AN APPLICATION OF MONTAGUE GRAMMAR TO ENGLISH-JAPANESE MACHINE TRANSLATION

Toyoaki NISHIDA and Shuji DOSHITA
Dept. of Information Science
Faculty of Engineering, Kyoto University
Sakyo-ku, Kyoto 606, JAPAN

ABSTRACT

English-Japanese machine translation requires a large amount of structural transformations in both grammatical and conceptual level. In order to make its control structure clearer and more understandable, this paper proposes a model based on Montague Gramamr. Translation process is modeled as a data flow computation process. Formal description tools are developed and a prototype system is constructed. Various problems which arise in this modeling and their solutions are described. Results of experiments are shown and it is discussed how far initial goals are achieved.

1. GOAL OF INTERMEDIATE REPRESENTATION DESIGN

Differences between English and Japanese exist not only in grammatical level but also in conceptual level. Examples are illustrated in Fig.1. Accordingly, a large amount of transformations in various levels are required in order to obtain high quality translation. The goal of this research is to provide a good framework for carrying out those operations systematically. The solution depends on the design of intermediate representation (IR). Basic requirements to intermediate representation design are listed below.

a) Accuracy: IR should retain logical conclusion of natural language expression. The following distinctions, for example, should be made in IR level:

- partial/total negation
- any-exist/exist-any
- active/passive
- restrictive use/ nonrestrictive use, etc.

In other words, scope of operators should be represented precisely.

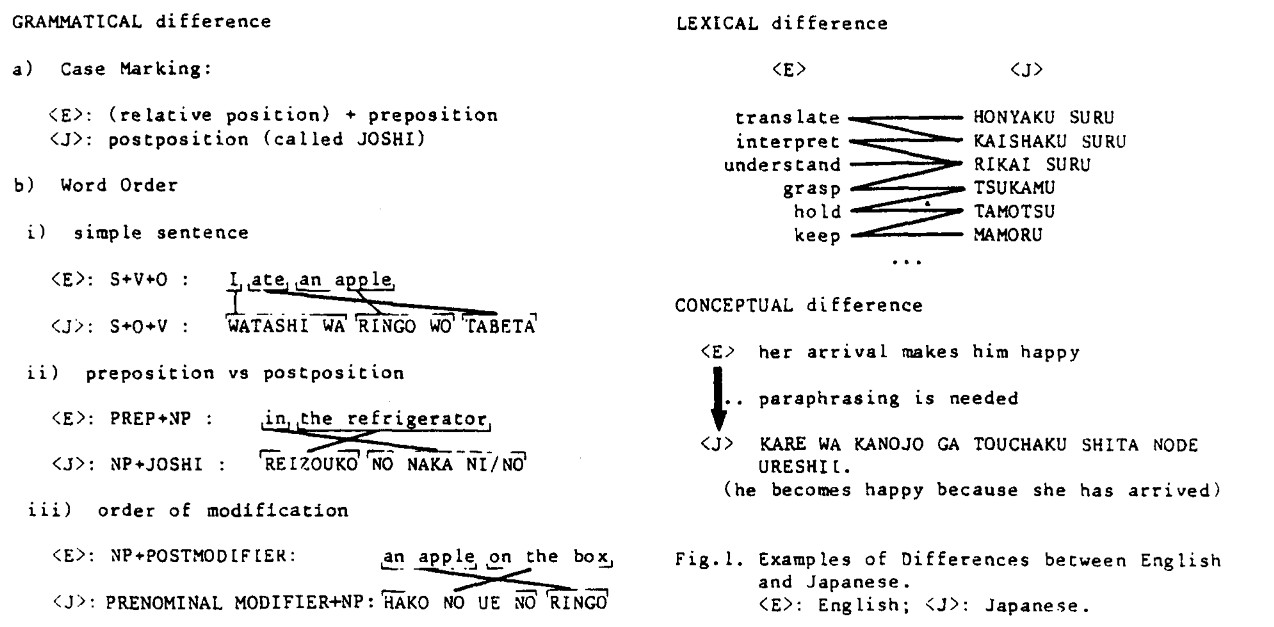

GRAMMATICAL difference
a) Case Marking:
<E>: (relative position) + preposition
<J>: postposition (called JOSHI)
b) Word Order
i) simple sentence
<E>: S+V+O : I ate an apple
<J>: S+O+V : WATASHI WA RINGO WO TABETA
ii) preposition vs postposition
<E>: PREP+NP : in the refrigerator
<J>: NP+JOSHI : REIZOUKO NO NAKA NI/NO
iii) order of modification
<E>: NP+POSTMODIFIER: an apple on the box
<J>: PRENOMINAL MODIFIER+NP: HAKO NO UE NO RINGO
LEXICAL difference
<E>
<J>
translate
interpret
understand
grasp
hold
keep
HONYAKU SURU
KAISHAKU SURU
RIKAI SURU
TSUKAMU
TAMOTSU
MAMORU
...
CONCEPTUAL difference
<E> her arrival makes him happy
.. paraphrasing is needed
<J> KARE WA KANOJO GA TOUCHAKU SHITA NODE URESHII.
(he becomes happy because she has arrived)

Fig.1. Examples of Differences between English and Japanese.
<E>: English; <J>: Japanese.

b) Ability of representing semantic relations: In English-Japanese translation, it is often the case that a given English word must be translated into different Japanese words or phrases if it has more than one word meanings. But it is not reasonable to capture this problem solely as a problem of word meaning disambiguation in analysis phase; the needed depth of disambiguation depends on target language. So it is also handled in transfer phase. In general, meaning of a given word is recognized based on the relation to other constituents in the sentence or text which is semantically related to the given word. To make this possible in transfer phase, IR must provide a link to semantically related constituents of a given item. For example, an object of a verb should be accessible in IR level from the verb, even if the relation is implicit in the surface structure (eg., passives, relative clauses, and their combinations, etc.)

c) Prediction of control: given an IR expression, the model should be able to predict explicitly what operations are to be done in what order.

d) Lexicon driven: some sort of transformation rules are word specific. The IR interpretation system should be designed to deal with those word specific rules easily.

e) Computability: All processings should be effectively computable. Any IR is useless if it is not computable.

