ICOT Technical Report: TM-0058

TM-0058

MAID: a MAn-machine Interface for Domestic affairs

by
Suzuki Hiroyuki
(Matsushita Erectric Industrial Co., Ltd.)

April, 1984

[Abstract]

This paper describes a design of the 'MAID' system, a natural language interface for a house keeping model. MAID understands a Japanese sentence and performs what is intended by the user.

MAID parses a Japanese sentence using Lexical Functional Grammar (LFG). As a result, an F-structure is passed to the next stage.

MAID uses a new frame system for the knowledge representation. We adopted situation semantics for our underlying theory. It has generic frames described in a form of an event type or a role. The system constructs an event type for a sentence meaning from the F-structure. The algorithm fits Prolog very well.

For getting the utterance meaning, MAID uses two contexts. One is the discourse context and the other is the status of the house. The discourse context is courses of events describing who is saying, and who is meant by 'he', and so on. The status of the house is gotten from the house model using messages. For the utterance meaning, a new event type is gotten by enchoring the event type for the sentence meaning in these contexts.

MAID's model of the house is written in an object oriented form using MANDALA. This makes the system a portable natural language interface and applicable to a sort of hardware systems with slight modifications.

1. Introduction

We are now developing a natural language interface. For the first step, we selected a house keeping model for the application domain. The reasons why we selected that model are:

(1) It is a closed world.

(2) It is a miniature of a real world.

(3) Declarative, interrogative, and imperative sentences can be used.

(4) There is a concept of time.

(5) Attitudes can be ignored.

(6) It is meaningful to use a natural language for user interface.

We adopted situation semantics [Barwise 83] for our theory of meaning. Using the concepts of that theory we can formulate a new frame system that fits Prolog very well.

2. Image of the MAID system

There is a house manager HM which manages all the instruments in the house. For example it knows how the TV set is now, and is able to change channels or to switch the set off. So the user can ask it any questions and order it to do anything about the house as if it were a maid of all work.

When a question is asked, HM interprets the question as a request of how the situation is now. It examines the instrument of the question, and makes the answer. When an order is made, HM performs it, i.e. it changes the status of the instrument. When a declarative sentence is said, HM interprets it as an assertion. So it adjusts the status of the house to the situation described by that sentence.

3. The total system configuration

The MAID system has five sub systems (see Fig. 1). Each sub system is argued in detail later. Here we limit ourselves to the total flow of data and control.

The Japanese sentence by the user is first entered to the parser. The parser is the LFG system in Prolog [Yasukawa 83]. If the parsing process fails, the user is informed why the process fails, for example there is an unknown word or the usage of the verb is wrong. In such cases the user interface tries to eliminate the cause by conversation with the user. The output of the parser is an F-structure.

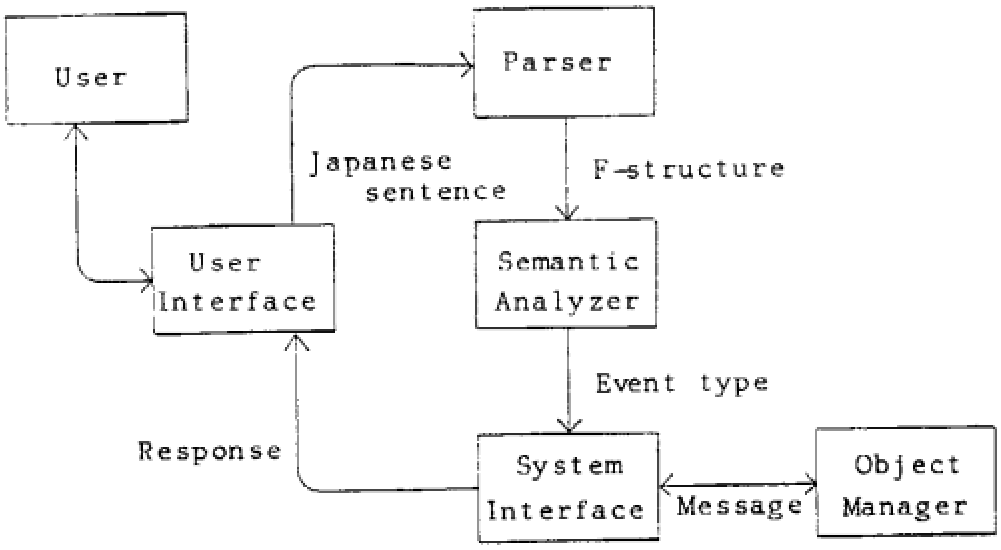

Fig. 1 System Configuration

Next the semantic analyzer analyzes the F-structure and formulates an event type. This event type is the representation of the meaning of the sentence. The event type is interpreted as a relation between the context and the courses of events that represent the utterance meaning.

