HIERARCHICAL MEANING REPRESENTATION AND ANALYSIS OF NATURAL LANGUAGE DOCUMENTS

Toyo-aki NISHIDA and Shuji DOSHITA

Department of Information Science
Faculty of Engineering, Kyoto University
Yoshida-honmachi, Sakyo-ku, Kyoto, 606, Japan

Abstract

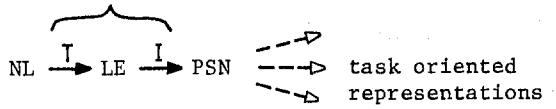
This paper attempts to systematize natural language analysis process by (1) use of a partitioned semantic network formalism as the meaning representation and (2) stepwise translation based on Montague Grammar. The meaning representation is obtained in two steps. The first step translates natural language into logical expression. The second step interprets logical expression to generate network structure. We have implemented set of programs which performs the stepwise translation. Experiments are in progress for machine translation and question answering.

1. Introduction

Conventional AI systems dealing with natural languages paid much efforts on the problem, how to translate natural language input into the internal knowledge structure such as micro PLANNER statements[14], semantic networks[6], frames[1], etc. Most of these systems directly translate input sentences into task oriented internal structure. The architecture of these systems will be much simplified if systematic meaning representation and analysis method based on a formal theory is incorpolated.

This paper proposes a stepwise translation system based on Montague Grammar (MG for short)[3]. Partitioned semantic network[6] is employed as a meaning representatin. Input sentence is firstly translated into logical expression and then semantic network is generated by interpreting it. Semantic network is the output of the natural language analyzer. This will be further compiled into task oriented representations to be used by a task oriented problem solver. This paper concentrates on the natural language analyzer. The following is a summary of our approach:

NATURAL LANGUAGE ANALYZER



where NL: natural language,
LE: logical expression,
PSN: partitioned semantic network,
T: translation mapping,
I: interpretation mapping.

We have developed natural language analyzers for English and Japanese respectively. This paper describes the one for English. The experiments are in progress with these systems. The applicability of the proposed approach is discussed briefly.

2. Overview of the approach

This section gives the readers an overview of the system by illustrating an example. Before illustration we shall present the formalisms of LE and PSN.

LE -- logical expression

The notion of LE is based on Cresswell's λ-categorial language[2]. The following is the syntax of LE:

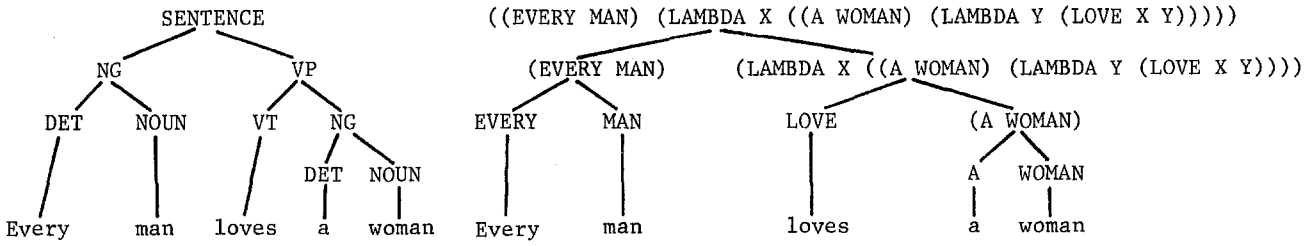
- the set of syntactic categories (Syn): two basic categories are used, i.e., 0 of sentence and 1 of name. Given categories $\tau,\sigma_1, \ldots ,\sigma_n$, then $\langle\tau,\sigma_1, \ldots ,\sigma_n\rangle$ is the category of a mapping that makes an expression of category τ out of expressions of category $\sigma_1, \ldots ,\sigma_n$, respectively.
- the set of symbols (F): $F=\bigcup_{\sigma\in Syn} F\sigma$, where $F\sigma$ is a finite set of symbols, and if $\sigma_1\neq\sigma_2$ then $F\sigma_1\cap F\sigma_2=\phi$.
- the set of variables (X): $X=\bigcup_{\sigma\in Syn} X\sigma$, where $X\sigma$ is a set of variables such that if $\sigma_1\neq\sigma_2$ then $X\sigma_1\cap X\sigma_2=\phi$, and that intersection of F and X is empty.
- the set of expressions (E): $E=\bigcup_{\sigma\in Syn} E\sigma$, where $E\sigma$ fills the following properties:

  (i) $X\sigma\subseteq E\sigma$,
  (ii) $F\sigma\subseteq E\sigma$,
  (iii) if $\delta\in E\langle\tau,\sigma_1, \ldots ,\sigma_n\rangle$ and $\alpha_1, \ldots ,\alpha_n\in E\sigma_1, \ldots ,E\sigma_n$, then the expression $\delta(\alpha_1, \ldots ,\alpha_n)\in E\tau$,
  (iv) if $\beta\in X\sigma$ and $\alpha\in E\tau$, then the expression $\lambda\beta[\alpha]\in E\langle\tau,\sigma\rangle$, where λ is a distinguished symbol in E.