2. PRINCIPLE OF TRANSLATION

This section outlines our solution to the requirements posed in the preceding section. We employ Montague Grammar (Montague 1974, Dowty 1981) as a theoretical basis of translation model. Intermediate representation is designed based on intensional logic. Intermediate representation for a given natural language expression is obtained by what we call functional analysis.

2.1 Functional Analysis

In functional analysis, input sentence is decomposed into groups of constituents and interrelationship among those groups are analyzed in terms of function-argument relationships. Suppose a sentence:

I don't have a book. (1)

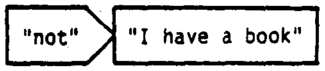
The functional analysis makes following two points:

a) (1) is decomposed as:

"I have a book" + "not". (2)

b) In the decomposition (2), "not" is an operator or function to "I have a book."

The result of this analysis can be depicted as follows:



(3)

where denotes a function and denotes an argument. The role of "not" as a function is:

"not" as a semantic operator:
  it negates a given proposition;
"not" as a syntactic operator:
  it inserts an appropriate auxiliary verb and a lexical item "not" into appropriate position of its argument. (4)

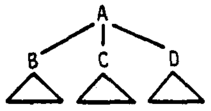
This kind of analysis goes on further with embedded sentence until it is decomposed into lexical units or even morphemes.

2.2 Montague Grammar as a Basic Theory

Montague Grammar (MG) gives a basis of functional analysis. One of the advantages of MG consists in its interpretation system of function form (or intensional logical form). In MG, interpretation of an intensional logical formula is a mapping I from intensional logical formulas to set theoretical domain. Important property is that this mapping I is defined under the constraint of compositionality, that is, I satisfies:

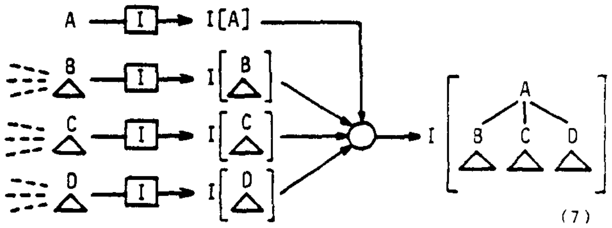
$$I[f(a,b,\ldots)]=I[f](I[a],I[b],\ \ldots),\qquad (5)$$

without regard to what f, a, b, etc. are. This property simplifies control structure and it also specifies what operations are done in what order. For example, suppose input data has a structure like:



(6)

For the sake of property (5), the interpretation of (6) is done as a data flow computation process as follows:



(7)

By this property, we can easily grasp the processing stream. In particular, we can easily shoot trouble and source of abnormality when debugging a system.

Due to the above property and others, in particular due to its rigorous framework based on logic, MG has been studied in information science field (Hobbs 1978, Friedman 1978, Yonezaki 1980, Nishida 1980, Landsbergen 1980, Moran 1982, Moore 1981, Rosenschein 1982, ...). Application of MG to machine translation was also attempted (Hauenschild 1979, Landsbergen 1982), but those systems have only partially utilized the power of MG. Our approach attempts to utilize the full power of MG.

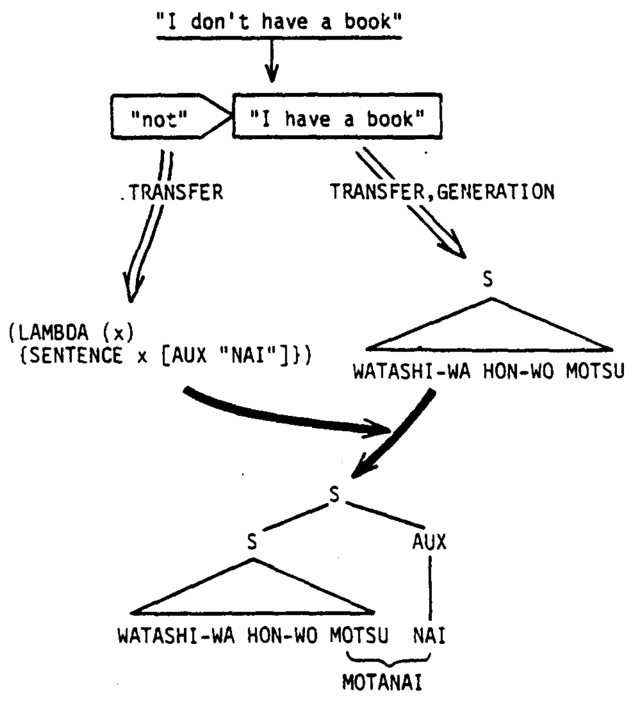
2.3 Application of Montague Grammar to Machine Translation

In order to obtain the syntactic structure in Japanese from an intensional logical form, in the same way as interpretation process of MG, we change the semantic domain from set theoretical domain to conceptual domain for Japanese. Each conceptual unit contains its syntactic expression in Japanese. Syntactic aspect is stressed for generating syntactic structure in Japanese. Conceptual information is utilized for semantic based word choice and paraphrasing.