The system interface does two things. First it anchors the event type and gets courses of events (c.o.e.) for the utterance meaning. Second it performs the c.o.e.'s meaning and get the answer. When anchoring, MAID uses two contexts. The discourse context is stored in this sub system in the form of c.o.e.'s. The status of the house is gotten by message passing mechanism from the next sub system.

The object manager stores the status of the house. It is written in MANDALA/Prolog [Furukawa 83], [Noda 84]. The object manager itself is one of the objects, corresponding to the HM. Therefore all the messages are interpreted as it is defined in the model and the answers are returned.

The user interface gets the answer from the system interface and somewhat modifies it and returns to the user.

4. Parser

We use the LFG system in Prolog. The grammar rules are written according to [Teramura 82]. The core of this grammar is the following three rules. This grammar is one of the X^* V type. Explanation of the categories is presented below.

(1) sentence --> koto + (mood)

(2) koto --> (hogo)* + jutsubu

(3) hogo --> np + (case_marker)

'Koto' is the part of a sentence which describes how the world is. 'Mood' is the other part of a sentence which describes the attitudes towards 'koto' and the listener. In general the analysis of 'mood' is difficult and needs a complicated method. But in this case we have restricted the application area, so grammar rules for mood can be simplified. 'Jutsubu' is made, in principle, of a verb, a adjective, or np+copula. 'Np' is either just a single noun or a noun prefixed by any number of kakari. 'Kakari' is one of (i) np+"no", (ii) sentence which ends in 'rentai kei', (iii) 'rentai_shi'. 'Case_marker' is a particle or particles.

Sample output of this parser is shown in Fig. 2 and 3. Fig. 2 shows a C-structure and Fig. 3 an F-structure.

Two special features of F-structure are used to get the meaning, 'sem' and 'anchor'. 'Sem' contains a name of an event type or a role which is originally written in the dictionary. 'Anchor' contains relations between a surface case and an indeterminate used in an event type that appears in 'sem'.

5. Semantic Analyzer

5.1 Theory of semantics

We have already mentioned that we employ situation semantics for our theory of meaning. But our

```
LFG System (in BUP) Start. Please Input Sentence.

:: daidokoro no suihanki no timer wo 16 ji 45 hun ni set suru.
      ::          ::                        ::
   kitchen     rice-cooker              16 : 45

(The sentence above means
    "Set the timer of the rice cooker in the kitchen at 16:45" )

Time used in analysys is
     2754 ms. for parsing
     3 ms. for checking constraints
     0 ms. for checking completeness

Parsing Result is as follows
```

```
                                                  s
                                                  :
                                                koto
                                                  :
                             hogo--------------------hogo---------jutsubu
                               :                       :             :
                    meishiku-------kaku      meishiku----kaku     doushi
                        :           :            :        :          :
                 kakari---meishiku  :         meishi      :   meishi----+
                   :          :     :            :        :     :       :
                   no      meishi   :         jikoku      :     :       :
                   :          :     :            :        :     :       :
        meishiku--------+     :     :       ji------hun   :     :       :
            :           :     :     :        :       :    :     :       :
      kakari-meishiku   :     :     :   kazu--+ kazu--+   :     :       :
         :       :      :     :     :     :   :   :   :   :     :       :
        no    meishi    :     :     :     :   :   :   :   :     :       :
         :       :      :     :     :     :   :   :   :   :     :       :
meishiku---+     :      :     :     :     :   :   :   :   :     :       :
    :      :     :      :     :     :     :   :   :   :   :     :       :
 meishi    :     :      :     :     :     :   :   :   :   :     :       :
    :      :     :      :     :     :     :   :   :   :   :     :       :
daidokoro no suihanki no   timer  wo    16   ji 45  hun  ni   set    suru
```

(Where   meishi   == noun
             meishiku == noun phrase
             doushi   == verb
             kakari   == adjective phrase)

