American Journal of Computational Linguistics *Microfiche 31*

A CASE-DRIVEN PARSER FOR NATURAL LANGUAGE

Brock H. Taylor and Richard S. Rosenberg

Department of Computer Science
University of British Columbia
Vancouver, B. C., Canada V6T 1W5

Abstract

This paper describes a system for analysing natural language based on the concept of case. After a preliminary parse using an augmented transition network, the case routines attempt to find the appropriate verb meaning. These routines search for parts of the syntactic structure which best satisfy the requirements of the verb case frames and under back-up are able to weaken gradually the conditions for success. The resulting structure is similar to the conceptual dependency networks of Schank, and is an attempt to represent as fully as possible the meaning of the input sentence. The system has been designed to be quite flexible and allows for the incorporation of domain specific knowledge. This knowledge has its effect both in the nature of the dictionary and in modifications in the search routines. At present the system incorporates procedures for resolving anaphoric references which depend on examining previous sentences.

Table of Contents

A CASE-DRIVEN PARSER FOR NATURAL LANGUAGE

Brock H. Taylor and Richard S. Rosenberg

1. Introduction

It has been apparent for some time now that a linear approach to natural language processing consisting of successive syntactic, semantic, and retrieval or inference phases is inadequate. An important advance was made by Winograd[9,10] in integrating these phases into a system which utilizes the various kinds of linguistic knowledge at the appropriate time. The approach described in this paper is somewhat less ambitious, but this is partly compensated for by its very flexible structure. In fact, it is useful to view this system as a skeleton which can be fleshed out to serve a wide variety of purposes.

Although called a parser, it is far more powerful than a traditional parser because it incorporates semantic knowledge to produce a representation of the input sentence which is as rich as possible in terms of the system's basic knowledge. Since the

[1] Contact R. S. Rosenberg for information on [8].

detailed operation of the system will be described subsequently, we will now present the major influences on this work, some similar systems, and what we believe the important contributions to be.

Beginning at the end, we decided to represent an input sentence with a structure which is very similar to the conceptual dependency networks of Schank[5,6]. This does not imply agreement with the overall philosophy of Schank, but rather a recognition that an underlying representation should contain as much knowledge as possible, as it may be crucial for subsequent analysis. As will be seen, our representation, in addition to the basic syntactic relations, also reveals semantic relations not explicitly given in the input sentence. This latter knowledge is derived from a complex semantic lexicon organized around the concept of case as first formulated by Fillmore[3]. One point should be emphasized: Fillmore, as a linguist, was concerned with formulating a theory to explain data which a transformational approach seemed unable to do. As such, he felt the need to worry about the number and nature of the cases necessary to treat the linguistic data adequately. This is to be contrasted with our approach which is to use an extended version of case in order to represent the meaning of a sentence as fully as possible. The basic difference is revealed in Fillmore's description of five or six cases, whereas our system uses, at present, twenty-four cases.

Perhaps the term "case" is inappropriate here, but there is enough similarity with, and motivation from, Fillmore's work that we decided to use the term. The most complex part of the lexicon is the verb with its associated case frame: actually an environment of obligatory and optional cases associated with the verb. One basic problem of sentence analysis is to choose among alternate verb meanings for an appropriate candidate. This selection process is governed, in part, by attempting to satisfy the constraints imposed by the case frames associated with each verb meaning. But prior to activating the case-driven part of the system, a preliminary stage of analysis must be initiated. This is an almost purely syntactic phase carried out by a rather simple augmented transition network (ATN), (Woods[11,12]). The ATN has proven to be very useful in natural language processing mainly because of the ease of representing complicated and interrelated syntactic structures.

For our purposes, the ATN is used to produce a very fast preliminary parse of the input sentence which indicates gross structural relations. Using this parse, the case-driven component seeks to select the appropriate verb meaning. It is important to note that if the case procedures fail on the first pass, conditions for success are progressively weakened until the most suitable meaning is chosen. Although not a feature of the present system, it would be possible to re-enter the ATN phase in order to produce another parse if the case phase were unable to complete its task in a satisfactory manner.

We would like to stress those aspects of the system which make it flexible and useful for a wide range of language processing applications. It is straightforward to incorporate different kinds of knowledge necessary for adequate processing. For example, the current system has a procedure for resolving a fair range of anaphoric references for pronouns. If additional procedures are developed, they can also be incorporated into the system, and will exert their influence by modifying the search procedures for candidates which satisfy the requirements of the case frame for verbs. Another important feature is the facility within the dictionary entries of nouns for providing information about relevant properties, such as superset and subset. Thus a kind of semantic network links nouns of the dictionary, and this knowledge is available to aid in the processing.