For example, the following function in Japanese syntactic domain is assigned to a logical item "not":

(LAMBDA (x) (SENTENCE x [AUX "NAI"])). (8)

Transfer-generation process for the sentence (1) looks like:



3. FORMAL TOOLS

Formal description tools have been developed to provide a precise description of the idea mentioned in the last section.

3.1 Definition of Formal Tools

a) English oriented Formal Representation (EFR) is a version of intensional logic, and gives a rigorous formalism for describing the results of functional analysis. It is based on Cresswell's lambda deep structure (Cresswell 1973). Each expression has a uniquely defined type. Lambda form is employed to denote function itself.

b) Conceptual Phrase Structure (CPS) is a data structure in which syntactic and semantic information of a Japanese lexical unit or phrase structure are packed.

i) example of CPS for a lexical item:

```
EIGO:[NP "EIGO" with ISA=LANGUAGE; ... ]   (9)
      |     \            \
   category; lexical item; conceptual info.
```

; "EIGO" means English language.

ii) example of CPS for phrase structure:

```
[NP [ADJ "AKAI" with ... ]
    [NOUN "RINGO" with ... ] with ... ]   (10)
```

; "AKAI" means red, and "RINGO" means apple.

c) CPS Form (CPSF) is a form which denotes operation or function on CPS domain. It is used to give descriptions to mappings from EFR to CPS. Constituents of CPSF are:

i) Constants: CPS.

ii) Variables: x, y, ...
(indicated by lower case strings).

iii) Variables with constraints:
e.g., (! SENTENCE x).
; variable x which must be of category SENTENCE.

iv) Transformations:

```
e.g., (+ TENSE (TENSE=PAST) x).
       /    \        \        \
indicator; operator-name; PARAMs; argument
```

v) CPS construction:

```
e.g., (SENTENCE (x y) with ... ).
        /         \
  new category; descendents
```

vi) Conditionals:
[<condition>$_1$ -> <CPSF>$_1$; ...].

vii) Lambda form:
e.g., (LAMBDA (x) (+ PASSIVE () x))

Using those description tools, translation process is modeled as a three staged process:

stage 1 (analysis): anlyzes English sentence and extracts EFR form,

stage 2 (transfer): substitutes CPSF to each lexical item in the EFR form,

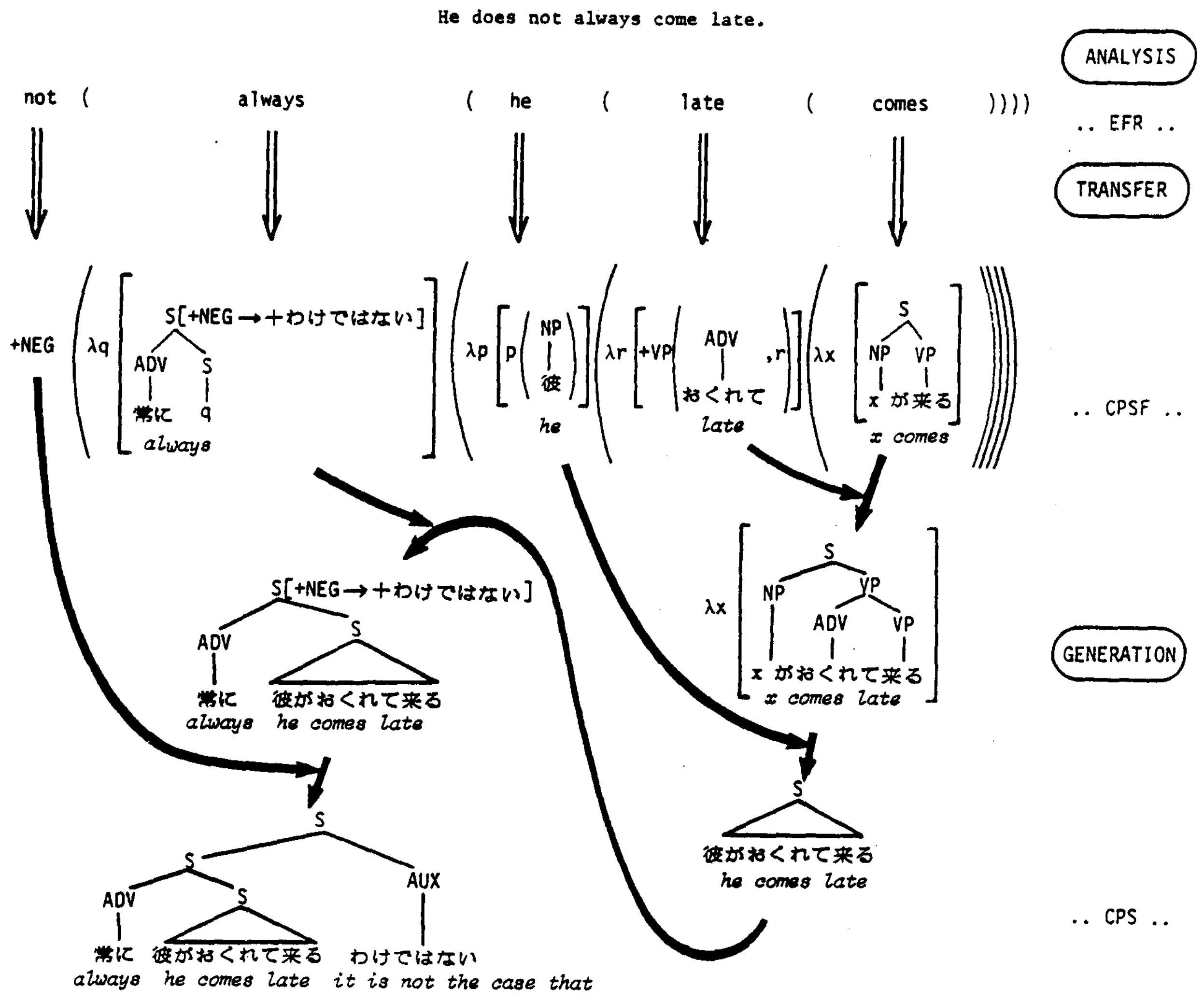


Fig.2. Example of Translation Process described using Formal Tools. / Prefix notation is used for CPSF, and syntactic aspect is emphasized.

stage 3 (generation): evaluates the CPSF to get CPS; generation of surface structure from CPS is straightforward.