Fig. 2   An example of C-structure

```
Assignments for category s is

wo
          adjunct
                   {
                    adjunct
                             {
                              sem       sem(role($loc,DAIDOKORO))
                              type      no

                             }
                    sem      sem(role($ins,SUIHAN))
                    type     no

                   }
          sem      sem(role($obj,TIMER))
          pcase    wo

ni
          ji
                    sem      16
                    tan_i    ji
          hun
                    sem      15
                    tan_i    hun

          sem       sem(JIKOKU($x,$y,$z))

          anchor
                   {
                    sem($x(ji))
                    sem($y(hun))
                    sem($z(byou))
                   }

doushi_ka +
anchor
         {
          sem($loc(ni))
          sem($obj(wo))
          sem($hum(ga))
         }
mood      none
sem       sem(SET($hum,$obj,$loc))
yougen    doushi
```

Fig 3.   An example of F-structure

implementation differs from their original theory in two points mentioned below.

(1) We use an event type for the representation of the meaning of a sentence, which is a relation between situations.

(2) The real world is not a static collection of situations, rather it is a collection that dynamically increases during the discourse.

Why we say that an event type is a relation between a situation where the utterance is made and a situation that the utterance means is as follows. From now on, we call the situation where the utterance is made the context of the sentence. The utterance meaning is gotten by anchoring the event type, meaning of the sentence, in its context. So the event type is a relation between the context and the meaning of the utterance.

Hence when the system reads a new sentence, the context should be enriched. And when the next sentence is read, the context has the previous utterance's meaning, i.e. the c.o.e. So the world must be dynamic.

5.2 How to make a meaning of sentence

The sentence meaning has two levels, the attitude level and the fact level. A sentence always has an attitude level meaning. And there are two speaker's attitudes, the attitude to the fact and the attitude to the listener. The attitude towards the fact means what he thinks and feels of the fact, for example, his certainty about the fact, his estimation of the fact, and his desire for the fact. Examples of the attitude towards the listener are his respect to the listener and his desire for the listener how to interpret the fact. The fact level meaning is embedded in the attitude level meaning.

The attitude level meaning mainly comes from particles, auxiliary verbs, and adverbs. The top frame of the fact level meaning comes from the main verb of the sentence.

Now let's ignore the attitude level meaning and see how to make the fact level meaning of the sentence.

The dictionary used by the parser has name entries to event types. A noun corresponds to a role, and a verb, a adjective, and so on to a event type.

See the F-structure shown in Fig. 4. That F-structure is a part of the one generated for the sentence " Terebi wo tsukero. ", which means " Switch the TV on. " The rest of the figure is showing how to make the event type for that sentence.

The outermost event type for the fact level meaning is generated from the event type of the dictionary entry for the main verb of the sentence. In this case it is the event type named 'TSUKERU($hum,$con)'.

Then the indeterminates used in that event types are to be anchored according to the information held by the 'anchor' feature. In this case, $con, which means a role named con, is anchored (or rather better to say 'unified', here) with the role that is held in the 'sem' of 'wo', i.e. $tv.

As a result, the event type for the sentence is

```
wo
                        sem      role($tv,TELEVISION($tv))
                        pcase    wo

sem                     TSUKERU($hum,$con)

anchor
                       {
                        $con(wo)
                        $hum(ga)
                       }
```