The user can construct the dictionary appropriate for his purposes and can readily add necessary domain specific features. This system is considerably more than just a front-end for a traditional linear language processor. It integrates syntactic and semantic linguistic knowledge in a particularly transparent and flexible manner.

There are other current language systems which are based on notions of case but which differ mainly in the way the processed sentence is represented. We might mention Simmons[7], Martin[4], and Bruce[1].

The system is written in LISP/MTS, and runs on an IBM 370-168 under the MTS operating system at the University of British Columbia. The code occupies 240K bytes, and the current dictionary of 450 words occupies an additional 90K bytes. When the system is running the total space used is 470K bytes. In spite of its large size, it is relatively fast. For example, the total time taken to parse sentence (11) below, is .90 CPU seconds, executing interpretively. A compiled version of the program would run approximately 10 times faster.

2. Cases and Verbs

The question of how many cases we need to describe English is a contentious one. Fillmore[3] is vague on the issue, whereas Celce-Murcia[2] claims that five cases will do. The purpose of this work is to capture as much of the meaning of a sentence as is possible, and to make it explicit in a formal structure. Case, then, is an explanation of the semantic function of a sentence part; therefore a fairly large number of cases have been used, one for each of these "semantic functions". We are not adamant about our set of cases. The system is flexible and structured enough, that the addition or deletion of cases is a simple operation.

The system was originally designed with Martin's[4] thirty-odd cases as its basis. There are currently twenty-four general cases implemented, plus numerous verb-specific cases. Some of Martin's cases have been dropped completely, some have simply not been implemented yet, and several new cases have been added.

For a complete list of these cases we refer you to Taylor[8]. A few will be listed here, and they will later be used to aid in the description of the system. The underlined phrases represent the appropriate cases.

2.1 A Partial List of Cases

Agent

<u>The man</u> in white says he has no friends.

I got wiped out by <u>several charismatic holy men</u>.

Patient

He trained <u>a hundred women</u> just to kill
<u>an unborn child</u>.

I washed <u>my eyelids</u> in the rain.

<u>Path</u>

We started up <u>the mountain</u>.

Through <u>the graves</u> the wind is blowing.

Flagged by by, about, along, up, down, around, across.

<u>Exchange</u>

I bought it with <u>a nickle</u> and I sold it for <u>a dime</u>.

He wants to trade the game he plays for <u>shelter</u>.

Flagged by with, for.

<u>Beneficiary</u>

I fought every man for <u>her</u> until the night was over.

I sing this to <u>the crickets</u>, I sing this for
<u>the army</u>.

Flagged by for, to, before.

<u>Descriptive</u> (A case of the Noun)

Suzanne takes you down to her place near <u>the river</u>.

The woman in <u>blue</u> is asking for revenge.

The hand of <u>your beggar</u> is burdened down with money.

Flagged by of, from, at, in, on, with, by, near, beside, before, after, along, up, down, around, across, under.

Enable

I had to kick you down the stairs so I could savour unemployment once again.

We put her away so we could get back to the war.

Flagged by so.

Topic

It is time we began to laugh about it all again.

Lets not talk of love or chains.

Sometimes I find I get to thinking of the past.

Flagged by of, about.

2.2 Determination of Case

Most of the cases listed above are associated with the prepositions that flag them. This is, of course, a gross over-simplification of the relationship between verbs, prepositions, and cases. A preposition which flags one case for one verb may very well flag another case for a different verb.

(1) I walked about the room.

(2) I talked about the room.

For verbs of movement like "walk", "about" flags the path case as in (1), but for verbs of communication like "talk", it usually flags the topic case as in (2). Martin[4] notes this, and proposes for each verb meaning to list all of the cases flagged by each preposition. This involves a great deal of repetition, however, since most prepositions flag almost the same set of cases for most verbs. For this system, therefore, we set up a master-table of all the cases flagged by each preposition, then for each verb, just the irregularities are noted.

(3) The scandal was whispered about the room.

Sentence (3) illustrates that "about" <u>can</u> flag the path case for a communication verb, so we do not want to rule out the path case: we just want topic to be tried first. In (3) the topic slot will already be filled by "the scandal", so topic will be rejected, and the path case will be tried next.

This foregrounding of cases is specified in the dictionary. For instance, for the verbs talk, laugh, whisper, etc., it is specified that the occurrence of the preposition "about" should trigger the topic case before the path case.

There is another obvious way of determining case names for prepositional and noun phrases. Consider the sentences:

(4) Fred bought the car for mary.