In order to give readers an overall perspective, we illustrate an example in Fig.2. Note that the example illustrated includes partial negation. Thus operator "not" is given a wider scope than "always".

In the remaining part of this section we will describe how to extract EFR expression from a given sentence. Then we will discuss the problem which arises in evaluating CPSF, and give its possible solution.

3.2 Extracting EFR Expression from Input Sentence

Rules for translating English into EFR form is associated with each phrase structure rules. For example, the rule looks like:

NP -> DET+NOUN where <NP>=<DET>(<NOUN>) (11)

where, <NP> stands for an EFR form assigned to the NP node, etc. Rule (11) says that EFR for an NP is a form whose function section is EFR for a DET node and whose argument section is EFR for a NOUN node. This rule can be incorporated into conventional natural language parser.

3.3 Evaluation of CPSF

Evaluation process of CPSF is a sequence of lambda conversions and tree transformations. Evaluation of CPSF is done by a LISP interpreter-like algorithm. A problem which we call higher order problem arose in designing the evaluation algorithm.

Higher Order Problem

By higher order property we mean that there exist functions which take other functions as arguments (Henderson 1980). CPSF in fact has this property. For example, an adjective "large" is modeled as a function which takes a noun as its argument. For example,

large(database).
"large database" (12)

On the other hand, adverbs are modeled as functions to adjectives. For example,

very(large), extremely(large),
comparatively(large), etc. (13)

The difficulty with higher order functions consists in modifiction to function. For explanation, let our temporal goal be regeneration of English from EFR. Suppose we assign to "large" a lambda form like:

(LAMBDA (x) (NOUN [ADJ "LARGE"] x)) (14)

which takes a noun and returns a complex noun by attaching an adjective "large". If the adjective is modified by an adverb, say "very", we have to modify (14); we have to transform (14) into a lambda form like:

```
(LAMBDA (x)
     (NOUN [ADJ [ADV "VERY"]
                [ADJ "LARGE"]] x)),      (15)
```

which attaches a complex adjective "very large" to a given noun. As is easily expected, it is too tedious or even impossible to do this task in general. Accordingly, we take an alternative assignment instead of (14), namely:

large <= [ADJ "LARGE"]. (16)

Since this decision cuases a form:

[ADJ "LARGE"]([NOUN "DATABASE"]), (17)

to be created in the course of evaluation, we specify what to do in such case. The rule is defiend as follows:

[ADJ]([NOUN]) = [NOUN [ADJ] [NOUN]]. (18)

This rule is called an application rule.

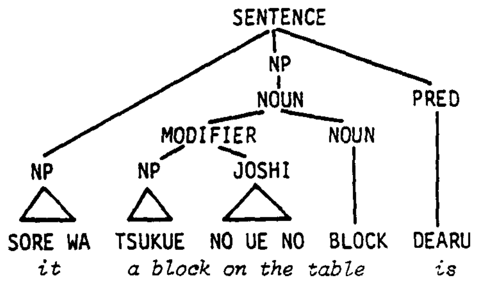
In general, evaluation of lambda form itself results in a function value (function as a value). This causes difficulty as mentioned above. Unfortunately, we can't dispense with lambda forms; lambda variables are needed to link gap and its antecedent in relative clause, verb and its dependants (subject, object, etc), preposition and its object, etc. For example, in our model, an complex noun modified by a PP:

"block on the table" (19)

is assigned a following EFR:

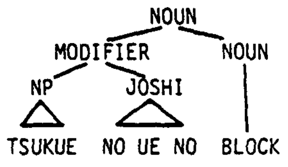
λy[(the(table))
  (λx[(((*ap(on))(y))(block))(x)])], (20)
; which may read: is y:[there is a uniquely specified object y referred to by an NP "the table", such that y is a block which is restricted to be located on x.]

This lambda form is too complicated for tree transformation procedure to manipulate. So it should be transformed into equivalent CPS if it exists. The type of the lambda form is known from the context, namely one-place predicate. So if we apply the lambda form (20) to "known" entity, say "it", we can obtain sentence structure like:



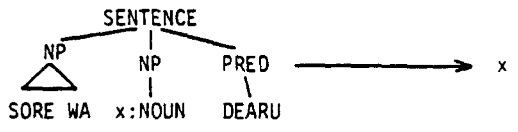
(it is a block on the table) (21)

From this result, we can infer that the lambda form (20) is equivalent to a noun:



(block on the table) (22)

The extraction rule can be written as a pattern matching rule like:



(It is an x) (23)

Of course, this way of processing is not desirable; it introduces extra complexity. But this is a trade off of employing formal semantics; the same sort of processing is also done rather opaque procedures in conventional MT system.

4. MODELING TRANSLATION PROCESS

This section illustrates how English-Japanese translation process is modeled using formal tools. Firstly, how several basic linguistic constructions are treated is described and then mechanism for word choice is presented.