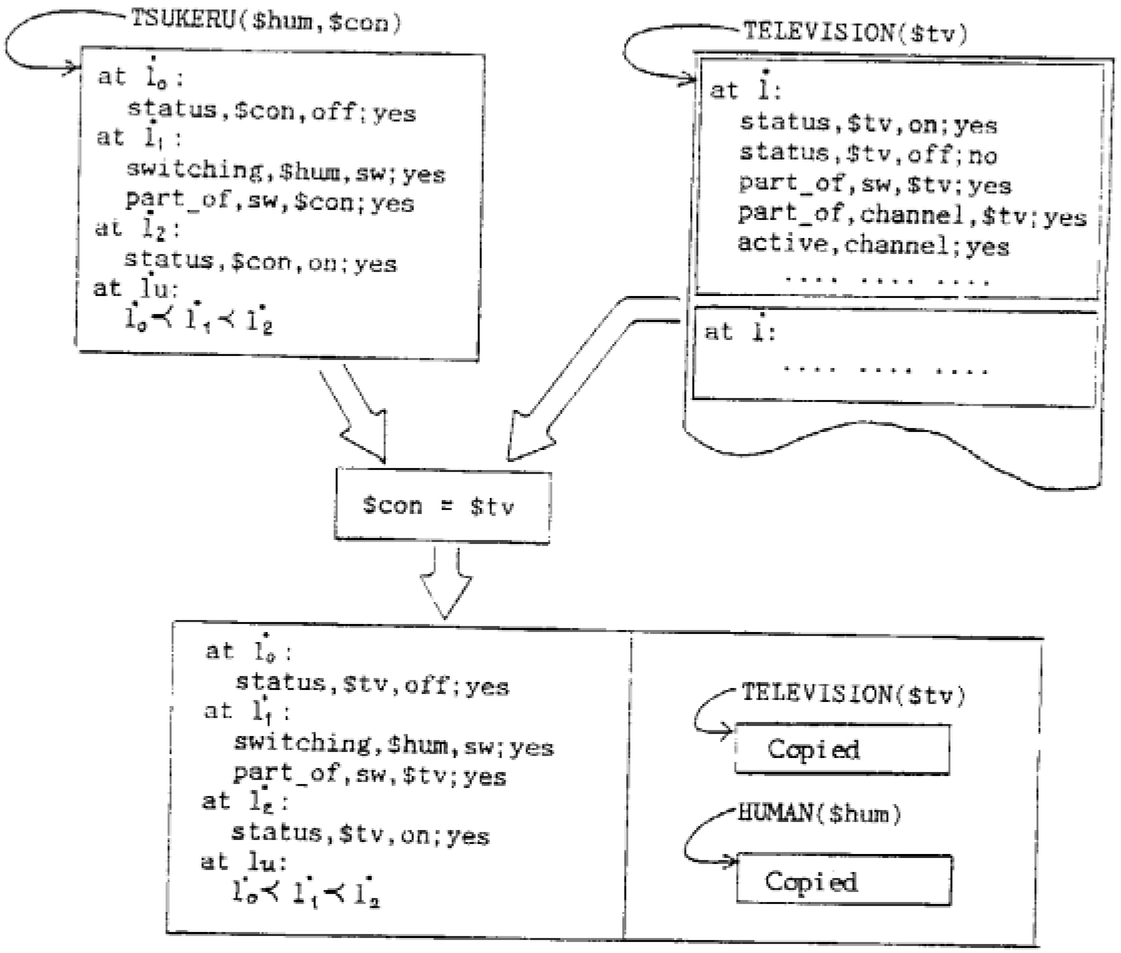

Fig. 4 How to make an event type from an F-structure

generated and it is shown at the bottom of Fig. 4.

Above is the one main method. There is another main method for adjective phrases, i.e. ap + np. In that case, if ap corresponds to the event type Ea and np corresponds to the role $r=<$x,En>, then ap + np corresponds to <$x,En+Ea>. The addition of event types is principally a set union, and the relations between the indeterminates used in both event types are determined by the context.

Using these two methods, one can get an event type for the fact level meaning of any sentence.

Here We would like to mention that using this strategy any part of sentence always corresponds to an event type or a role. So elliptical sentences can be treated in the same manner.

The attitude level meanings are not treated in this system except if the sentence is declarative, imperative, or interrogative.

6. System Interface

This sub system's main purpose is to get an utterance meaning and to do its interpretation.

First this sub system receives an event type from the semantic analyzer and then anchors it in the context. The context has two parts in this design, the discourse context and the status of the house. The discourse context is stored in this sub system in the form of the collection of courses of events (c.o.e.'s). The other is described as the next sub system.

The discourse context is mainly used to solve references of pronouns. So it is used to anchor special roles. This anchoring is done by finding the c.o.e. of the event type used in that special role.

The status of the house is the situations of the world. The reality is held by the the object manager in a object oriented form. So this system interface should generate an abstract situation from that. And using the abstract situation, anchoring is done by the same way as above.

Next this sub system should generate a message which causes the object manager to change its status as desired by the user. This generation is done by examining if the c.o.e. is meaningful with respect to a message-generating constraint. Then it sends the message to the object manager and get the response. The response is then passed the user interface.

Why we divide the context in two is as follows. First we'd like to make a portable interface, so the application domain's status should not be stored in the three core sub systems described above. Second to test our representation of meanings, we need an other representation of the world. And the representation employed in the object manager system is very natural and very similar to hardware systems.