(5) Fred bought the car for one dollar.

The preposition "for" flags many cases. In (4) it flags the beneficiary case, and in (5) it flags the exchange case. Associated with each case is a test which a phrase must pass to be accepted. In the beneficiary case the test is: (MUST-BE ANIMATE), indicating that the beneficiary has to be animate, while the test for the exchange case is: (NOT (SHOULD-BE HUMAN ABSTRACT)), indicating that one does not usually exchange something for a person or something abstract. These tests correctly sort out the cases in sentences (4) and (5).

In general, tests on cases are very difficult to design adequately. What test would be appropriate for the topic case? What could not be talked, laughed, or cried about? Perhaps some complex verb and context dependent test could be concocted, but one has not been designed for this system. The test for the topic case is therefore one which will always pass. One must therefore be careful when invoking the topic case.

The exchange case has similar problems. Anything can be exchanged for something. The weak test

(NOT (SHOULD-BE HUMAN ABSTRACT))

is put in, which will at first fail if a human or an abstract noun is the candidate, but will pass if nothing else seems to fit either. This simple test runs into problems with certain sentences.

(6) I paid the money for my mother's release.

(7) I paid the money for my mother.

(8) I paid the money for the prostitute.

It will initially force the exchange case to reject "for my mother's release" in (6) because it is abstract, but later on it will accept it since all of the other cases flagged by "for" will also reject it. Sentence (7) is ambiguous, but "mother" is almost certainly in the beneficiary role here, so again the test works correctly by rejecting the exchange case. Sentence (8) is also ambiguous, but our interpretation would usually be that "prostitute" is in the exchange case here. The system will, however, assign it the beneficiary case as it did in (7). Additional work must be done on case tests if this paradigm is to be useful.

2.3 Verb Specific Cases

Many verbs have special constructs or cases which are not used with most other verbs. These irregularities are handled by writing special functions to find these cases. A few examples will illustrate.

The verb "to be" has eight meanings in this system. The third meaning is "to have the property. . ", as in sentence (9).

(9) The house is red.

This meaning is the one being used if an adjective phrase immediately follows the verb. An adjective phrase in this position is therefore a special case of the verb "to be", and there is a special function, ADJ-LIST, which looks for it.

The sixth meaning of "to be" is "to be from . . "., as in sentence (10).

(10) The lady is from Ouagadougou.

This could be interpreted as an example of the source case, which is the case that "from" usually flags; but what the sentence really means is that the lady has been living in Ouagadougou. This is, therefore, not the source case, but another special case of "to be".

2.4 Verb Definitions

The verb is treated as the focal point of the sentence. A verb can have many meanings. The system discovers which meaning is intended by looking at the rest of the sentence. In so doing, it builds a structure representing a parse of the sentence.

As stated above, each verb has associated with it a case-frame, which is a set of cases of the verb: some obligatory, some optional, and some conditionally optional. These cases are embedded in a form on the property list of the verb. Consider the verb "to order." Its dictionary entry is as follows:

```
(ORDER V
       S-ED
       PREP-CASE ((WITH WITH))
       V-MEAN
       (IF ((AGENT (MUST-BE HUMAN))
             AG
             (OPT (GETR PASSIVE) 'SOMEONE)
             (PATIENT (MUST-BE ANIMATE))
             PA
             OBL
             (TO-COMP (GETR PA))
             TOC
             OBL)
           (BUILDQ ("<==>" ? "+" ("<--" ORDER "+")) AG TNS TOC)
           ((AGENT (MUST-BE HUMAN))
             AG
             (OPT (GETR PASSIVE) 'SOMEONE)
             (PATIENT (AND (MUST-BE THING) (NOT (MUST-BE HUMAN)))
             PA
             OBL)
           (BUILDQ ("<==>" ? "+" ("<--" ORDER ?)) AG TNS PA)))
```

Under the indicator V-MEAN there is a form beginning (IF ((AGENT . . . IF is a function which takes an even, but otherwise variable, number of arguments, each pair representing a meaning of the verb. The first element of each pair is a set of cases to be looked for, and the second is the structure to be built if they are found. It is in the first element of the pair that the complexity lies. Let us look at it more closely.

The list of cases is, in fact, a list of triples. The first element of the triple is a form to be EVALed. It is usually looking for a case, but any form is admissible. The second element of the triple is an atom: a register name. If the first form EVALs to a non-NIL value, the value is put into this register. In our example, for instance, the first triple is:

(AGENT (MUST-BE HUMAN)) AG (OPT (GETR PASSIVE) 'SOMEONE)

The function of the first form is to find the agent of the sentence. If it succeeds, this agent is put into register AG.