4.1 Translating Basic Constructions of English

a) Sentence: sentence consists of an NP and a VP. VP is analyzed as a one-place predicate, which constructs a proposition out of an individual referred to by the subject. VP is further decomposed into intransitive verb or transitive verb + object. Intransitive verbs and transitive verbs are analyzed as one-place predicates and two-place predicate, respectively. One-place predicate and two-place predicate are assigned a CPSF function which generates a sentence out of an individual and that which generates a sentence out of a pair of individuals, respectively. Thus, a transitive verb "constructs" is assigned a CPSF form:

```
(LAMBDA (x y)
  (SENTENCE
    (+ CASE-MARKER (CASE=AGENT) x)
    (+ CASE-MARKER (CASE=OBJ) y)
    [PREDICATE [VERB "SAKUSEI-SURU"]])),  (24)
```

; given two individuals, this function attaches to each argument a case marker (corresponding to JOSHI or Japanese postfix) and then generates a sentence structure.

The assignment (24) may be extended later to incorporate word choice mechanism.

Treatment of NP in Montague-based semantics is significant in that EFR expression for an NP is given a wider scope than that for a VP. Thus the EFR form for an NP-VP construction looks like:

```
<NP>(<VP>),                           (25)
```

where <x> means EFR form for x, x=NP,... . The reason is to provide an appropriate model for English quantifier which is syntactically local but semantically global. For example, first order logical form for a sentence:

"this command needs no operand"   (26)

looks like:

```
not(there-exists x
       [needs("this-command",x) &
        operand(x)]),                  (27)
```

where operator "not", which comes from a determiner "no", is given a wider scope than "needs". This translation is straightforward in our model; the following EFR is extracted from (26):

```
(this(command))
   λx[(no(operand))(λy[needs(x,y)])]).    (28)
```

If we make appropriate assignment including:

```
no <= (LAMBDA (p)
        (LAMBDA (q)
          "not(there exists x
                 [p(x) & q(x)])")),       (29)
```

we can get (27) from (28).

In English-Japanese machine translation, this treatment gives an elegant solution to the translation of prenominal negation, partial negation, etc. Since Japanese language does not have a syntactic device for prenominal negation, "no" must be translated into mainly two separate constituents: one is a RENTAISHI (Japanese determiner) and another is an auxiliary verb of negation. One possible assignment of CPSF looks like:

```
no <= (LAMBDA (p)
        (LAMBDA (q)
          (+ NEG ()
            (q (NP "DONNA" (! NOUN p) "MO"))))).
                                          (30)
```

In general, correspondence of NP and individual is indirect in EFR. The association of an NP with its referent x is indicated as follows:

$$\langle NP\rangle(\underbrace{\lambda x[\underbrace{\ldots\ x\ \ldots}_{\text{sentence type}}]}_{\text{one-place predicate type}}). \qquad (31)$$

; <NP> stands for EFR expression for NP.

Most of other NP's correspond to its referent more directly. The application rule reflecting this fact is:

```
[NP]([ONE-PLACE-PRED]) = [ONE-PLACE-PRED]([NP*]),
                                          (32)
```

where, [x] stands for a CPS for x.

b) Internal structure of NP: the below illustrates the structure of EFR expression assigned to an NP:

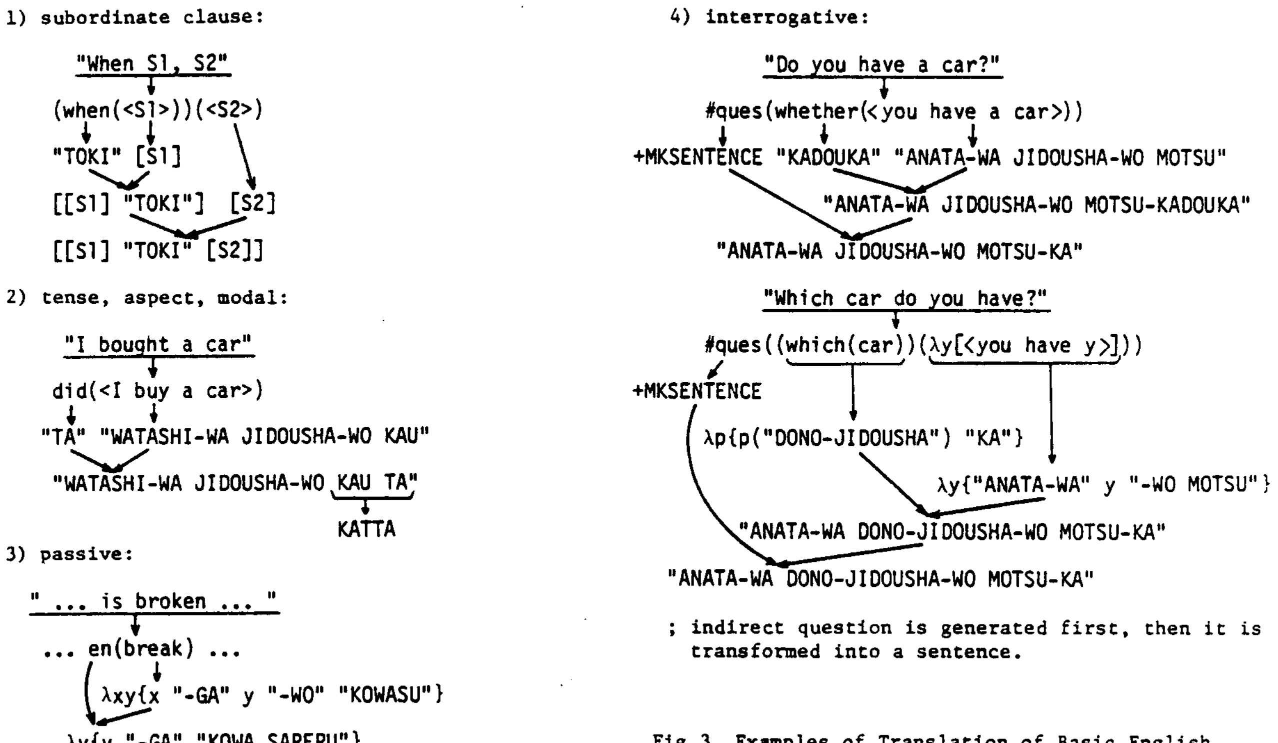
```
<DET>(<MODIFIER>(...(<MODIFIER>(<NOUN>)) ...)).
                                          (33)
```