PSN -- partitioned semantic network

PSN denotes the semantics of LE. The notion of network partitioning is based on Hendrix's K-net [6]. The constituents of PSN are:

- a space which denotes a possible world
- a typed nodes
- an arc with a case label



(a) the syntax tree. (b) the semantic tree.

Fig. 3. Syntax analysis and generation of LE for the sentence, "Every man loves a woman."

Sometimes linear notations are used instead of PSN structure. The linear notation can be further interpreted by meta language [8], however this is beyond the scope of this paper. Fig. 1 illustrates PSN structures together with linear notations.

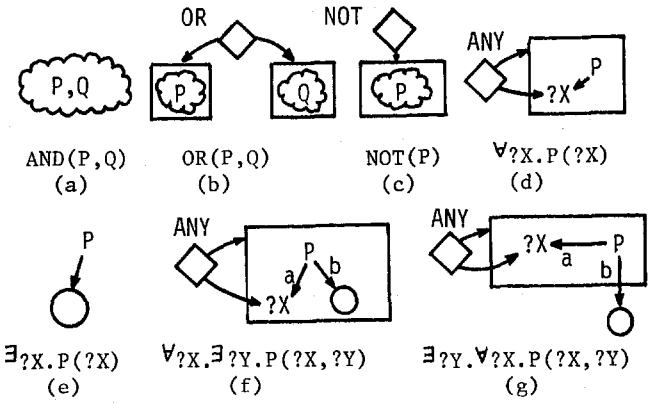


Fig. 1. Basic PSN structures, (with linear notations).

Some special structures are used to denote intensional entities. See fig. 2 below.

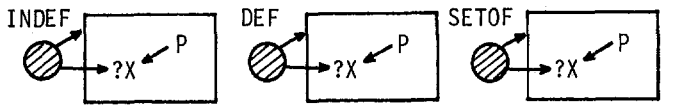


Fig. 2. Intensional structures in PSN.

Overview of the meaning analysis process

Now we illustrate an example. Consider the following simple sentence:

(EX-1) Every man loves a woman.

(STEP 1) Morphological analysis

Input sentence is analyzed by consulting dictionary in which LE expression and grammatical category are assigned to each word. For the given sentence (EX-1), we obtain:

```
Every: cat=DET, sem=EVERY
man: cat=NOUN, sem=MAN
loves: cat=VT, sem=LOVE, (form=+S)
a: cat=DET, sem=A
woman: cat=NOUN, sem=WOMAN
```

(STEP 2) Syntax analysis and generation of LE

Morphologically analyzed sentence is further analyzed by a set of grammar rules. Each grammar rule consists of syntactic generation part and semantic composition part. For the sake of illustration, let the grammar rules be:

```
SENTENCE → NG+VP,
   sem=list[sem[NG];sem[VP]]
NG → DET+NOUN,
   sem=list[sem[DET];sem[NOUN]]
VP → VT+NG,
   sem=list[LAMBDA;?X;
            list[sem[NG];list[LAMBDA;?Y;
               list[sem[VT];?X;?Y]]]]
```

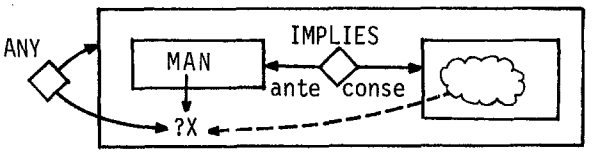
The syntax analysis and semantic composition are done in parallel. If one of them detects anomaly, the application of the rule is aborted. Fig. 3 illustrates the result of the syntax analysis and semantic composition for our example. The syntax tree shows the phrase structure of the sentence. The semantic tree shows the history of semantic composition. The root node of the semantic tree is the LE (in LISP notation) obtained from the sentence.