The third element of the triple indicates what to do on failure. If it is the atom "OBL", this indicates that the case was obligatory; so if it was not found, IF should fail on this meaning of the verb. If the atom is "OPT", then the case is optional, the register is left empty, and IF continues with this meaning. The third possibility is that this third element is a form, in which case it is EVALed. If it returns "OBL" or "OPT", then the result is as described above. If it returns anything else, then that is put into the register, and IF continues with this meaning of the verb.

The third element of the first triple for "to order" is (OPT (GETR PASSIVE) 'SOMEONE). OPT is a very simple function which, if its first argument is non-NIL, returns its second argument. Otherwise it returns "OBL". (GETR PASSIVE) is true if the sentence is in the passive voice. The first triple can be read as follows:

Look for an agent which must be human. If you find one, put it in register AG. Otherwise, if the sentence is passive, make SOMEONE the agent. Otherwise fail.

The second triple is simpler. It merely says: If you find an animate patient, then put it in register PA, else fail.

The third triple is equally simple: it is not looking for a case, but a to-complement.[1] If these three elements are found in the sentence, then the system will look no further, but assume that it has found the correct meaning of the verb. It will EVAL the second form of the pair, in this case:

```
(BUILDQ ("<==>" ? + ("<--" ORDER +)) AG TNS TOC)
```

which builds the basic structure for the sentence.

BUILDQ takes a variable number of arguments. The first is a kind of template with slots in it. The rest of the arguments fill the slots. The "+" denotes a slot which is filled by the contents of a register. NOUN-PUT returns the structure of the noun phrase associated with the noun in this register. The "?" is filled by the application of the function NOUN-PUT to the contents of a register. Finally, the "#" (see Appendix) indicates that a form is to be EVALed, and the result put into

[1] An example of a to-complement is: "Fred took the book to anger Mary."

the slot. The slots are filled in order by the second, third, etc, arguments. It should be noted that the form of BUILDQ has been strongly motivated by its use in Woods' ATN [11].

So in this case:

(NOUN-PUT (GETR AG)) is put in for the ?.

(GETR TNS) in place of the first +.

(GETR TOC) for the second +.

where GETR returns the contents of a register.

3. A Detailed Example

Programming details do not belong in a paper of this kind. All of the code is in Taylor[8] for those interested. In the following example, then, function names and excessive details will be, on the whole, left out. A detailed account of the basic algorithm and control structure will be given. We will look at a simple sentence. More complex structures such as relative and subordinate clauses are treated in much the same way as their parent sentences. Consider the sentence:

(11) The man beside the window played the piano for Mary.

As stated above, the first step in the process is a partial parse using an ATN. The structural description usually derived from this parse is incomplete. That is, no decisions are made about what modifies what, what meaning of the verb is being used, etc. The basic idea behind the ATN is to find the verb but while it is doing this, it seems useful to chop the sentence up into its parts. There are problems with just how this chopping should be done, but with most sentences it is straightforward.

The ATN parse returned for sentence (11) will have the following form:

```
S
    NP NIL
        DET THE
        N MAN
            NUMBER SG
    PP NIL BESIDE
        NP NIL
            DET THE
            N WINDOW
                NUMBER SG
    VP NIL
        TNS
            PAST
            VOICE ACTIVE
        V PLAY
    NP NIL
        DET THE
        N PIANO
            NUMBER SG
    PP NIL FOR
        NP NIL
            NPR MARY
```

It is on this preliminary parse that the program works.

First, the main verb is found, and a function is invoked which controls the top-level back-up. This function EVALs the form on the property list of the verb under the indicator V-MEAN. This form for PLAY is a very long one, and is given in the appendix. The form in question is a call to IF, whose mechanism has been briefly described above. In this instance IF has ten arguments, indicating that there are five meanings to the verb PLAY in the system. The first meaning is "to play a musical instrument."

The first case looked for is the AGENT. This agent should be a musician, and must be human. This search is initiated by EVALing the first form in the first triple of the first argument to IF: (AGENT (AND (SHOULD-BE MUSICIAN)(MUST-BE HUMAN))). AGENT is fairly complex, but basically it looks for a component of the ATN parse (in future called the "p-parse", for partial-parse) which is in an appropriate position to be an agent, and which passes the test (the argument to AGENT.) By 'appropriate position' is meant, for instance, that if the sentence is in the active voice, the agent is probably the first noun phrase in the sentence.