7. Object Manager

The object manager contains the status of the house. The object manager simulates the real world. So it is

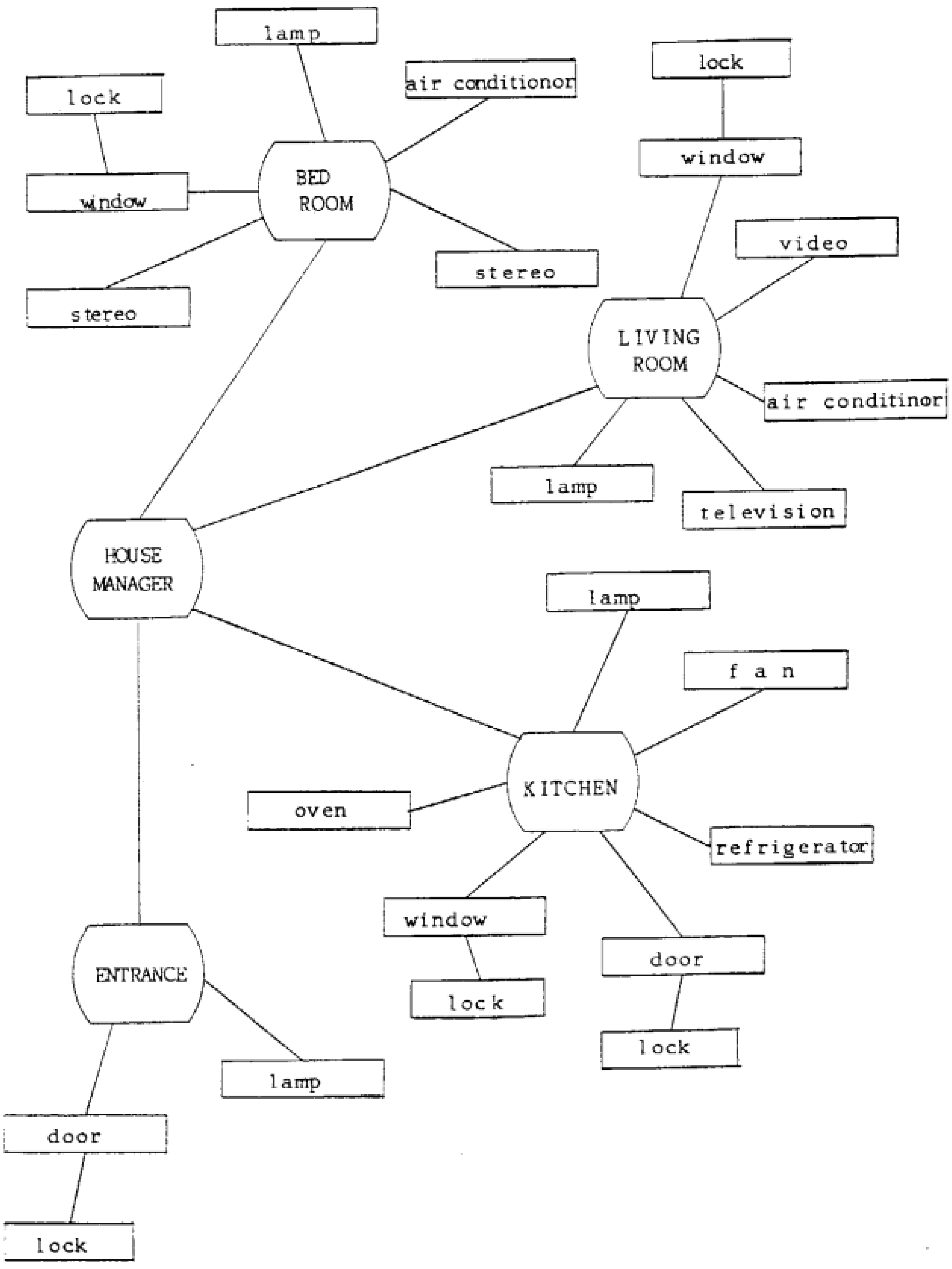

Fig. 5 The image of the object manager

desirable that it works just like the real systems. And this part is not the main focus of this study, the time to develop the object manager should be short. Those reasons make us to choose Mandala/Prolog as the language to describe this part.

The image of the house is shown in Fig. 5. An instrument in the house is described as an object of its class, e.g. the video recorder in the living room belongs to the 'video' class. The house manager and the room managers are described as objects of the 'distributor' class.