(STEP 3) Interpretation of LE

Extracted LE is evaluated by a network generation procedure as follows:

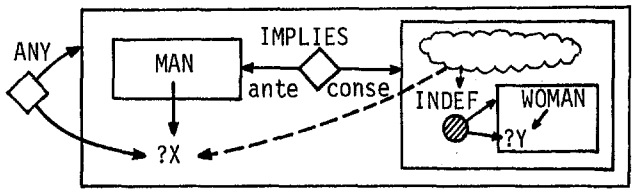
(1) interpreting (EVERY MAN)

A fragment of a universal quantification is generated.



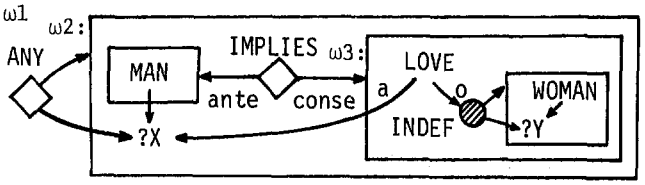
(2) intepreting (A WOMAN)

An intensional node is generated.



(3) interpretating LOVE

A relational node for the two place predicate LOVE is generated.

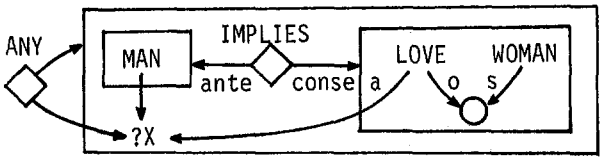


(4) to replace the OBJECT slot of LOVE by its extension

Since the verb "love" is an extensional verb, the OBJECT slot is extensioned, i.e., is replaced by an existentially quantified variable. In our system new individual node is generated in the sense of Skolem constant. We treat scope ambiguity at this time; if this Skolemization is to be done in a local world, scope ambiguity is announced. In this case three ambiguities are detected as for where the Skolemized node is placed, i.e.,

(i) the innermost space ω3:
$\forall ?X.[MAN(?X) \rightarrow \exists ?Y[LOVE(?X,?Y)]]$,
(ii) the middle space ω2:
$\forall ?X.\exists ?Y[MAN(?X) \rightarrow ?Y[LOVE(?X,?Y)]$,
(iii) the outermost space ω1:
$\exists ?Y.\forall ?X[MAN(?X) \rightarrow LOVE(?X,?Y)]$.

Since the reading (i) and (ii) are logically equivalent, there are essentially two ambiguities. If the reading (i) is selected, the final network structure is:



Comments on scope ambiguity

One of the interesting feature of MG is the treatment of scope ambiguity of quantification. In MG, scope ambiguities are captured as ambiguities of semantic composition. However, considering the following two points:

- how to filter out redundancies; sometimes this redundancy is reduced in the interpretation process,
- the resulting parsing sometimes involves inaccurate readings,

it is plausible to treat scope ambiguities in the interpretation process as shown in the above example.

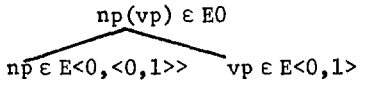
3. Implementation of translation mapping (T)

This section treats the translation mapping T. Firstly, we show how we associate LE with each phrase of English. Then we describe the rule based parser.

The association of LE with English phrases

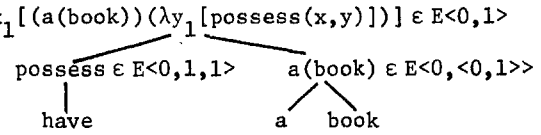
(1) Simple sentence

Simple sentence is composed of a subject and a verb phrase. Subject is a noun phrase. The LE for a noun phrase is in category <0,<0,1>>. The LE for a verb phrase is in category <0,1>. The LE for a sentence is a functional composition of an NG and a VP. See the following illustration:



(2) Verb phrase

Basic part of a verb phrase is composed of either an intransitive verb or a transitive verb plus an object. For example, The LE for a phrase "have a book" is obtained as follows:

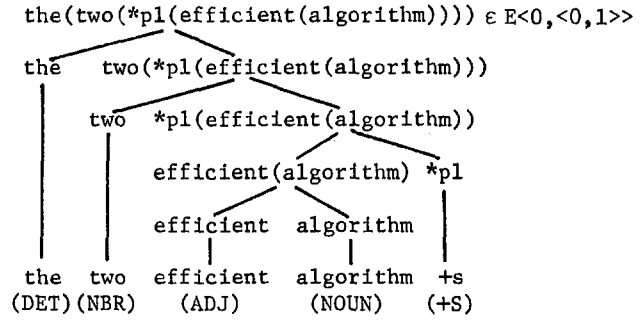


(3) Noun phrase

A noun phrase maps a one-place predicate into a sentence, that is, in category <0,<0,1>>. The constituents of a noun phrase are:

- determiner (DET) in E<<0,<0,1>>,<0,1>>
- number (NBR) in E<<0,1>,<0,1>>
- adjective (ADJ) in E<<0,1>,<0,1>>
- head noun (NOUN) in E<0,1>
- plural morpheme (+S) in E<<0,1>,<0,1>>
- post modifier (Q) in E<<0,1>,<0,1>>

(example) "the two efficient algorithms"



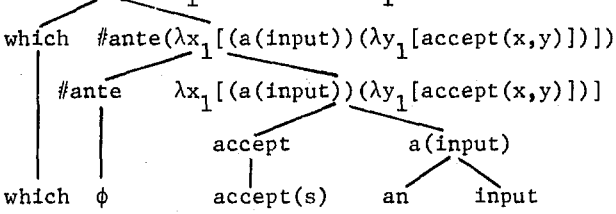
(4) Postmodifier

(i) Relative clause

A relative clause is composed of the symbol 'which' and a sentence. 'Which' makes a postmodifier of a noun out of the sentence. A special symbol #ante (in E<0,<0,1>>) is supplied for the eliminated antecedent in the relative clause. See the following example:

ε E<<0,1>,<0,1>>

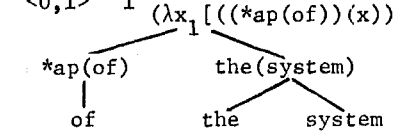
$which(\#ante(\lambda x_1[(a(input))(\lambda y_1[accept(x,y)])]))$



(ii) Adjective prepositional phrase

The LE for a preposition is in category <<0,0>,1>, that is, makes an adverb out of a name. The LE for an adjective prepositional phrase is constructed using a special symbol *ap ε E<<<<0,1>,<0,1>>,1>,<<0,0>,1>>. Roughly speaking *ap converts an preposition into "an adjective preposition" which makes an adjective phrase out of a name. See the following example:

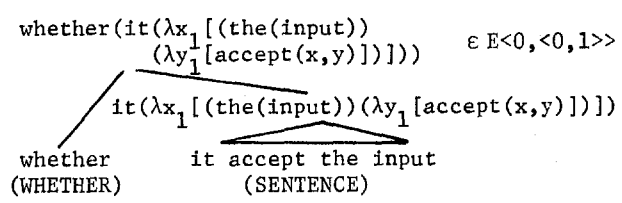
$\lambda p_{<0,1>}[\lambda y_1[(the(system))(\lambda x_1[((*ap(of))(x))(p(y))])]]$ ε E<<0,1>,<0,1>>



(5) Noun clause

A noun clause is constructed from a key word (e.g., "that", "whether", etc) and a complement sentence. The LE for the keyword maps a sentence into a noun phrase, that is, in category <<0,<0,1>>,0>. See the following example:

$whether(it(\lambda x_1[(the(input))(\lambda y_1[accept(x,y)])]))$ ε E<0,<0,1>>



(6) Mood

We treat different mood of a sentence uniformly.

- Declarative is default.
- Interrogative is denoted with a symbol #QUES. #QUES may be paraphrased as:

$\lambda p[I(\lambda x_1[(you(\lambda y_1[p(\lambda z_1[ask(x,y,z)])]))])]$
"I ask you ..." ε E<0,<0,<0,1>>>

Indirect questions and direct questions are treated uniformly. For YES-NO questions, a symbol (whether) is used which maps a sentence into a noun clause. For example,

"Does he run?" $\overset{T}{\Longrightarrow}$ #QUES(whether(he(run))) ε EO

For WH-questions, see the following example:

"Who runs?" $\overset{T}{\Longrightarrow}$ #QUES(who(#ante(run))) ε EO

where, the symbol (who) maps a sentence into a noun clause.