By <MODIFIER> we mean modification to noun by adjectives, prepositional phrases, infinitives, present/past particles, etc. The translation process is determined by a CPSF assigned to <DET>. In cases of "the" or "a/an", translation process is a bit complicated. It is almost the same as the process described in detail in section 3: firstly the <MODIFIER>s and <NOUN> are applied to an individual like "the thing" (the) or "something" (a/an) and a sentence will be obtained; then a noun structure is extracted and appropriate RENTAISHI or Japanese determiner is attached.

c) Other cases: some other cases are illustrated by examples in Fig.3.

4.2 Word Choice Mechanism

In order to obtain high quality translation, word choice mechanism must be incorporated at least for handling the cases like:


1) subordinate clause:
"When S1, S2"
(when(<S1>))(<S2>)
"TOKI" [S1]
[[S1] "TOKI"] [S2]
[[S1] "TOKI" [S2]]
2) tense, aspect, modal:
"I bought a car"
did(<I buy a car>)
"TA" "WATASHI-WA JIDOUSHA-WO KAU"
"WATASHI-WA JIDOUSHA-WO KAU TA"
KATTA
3) passive:
" ... is broken ... "
... en(break) ...
λxy{x "-GA" y "-WO" "KOWASU"}
λy{y "-GA" "KOWA SARERU"}
4) interrogative:
"Do you have a car?"
#ques(whether(<you have a car>))
+MKSENTENCE "KADOUKA" "ANATA-WA JIDOUSHA-WO MOTSU"
"ANATA-WA JIDOUSHA-WO MOTSU-KADOUKA"
"ANATA-WA JIDOUSHA-WO MOTSU-KA"
"Which car do you have?"
#ques((which(car))(λy[<you have y>]))
+MKSENTENCE
λp{p("DONO-JIDOUSHA") "KA"}
λy{"ANATA-WA" y "-WO MOTSU"}
"ANATA-WA DONO-JIDOUSHA-WO MOTSU-KA"
"ANATA-WA DONO-JIDOUSHA-WO MOTSU-KA"

; function "en" transforms a CPSF for a transitive verb into intransitive.

; indirect question is generated first, then it is transformed into a sentence.

Fig.3. Examples of Translation of Basic English Construction. <x>, {x}, [x] and "x" stand for EFR for x, CPSF for x, CPS for x, and CPS for Japanese string x, respectively.

verb in accordance with its object or its agent,
adjective-noun,
adverb-verb, and
preposition.

Word choice is partially solved in the analysis phase as a word meaning disambiguation. So the design problem is to determine to what degree word sense is disambiguated in the analysis phase and what kind of ambiguities is left until transfer-generation phase. Suppose we are to translate a given preposition. The occurence of a preposition is classified as:

(a) when it is governed by verbs or nouns:
  (a-1) when government is strong:
    e.g., study on, belong to, provide for;
  (a-2) when government is weak:
    e.g., buy ... at store;
(b) otherwise:
  (b-I) idiomatic:
    e.g., in particular, in addition;
  (b-2) related to its object:
    e.g., by bus, with high probability, without+ING.

We treat (a) and (b-1) as an analysis problem and handle them in the analysis phase. (b-2) is more difficult and is treated in the transfer-generation phase where partial semantic interpretation is done.

Word choice in transfer-generation phase is done by using conditional expression and attributive information included in CPS. For example, a transitive verb "develop" is translated differently according to its object:

$$\text{develop} \begin{cases} (+\ \text{system}) \ldots \text{KAIHATSU-SURU} \\ (+\ \text{film}) \ \ldots \text{GENZOU-SURU.} \end{cases} \qquad (34)$$

The following assignment of CPSF makes this choice possible:

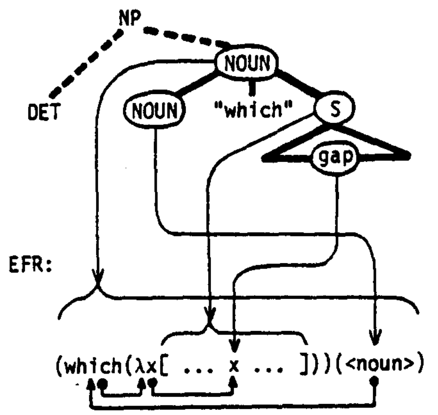
```
develop
  <= (LAMBDA (x y)
       [(CLASS y)=SYSTEM ->
          {"x-GA y-WO KAIHATSU-SURU"};
        (CLASS y)=FILM  ->
          {"x-GA y-WO GENZOU-SURU"};
                                  ... ]),    (35)

operating-system
  <= [NOUN "OS" with CLASS=system; ... ], (36)

film
  <= [NOUN "FUILUMU" with CLASS=film; ... ].
                                             (37)
```

To make this type of processing possible in the cases where the deep object is moved from surface object position by transformations, link information between verb and its (deep) object should be represented explicitly. The below shows how it is done in the case of relative clause.