The message from the system interface is first feeded to the 'House Manager'. Then it distributes the message to the object it is addressed via a room manager. The object responds the message according to the definition in its class description and changes its status. One can also ask the status of the object. The response is returned to the system interface.

These message passing is done through channels. So the replacement of the instrument in the house can also be simulated by detaching and attaching the channel between a room manager and the object. This is realizable by a message which the 'distributors' understands.

8. New Frame System

If you carefully examine the algorithm written above, you'll see that this is a new FRAME SYSTEM and it fits Prolog. Fig. 6 is the correspondence chart between the

	Our System	Frame System
	Event type	Frame
	Role	Slot
slot value restriction	Event type	special feuture or functions
	Anchoring	Instantiation
inheritance	predicate	function

Fig. 6 Correspondence chart

frame systems implemented in LISP and our frame system.

An event type is a frame and roles abstracted from the event type are slots of the frame. And slot restrictions are described in two ways. Characteristics of the slot value are described by that the indeterminate itself is an some other role, and relations between slots are written as they are, i.e. constituent sequences of the event type.

Event types in the dictionary are generic frames. The semantic analyzer copies those generic frames and makes a new event type for a skeleton of the meaning of the utterance. This is done by the unification of the indeterminates. And the system interface makes the specific frame by anchoring the skeleton in the context. This anchoring method makes use of the backtracking mechanism essentially. Just see one example (Fig. 7). The event type 'LOVE' is composed of only one constituent sequence. And the roles $lover and $beloved are

```
LOVE := at l: love $hum1 $hum2;yes.

$lover   = <$hum1,LOVE>
$beloved = <$hum2,LOVE>


SUICIDED :=
     at ls:
         sex_of $lover female;yes
         sex_of $beloved male;yes
         love $beloved $lover:no
         kill $lover $lover;yes.

(the above event type corresponds to the sentence,
    "The girl who is not loved  by the man who she  loves
     kills herself.")

in Context :
     at lc:
         sex_of mary female;yes
         sex_of jane female;yes
         sex_of john male;yes
         love mary john;yes
         love jane john;yes
         love john mary;yes
         love john jane;no.


Then the anchor is
         $lover   ==> jane
         $beloved ==> john
```

Fig. 7 Anchoring an event type in a context

abstracted from it. When the event type 'SUICIDED' is anchored in the context, backtracking is used.

Furthermore, if you want to do an expectation by keywords, you can do so by describing the expected event type in the lexicon of the keyword. (There is another method for this purpose, using constraints.)

There is already an implementation of some part by a slightly different idea. see [Mukai 84].

9. Summary

We have just designed the QA system for the house keeping model. This system uses:

(1) LFG Parser in Prolog,

(2) New Frame System based on Situation Semantics for Knowledge Representation, and

(3) Mandala/Prolog for modelling the House.

The frame system is newly conceptualize by the Prolog's point of view.

There remains many works, including:

(1) Implementing the whole system,

(2) Refine the New Frame System, and

(3) Designing the sentence generation algorithm.

11. Acknowledgement

I would like to thank Masaki Kiyono for the suggestion to use constraints for generating messages. I am very grateful to Kuniaki Mukai and Hideki Yasukawa for helpful discussions and advices. I also thank all members of the ICOT Research Center and the New Generation Computer Lub. both for help with research and for providing a stimulating place.

This research is sponsored by the Institute for New Generation Computer Technology (ICOT), Order 3301-02.

12. References

[Barwise 83]

J.Barwise and J.Perry : "Situations and attitudes", MIT Press,1984

[Furukawa 83]

K.Furukawa, A.Takeuchi, and S.Kunifuji: " Mandala, A Knowledge Programming Language on Concurrent Prolog", ICOT TM-0028, 1983 (in Japanese)

[Mukai 84]

K.Mukai : "Towards a Computational Model of Situation Semantics and its Implementation in Prolog", ICOT TM-0051, 1984 (in Japanese)

[Noda 84]

Y.Noda et al. : " CAD System kouchiku ni kansuru ichi kentou", Proc. of the Logic Programming Conference '84, 1984 (in Japanese)

[Teramura 82]

H.Teramura : "Syntax and Semantics of Japanese I", Kuroshio Press, 1982 (in Japanese)

[Yasukawa 83]

H.Yasukawa : "LFG in Prolog", ICOT TR-019, 1983