- Imperative is denoted with a symbol #IMP. #IMP may be paraphrased as:

$\lambda p[I(\lambda x_1[(you(\lambda y_1[p(\lambda z_1[order(x,y,z)])]))])]$
"I order you ... " ε E<0,<0,<0,1>>>

For example,

"Take it." $\overset{T}{\Rightarrow}$ #IMP(inf(λx[it(λy[take(x,y)])]))
ε EO

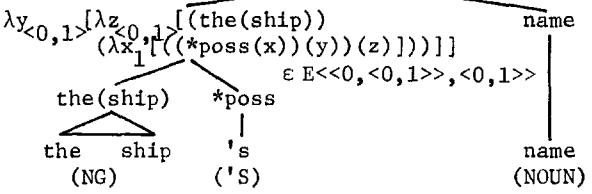
(7) Other features

Some other features are shown by example.

(i) possesive

ε E<0,<0,1>>

$\lambda z_{<0,1>}[(the(ship))(\lambda x_1[((*poss(x))(name))(z)])]$

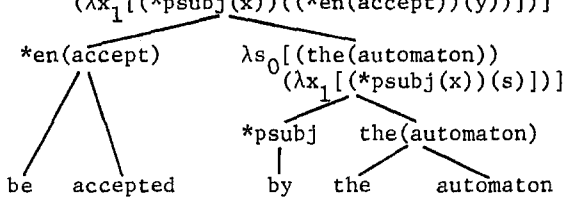


where *poss ε E<<<0,<0,1>>,<0,1>>,1>.

(ii) Passive

ε E<0,1>

$\lambda y_1[(the(automaton))(\lambda x_1[(*psubj(x))((*en(accept))(y))])]$



where *en ε E<<0,1>,<0,1,1>>,
*psubj ε E<<0,0>,1>.

A rule based parser

This section describes a computer program which analyzes input sentence and translates it into LE. The set of rules defined in this section so far are given to the parser in the following format:

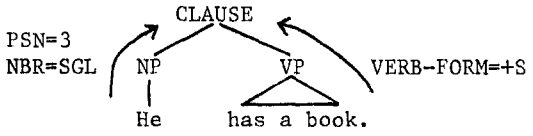
$$A \xrightarrow[\text{<sem>}]{\text{<advice>,<score>}} \alpha$$

where α is a sequence of nonterminals or nonterminals with holes.

The <advice> section treats syntax augmentation of a rule by means of message passing and testing mechanism. A program is embedded which tests whether the messages received from descendants are consistent and which may also send messages to its parent. These messages convey syntactic information about number, person, case, verb form, étc. The format of a message is:

(... <attribute$_i$>=<value$_i$> ...)

For example, see the following illustration:



The <sem> section is a semantic composition program which will construct LE for the node from decendant nodes. In implementing programs, the use of semantic markers is effective. A semantic marker conveys some auxiliary information approximately describing semantic constraints. The LE and semantic markers for a node are packed into a data structure, called a word frame, and manipulated by <sem> section programs.

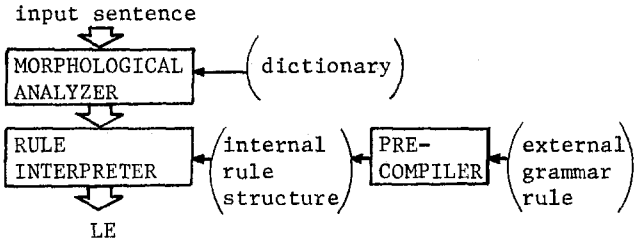
The <score> section determines the priority of the rule. A rule with the highest priority will be tried first.

The grammar system has a feature that allows a user to write elimination rules directly. For example, the following is a rule for a relative clause:

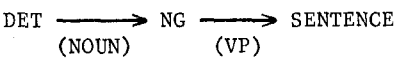
NP → NP+(CLAUSE-NP)

This means that a relative clause is a clause with just one NP eliminated. The semantic coupling of the antecedent and the eliminated noun phrase is described in the <sem> section of the rule.

Now we shall go into the detail of the parser, called EASY (for the English Analysis SYstem). The organization of EASY is summarized in the following diagram:



Before starting parsing, given set of rules are pre-compiled. Nonterminal nodes are connected together and a data structure like ATNG is generated. For example, if we compile the example grammar given in section 2, the following structure (called an expectation path) is generated for the nonterminal DET:



This reads that a DET will grow up to be an NG if a NOUN follows it, and the NG will, in turn, grow up to be a SENTENCE if a VP follows it.

