, *in: Proceedings 5th IJCAI-77.*
8. Landauer, T.K., Foltz, P.W., Laham,D. (1998). Introduction to Latent Semantic Analysis, *Discourse processes*, 25, 259-284.
9. Galitsky B.A. (1999) Automatic generation of the multiagent story for the Internet advertisement. *Symposium on AI and creative language: stories and humor*, in AISB-99, Edinburgh, UK.
10. Lenat, D. (1998) The dimensions of context-space, *Cycorp technical report*, www.cyc.com.
11. Buvac, S (1996) Quantificational logic of context. In Proceeding the 13th AAAI-96, Menlo Park, CA.

12. Warren, D.H.D. and Pereira, F.C.N (1982) An Efficient Easily Adaptable System for Interpreting Natural Language Queries, *American Journal of Computational Linguistics*, vol. 8, num.3-4.
13. Katz, B.(1990), "Using English for Indexing and Retrieving," In P. H. Winston and S. A. Shellard, editors, *Artificial Intelligence at MIT: Expanding Frontiers*, volume 1, MIT Press, Cambridge, MA.
14. Katz, B., Yuret, D., Lin,J. , Felshin, S., Schulman, R., Ilik, A., Ibrahim,A. and Philip Osafo-Kwaako, P. (1999) Integrating Web Resources and Lexicons into a Natural Language Query System, *IEEE International Conference on Multimedia Computing and Systems.*