Phrase Structure (for restrictive use):



link from head noun to place holder

CPSF assignment:

```
which ⇒ (LAMBDA (P) (LAMBDA (Q)
          (NOUN (+ MK-MODIFIER ()
                   (P (+ MK-NULL-NP () Q)))
          Q))),
```

; In EFR level, lambda variable x is explicitly used as a place holder for the gap. A functor "which" dominates both the EFR for the embedded sentence and that for the head noun. A CPSF assigned to the functor "which" sends conceptual information of the head noun to the gap as follows: firstly it creates a null NP out of the head noun, then the null NP is substituted into the lambda variable for the gap.

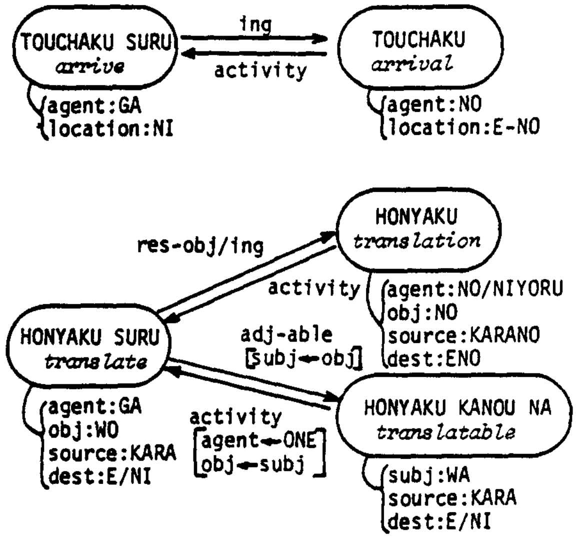
In word choice or semantic based translation in general, various kinds of transformations are carried out on target language structure. For example,

her arrival makes him happy, (38)

must be paraphrased into:

he becomes happy because she has arrived (39)

since inanimate agent is unnatural in Japanese. In order to retrieve appropriate lexical item of target language for transformation, mutual relations among lexical items are organized using network formalism (lexical net). The node represents a lexical item and a link represents an association with specification of what operation causes that link to be passed through. It also contains description of case transformation needed in order to map case structure appropriately. The below illustrate a part of lexical net:



5. EXPERIMENTS

We have constructed a prototype system. It is simplified than practical system in:

- it has only limited vocabulary,
- interactive disambiguation is done instead of automatic disambiguation, and
- word choice mechanism is limited to typical cases since overall definition of rules have not yet been completed.

Sample texts are taken from real computer manuals or abstracts of computer journals. Initially, four sample texts (40 sentences) are chosen. Currently it is extended to 10 texts (72 sentences).

Additional features are introduced in order to make the system more practical.

a) Parser: declarative rules are inefficient for dealing with sentences in real texts. The parser uses production type rules each of which is classified according to its invocation condition. Declarative rules are manually converted into this rule type.

b) Automatic posteditor: transfer process defined so far concentrates on local processings. Even if certain kinds of ambiguities are resolved in this phase, there still remains a possibility that new ambiguity is introduced in generation phase. Instead of incorporating into the transfer-generation phase a sophisticated mechanism for filtering out ambiguities, we attach a postprocessor which will "reform" a phrase structure yielding ambiguous output. Tree-tree transformation rules are utilized here.

Current result of our machine translation system is shown in Appendix.

6. DISCUSSION

Having completed initial experiments, it is shown that our framework is applicable to real texts under plausible assumption. The prototype system has a clear architecture. Central rule interpreter contains no complicated parts. Although several errors occured in the implementation of translation rules, they were easily detected and eliminated for the sake of data flow property.

The initial requirement for intermediate representation are filled in the following way:

Requirement a:	precise representation based on intensional logic,
Requirement b:	using lambda variables and scope rules,
Requirement c:	data flow computing model based on compositionality,
Requirement d:	any CPSF can be assigned to a given lexical item if type is agreed,
Requirement e:	fact that computer model has been implemented.

Some essential problems are left unsolved.

1) Scope analysis: correct analysis of scope of words are crucial but difficult. For example, scope relation of auxiliary and "not" differs case by case:

```
he can't swim
   -> not(can(<he>,<swim>))                (40)
you should not eat the banana
   -> should(not(<eat the banana>))        (41)

it may not be him
   -> may(not( <it=he> ))                  (42)

you may not eat the banana
   -> not(may( <you eat banana>))          (43)
```

2) Logic vs machine translation: The sentence (44) is logically equivalent to (45), but that paraphrasing is bad in machine translation.

```
he reads and writes English.               (44)
he reads English and he writes English.    (45)
```

7. CONCLUSION

Application of formal semantics to machine translation brings about new phase of machine translation. It makes the translation process clearer than conventional systems. The theory has been tested by implementing a prototype, which can translate real texts with plausible human assist.

REFERENCES

Cresswell, M.J. (1973): Logic and Languages, Methuen and Company.

Dowty, R. et al (1981): Introduction to Montague Semantics, Reidel.

Friedman, J. (1978): Evaluating English Sentences in a Logical Model, Abstract 16, COLING 78.

Hauenschild, C., et al. (1979): SALAT: Machine Translation Via Semantic Representation, Bauerle et al.(eds.): Semantics From Different Points of View, Springer-Verlag, 324-352.

Henderson, P.(1980): Functional Programming -- Application and Implementation, Prentice/Hall.

Hobbs, J.R. and Rosenschein, S.J. (1978): Making Computational Sense of Montague's Intensional Logic, AI 9, 287-306.