The rule interpreter analyzes input sentence with this compiled rules and a dictionary. EASY is a top-down parser and reads input sentences from the left to the right. EASY starts parsing by expecting the node SENTENCE. The main loop of the rule interpreter is:

- test if the current word has an expectation path to the expected node,
- if the path is found, select the path with the highest priority and save other paths,
- if no path is found, try the following two rules: (1) try a left recursive rule since this type of rule is not compiled in the pre-compile phase, and (2) test if the expected node is eliminated via antecedent elimination rule,
- if both of them fail, memorize the failure and backtrack.

4. Implementation of interpretation mapping (I)

The interpretation mapping I generates a partitioned network structure as a denotation of the meaning of a sentence.

We don't use the truth-conditional formalism. If complete knowledge about the world is given, a computer program can simulate the model to compute the truth value as in [4] or [7]. However in the actual situation of natural language understanding process, complete knowlegde cannot be given, but only partial knowledge is available. Accordingly, it is plausible that new knowledge is acquired from a given sentence in the context of old knowledge structure. For this purpose, Montague's truth conditional approach is indirect and more direct a programming language.

In what follows we try a direct approach. The style of generating networks resembles Scott-Strachey's semantic function [13] which generates a denotation from a statement of programming language.

In order to generate network structure, we use a system which consists of a supervisor function GEN plus dictionary. The arguments of the supervisor are:

(LE, space#, environment, message).

LE is a logical expression. The space# specifies the space in which LE is interpreted. The environment specifies the denotation of each variable by a list of variable-denotation pairs. The message is used for communication between network generating word specialists.

A dictionary entity for each lexicon of LE contains a case pattern or an embedded word

specialist program.

Interpretation of the LE for each category

In what follows we use linear notation of PSN beacuse of the space limitation, and we refer to the LE for each category simply by the category name.

(1) Interpretation of a sentence

The meaning of a simple sentence is governed by the meaning of the verb. A dictionary entity for a verb includes a case pattern for the verb. According to the verb type, the case pattern looks like:

```
intransitive verb: ((SUBJ, EXT, ... )),
extensional transitive verb:
  ((ACTOR, EXT, ... ) (OBJ, EXT, ... )),
intensional transitive verb:
  ((ACTOR, EXT, ... ) (OBJ, INT, ... )),
```

where, the first element of a case slot is a case label which is used only for distinguishing the slot, and the second element of a case slot indicates extensionality of the slot. If the slot indicates extensionality, the filler will be replaced by its extension. This manipulation will be treated later in this section.

(2) Interpretation of a noun phrase

Most significant noun phrase may be in the form, DET+NOUN. The formula is interpreted as follows:

```
(a/an)+noun: λ?P[?P(INDEF[?X; noun*(?X)])],
the+noun: λ?P[?P(DEF[?X; noun*(?X)])],
every+noun: λ?P[ANY[?X; noun*(?X)→?P(?X)]],
no+noun: λ?P[ANY[?X;noun*(?X)→~?P(?X)]],
        where p* means the denotation of p.
```

Personal pronouns is interpreted as follows:

```
I: the SPEAKER attribute,
you: the HEARER attribute,
he: paraphrased as the male,
she: paraphrased as the female.
```

Proper name is interpreted as follows:

```
proper-name:  DEF[?X; NAME('proper-name,?X)].
```

(3) Interpretation of an adjective

An adjective maps a noun into another noun. Here we treat those that plays this role.

Interpretation of plural is:

```
*pl(noun):
   λ?X[SUBSET(?X,SETOF[?Y; noun*(?Y)])]
```

i.e., *pl(noun) denotes a predicate which is true iff the argument is a subset of {x| noun*(x)}.

Adjectives are interpreted by word specialists embedded in the dictionary. A word specialist for an adjective examines the argument (a noun) and maps it into another noun. Thus the word specialist can handle de dicto readings of adjectives. For example,

```
small(lion) ⇒ (I)
  λ?X[LION(?X)&LESS-THAN(DEF[?Y; SIZE(?X,?Y)],
                         average-size-of-lion)].
```

(4) Interpretation of a postmodification

A relative clause (in restrictive use) maps the head noun into a modified noun, as follows:

```
(which(sentence))(noun).
```

A distinguished symbol 'which' announces the occurence of a relative clause and sends the denotation of the antecedent as a message. The argument of 'which' is a sentence including the eliminated noun phrase '#ante' which will receive the message and substitute the denotation. See the following example:

```
the((which(I(λx[#ante(λy[attack(x,y)])])))
      (problem))
   "the problem which I attack"

 Interpreting the formula is:

DEF[?X;
  GEN [[(which(I(λx[#ante(λy[attack(x,y)])]))
        (problem))(z); space#; z:?X; NIL]]]
= DEF[?X;
  GEN [[and(problem(z),
         I(λx[#ante(λy[attack(x,y)])]));
      space#; z:?X; #ante:?X]]]
= DEF[?X; AND(PROBLEM(?X),ATTACK("I",?X))].
```

An adjective prepositional phrase also modifies a noun. An attributive noun or a de-verbal noun is treated as a noun which is a one-place predicate in LE, but which takes two or more arguments in PSN level. Adjective prepositional phrases supply these arguments to the head noun. For example, interpreting the LE:

```
the(λy
 [(the(car))(λx[(((*ap(of))(x))(color))(y)])]),
   "the color of the car",
```

results in:

```
DEF[?Y;
  GEN [[(((*ap(of))(x))(color))(y);
      space#; x:DEF[?X; CAR(?X)], y:?Y; NIL]]]
= DEF[?Y;
   GEN [[color(y); space#;
        y:?Y; *ap:of:DEF[?X; CAR(?X)]]]]
= DEF[?Y; COLOR(DEF[?X; CAR(?X)]; ?Y)].
```

Thus in the interpretation process, the message communications between specialists play a significant role.

(5) Interpretation of a noun clause

A space is used to denote the interpretation of a noun clause. A noun clause is interpreted as follows:

$$\text{fun(sentence)} \overset{I}{\Longrightarrow} \text{DEF[?X; fun*(?X,}\omega 1)\text{]},$$
where T(ω1,sentence*),

where 'fun' stands for a symbol such as 'that', 'whether' ... etc. that maps a sentence into a noun clause. Fun* is an appropriate PSN predicate. T(ω,p) is a meta predicate that means the object formula p is true in the possible world (or space) denoted by ω. For example, interpreting the LE:

```
why(not((the(program))(λx[work(x)]))),
   "why the program does not work"
```

results in:

```
DEF[?X; REASON(?X,ω2)],
where T(ω2,NOT(WORK(the-program*))).
```

In this case, fun=why and fun*=REASON. The resuting denotation roughly reads "the reason of the situation ω2 and in ω2 the object referred to by the expression the(program) does not work."

(6) Interpretation of other features

- Possessive form is treated as a compound determiner. See the following example:

```
λz<0,1>[(the(programmer))
        (λx[((*poss(x))(idea))(z)])],
   "the programmer's idea".
```

The resulting denotation is:

```
DEF[?X; AND(IDEA(?X),
   POSSESS(DEF[?Y; PROGRAMMER(?Y)]; ?X)]
```

- Interpretation of a passive including the deep subject. For example,

```
(the(sentence))
  (λy[(the(automaton))
     (λx[(*psubj(x))((*en(accept))(y))])]),
"the sentence is accepted by the automaton",
```

is interpreted as follows:

```
ACCEPT(DEF[?X; AUTOMATON(?X)],
       DEF[?Y; SENTENCE(?Y)]),
```

where, *psubj sends as a message the denotation of the deep subject, and *en receives the message to supply the OBJECT slot of the internal verb ACCEPT.

Extensioning intensional structures

An intensional PSN structure for a noun phrase is extensioned if the PSN structure is put into a case slot which indicates extensionality.

An INDEF type PSN structure is replaced by an I-unit (which denotes an individual constant). For example,

```
I(λx[(a(book))(λy[possess(x,y)])]),
  "I have a book."
```

The intermediate PSN structure is:

```
POSSESS("I",INDEF[?X; BOOK(?X)]).
```

Since the OBJECT slot of the predicate POSSESS indicates extensionality, this becomes

```
AND(POSSESS("I",C),BOOK(C)),
  where C is a Skolem constant.
```

For DEF type structure, since the denotation refers some uniquely determined object, a referent search program is activated. The program searches local contextual memory by matching each candidate against the given intensional PSN structure. The pattern matching operation in PSN corresponds to deduction on meta language, that is, the defintion of match is:

PSN_1 matches PSN_2
iff meta(PSN_1) implies meta(PSN_2)

In order to find the referent, various kinds of knowledge will be needed [5]. However, this topic is beyond the scope of this paper.

The intensional PSN structure is replaced by a PSN structure found. For example, consider the following two sentences:

```
This paper describes a system.   ... (1)
The system analyzes programs.    ... (2)
```

After the interpretation of the sentence (1), the local memory contains:

```
DESCRIBE(A,B)&PAPER(A)&SYSTEM(B).
```

For the sentence (2), the intermediate structure is:

```
ANALYZE(DEF[?X; SYSTEM(?X)],programs*).
```

After the referent search procedure, the structure becomes:

```
ANALYZE(B,programs*).
```

Since the denotation DEF[?X; SYSTEM(?X)] matches the node B (for, SYSTEM(B) holds), it is replaced by the node B.