Landsbergen, J. (1980): Adaptation of Montague Grammar to the Requirement of Question Answering, Proc. COLING 80, 211-212.

Landsbergen, J. (1982): Machine Translation based on Logically Isomorphic Montague Grammars, Proc. COLING 82.

Montague, R. (1974): Proper Treatment of Quantification in Ordinary English, Thompson (ed.) Formal Philosophy, Yale University, 247-270.

Moore, R.C. (1981): Problems in Logical Form, Proc. 19th Annual Meeting of the ACL, 117-124.

Moran, D.B. (1982): The Representation of Inconsistent Information in a Dynamic Model-Theoretic Semantics, Proc. 20th Annual Meeting of the ACL, 16-18.

Nishida, T. and Doshita, S. (1980): Hierarchical Meaning Representation and Analysis of Natural Language Documents, Proc. COLING 80, 85-92.

Rosenschein, S.J. and Shieber, S.M.(1982): Translating English into Logical Form, Proc. 20th Annual Meeting of the ACL, 1-8.

Yonezaki, H. and Enomoto, H. (1980): Database System Based on Intensional Logic, Proc. COLING 80, 220-227.

APPENDIX: Translation of a Sample Text.

INPUT TEXT

Ethernet is a system for local communication among computing stations. Our experimental Ethernet uses tapped coaxial cables to carry variable-length digital data packets among, for example, personal minicomputers, printing facilities, large file storage devices, magnetic tape backup stations, larger central computers, and longer-haul communication equipment.

The shared communication facility, a branching Ether, is passive. A station's Ethernet interface connects bit-serially through an interface cable to a transceiver which in turn taps into the passing Ether. A packet is broadcast onto the Ether, is heard by all stations, and is copied from the Ether by destinations which select it according to the packet's leading address bits. This is broadcast packet switching and should be distinguished from store-and-forward packet switching in which routing is performed by intermediate processing elements. To handle the demands of growth, an Ethernet can be extended using packet repeaters for signal regeneration, packet filters for traffic localization, and packet gateways for internetwork address extension.

Control is completely distributed among stations with packet transmissions coordinated through statistical arbitration. Transmissions initiated by a station defer to any which may already be in progress. Once started, if interference with other packets is detected, a transmission is aborted and rescheduled by its source station. After a certain period of interference-free transmission, a packet is heard by all stations and will run to completion without interference. Ethernet controllers in colliding stations each generate random retransmission intervals to avoid repeated collisions. The mean of a packet's retransmission intervals is adjusted as a function of collision history to keep Ether utilization near the optimum with changing network load.

Even when transmitted without source-detected interference, a packet may still not reach its destination without error; thus, packets are delivered *only with high probability*. Stations requiring a residual error rate lower than that provided by the bare Ethernet packet transport mechanism must follow mutually agreed upon packet protocols.

(cited from: Metcalfe, R.M. and Boggs, D.R. (1975): Ethernet: Distributed Packet Switching for Local Computer Networks, CSL-75-7, Xerox.)

OUTPUT TEXT

translation is carried out sentence by sentence; the result is assembled by hand.

ＥＴＨＥＲＮＥＴは計算ステーションの間の局所的な通信のためのシステムである。我々の実験的なＥＴＨＥＲＮＥＴは可変長のデジタルデータのパケットを例えば、個人のミニコンピュータ、印刷設備、大きなファイルのストレージの装置、磁気テープバックアップステーション、もっと大きな中央の計算機、及び長距離の通信設備の間に伝搬するためにタップを付けられたコアキシャルケーブルを用いる。

共有された通信設備（分岐状のＥＴＨＥＲ）は受動性のものである。ステーションのＥＴＨＥＲＮＥＴインタフェースは往来するＥＴＨＥＲに順にタップインされる送受信機にインタフェースケーブルを介してビット直列に接続される。パケットはそのＥＴＨＥＲの上に放送され，全てのステーションによって聞かれ，そのパケットの先導する番地ビットに従ってそれを選択する目的地によってそのＥＴＨＥＲから複写される。これは放送パケット交換であり，中間の処理要素によって経路決定が実行されるストア・アンド・フォワードパケット交換と区別されなければならない。増大の要求を取り扱うために信号再生のためのパケットリピータ、交通局所化のためのパケットフィルター、及びネットワーク間の番地の拡張のためのパケットゲートウェイを用いてＥＴＨＥＲＮＥＴは拡張されることができる。

制御は統計的調停を介して調整されるパケット伝送によって完全にステーションの間に分散される。ステーションによって起動される伝送は任意の既に進行におけるものであるかもしれないものに従う。一度伝送が開始されもし他のパケットについての妨害が検出されるならばそのソースステーションによって伝送は異常終了させられ，再スケジュールされる。妨害のない伝送の或る期間の後にパケットは全てのステーションによって聞かれ，妨害なしに完了に至るだろう。それぞれの衝突するステーションにおけるＥＴＨＥＲＮＥＴコントローラは繰り返される衝突を避けるためにランダムな再送間隔を生成する。パケットの再送間隔の平均は変化するネットワーク負荷によってＥＴＨＥＲ利用をその最適に近く保つために衝突歴史の関数として調整される。

たとえソース側で検出された妨害なしにパケットが伝送されても依然パケットはエラーなしにその目的地に到着しないかもしれない；このように高い確率によってのみパケットが配達される。その裸のＥＴＨＥＲＮＥＴパケット輸送機構によって提供されるものよりも低い残余のエラーの割合を要求するステーションは相互に同意されるパケット通信規約に従わなければならない。