5. Discussion

All the mechanisms presented so far has been implemented as LISP programs and are working on the personal LISP system in our laboratory. Now experiments and improvements are in progress.

As stated in the first section, advantages of our method can be shown if it is applied to wide applications. Experiments are in progress as for machine translation and question answering.

Machine translation [12]

As the first step to the machine translation, we are implementing a program which generates Japanese from the LE obtained by analyzing English. The generator program evaluates LE just the same way as the interpretation program does. This approach investigates the linguistic phenomena in analyzing and generating natural language.

Question answering [9], [10]

Another application is to answer questions about the integrated network structure. In order to make conversation with a user, the input sentence should be further evaluated. For example, for user's question actual question/answering process must be invoked. Thus a pattern directed procedure is used. This approach investigates meaning representation and deduction.

Extension to other languages [11]

The meaning representation is, in principle, independent of which language is used. To show this, we must analyze more than one languages. Although in this paper, the object language is English, we have implemented a Japanese parser and are in the course of implementation of Japanese to English machine translation program.

Further work

The important problems to be solved are:

- the problem of discourse, especially, how to treat focus attention or ellipsis in our formalism,
- the semantics of PSN; the semantics of PSN may be defined either by associating each network structure with a logic-oriented meta language or by defining inference rules on PSN explicitly; the semantics must explicate implications and synonyms among PSN structures; furthermore the semantics must be extended to treat the concepts such as action or event,
- accommodation of transformational aspects; it seems that the transformational theory further decomposes the translation mapping T; the introduction of transformational aspect will increase the feasibility of the system.

6. Conclusion

We have shown a logico-linguistic approach to the analysis of natural language by computer. AI techniques are combined with Montague-type grammar. The main features of the approach are shown for the fundamental subset of English. The promising applications may be semantic based machine translation and deductive question answering on natural language.

Acknowledgements

We would like to thank the other members of Prof. Doshita's laboratory, and in particular, Mr. Masaki KIYONO both for his participations of numerous discussions and for his assistance.

References

[1] D. G. Bobrow and T. Winograd, An overview of KRL, A knowledge representation language, CSL-76-4, Xerox, Palo Alto Reseasrch Center, 1976.

[2] M. J. Cresswell, Logics and languages, Methuen & Co. LTd, 1973. (translated into Japanese by Ishimoto and Ikeya, Kinokuniya, 1978).

[3] D. R. Dowty, A guide to Montague's PTQ, Indiana University Linguistic Club, 1978.

[4] J. Friedman, D. B., Moran and D. S. Warren, Evaluating English sentences with a logical model, Information Abstracts of COLING 78, University of Bergen, Norway.

[5] B. J. Grosz, The representation and use of focus in a system for understanding dialogs, in Proc. IJCAI-77, 1977, 67-76.

[6] R. Fikes and G. Hendrix, A network-based knowledge representation and its natural deduction system, in Proc. IJCAI-77, 1977, 235-246.

[7] J. R. Hobbs, Making computational sense of Montague's intensional logic, AI 9(1978), 287-306.

[8] R. C. Moore, Reasoning about knowledge and action, in Proc. IJCAI-77, 1977, 223-227.

[9] T. Nishida and S. Doshita, The framework of knowledge representation and its retrieval in LGS -- the literature guide system, in Proc. IJCAI-79, 1977, 662-664.

[10] T. Nishida and S. Doshita, A knowledge-based literature guide system -- A new approach to document retrieval, IFIP 80 (to appear).

[11] T. Nishida, Y. Sakakibara and S. Doshita, Analysis of predicates of Japanese and the generation of English, National Conventional Record of IPS Japan, 1980, (in Japanese).

[12] T. Nishida, M. Kiyono, T. Yamanaka and S. Doshita, English-Japanese translation based on semantic analysis, National Conventional Record of IPS Japan, (in Japanese).

[13] J. E. Stoy, Denotational semantics: The Scott-Strachey approach to programming language theory, The MIT press, 1977.

[14] T. Winograd, Understanding natural language, Academic Press 1972. (translated into Japanese by Fuchi, Tamura and Shirai, Sangyo-tosho, 1976).