For this situation, AGENT immediately finds "the man" as the obvious candidate, and it applies the test (AND (SHOULD-BE MUSICIAN) (MUST-BE HUMAN)). Now, unless something special has been put on the property list of MAN previously, the (SHOULD-BE MUSICIAN) part of the test will fail. (There are two levels of tests in this system: SHOULD-BE tests and MUST-BE tests. This mechanism is very useful for forcing a verb like PLAY to look very hard for a musician to play an instrument -- but to accept any human if it fails at first. This is especially powerful for resolving anaphoric references). Thus AGENT fails, which invokes the third element of the AGENT triple: (OPT (GETR PASSIVE) 'SOMEONE). This may be read as: AGENT is optional if the sentence is in the passive voice, in which case put SOMEONE in as the agent; otherwise AGENT is obligatory. Since the sentence is not passive, AGENT is obligatory. As the AGENT case was not found, this first meaning of PLAY fails.

IF then goes on to the next pair of arguments. This pair is designed to pick up the meaning of PLAY as in "to play music." Note that the test on AGENT is just like the previous one, which means failure here as well. The program moves on to the third meaning of to PLAY: "to play a sport." Here the test on AGENT is (AND (SHOULD-BE SPORTS-MAN) (MUST-BE HUMAN)). Once again, providing MAN does not have SPORTS-MAN on its property list, this attempt fails. The program therefore goes onto the fourth meaning which is designed to pick up the ergative usage of "to play" as in "The music played from the room." Since the test for this meaning is (MUST-BE MUSIC), this meaning will also fail. On to the fifth, and last, meaning, which is a sort of catch-all. It is the meaning of "to play" as in "to entertain oneself." Here the test on AGENT is (MUST-BE ANIMATE). "The man" passes this test, since MAN has the property ANIMATE. Since AGENT is the only case looked for, this meaning is taken to be the correct one, and the following structure is built by the call to BUILDQ:

```
<==>
    N MAN
        NUMBER SG
        <-DEFINITE- THE
    PAST
    <-- DO
    <-CAUSE-
        <==>
            N MAN
                NUMBER SG
                <-DEFINITE- THE
            PRESENT
            <-- HAVE-PROP ENTERTAINED
```

IF has completed its job. It has found what it takes to be the correct meaning of the verb. Now the rest of the sentence must be processed. The second element of every top-level list in the p-parse is a flag which is initially NIL, but which is turned on when that part of the sentence is considered to be correctly dealt with. In our example, so far only two parts are flagged: the first noun phrase: "the man", and the verb phrase. The function which takes care of the rest of the sentence simply goes down the p-parse checking these flags. If it finds one which is NIL it works on that part of the sentence until it either succeeds, or fails -- causing back-up.

For this example, then, the first phrase it comes upon needing work is the prepositional phrase; "beside the window". As mentioned above. there is a master-table in the system which associates each preposition with the cases it may flag. BESIDE flags the cases: LOCATION and DESCRIPTIVE. All of the cases but DESCRIPTIVE are cases of the verb. DESCRIPTIVE is a special case which is used for preposition phrases which modify nouns.

When the list of cases associated with a preposition is retrieved, there is a question as to which case to try first. For this there is a foregrounding routine, with several criteria for foregrounding:

First of all, in the dictionary definition of the verb, the user may specify that a certain preposition trigger a particular case program. Since there is no such specification for "to play" in the current dictionary, nothing happens here. Secondly, on the property list of each verb is kept a record of which prepositions flagged which cases in the previous sentences. The cases associated with the preposition in question (if there are any) are foregrounded, so that they will be tried first (the most recent case first, etc.) Finally, if DESCRIPTIVE is one of the cases in the list of cases for this preposition, and if a noun phrase or a prepositional phrase immediately precedes the phrase in question, and if the noun in that noun phrase or prepositional phrase is not a proper noun, then DESCRIPTIVE is put at the front of the list, and is thus tried first.

This seemingly obscure rule for foregrounding the DESCRIPTIVE case is just a heuristic. If the tests associated with each case are good enough, it makes no difference to the final outcome if the foregrounding is done or not. In some instances, however, if the DESCRIPTIVE case is not tried first, it will never be tried. In our example, for instance, it is the man who is beside the window (DESCRIPTIVE case); he did not play the piano beside the window (LOCATION case). But it is perfectly feasible for him to have played it beside the window (if we know nothing about the location of the piano.) Therefore either of the cases will succeed. It is only the position of the prepositional phrase that indicates which case is correct.

Continuing with our example: the DESCRIPTIVE case is foregrounded, and so the descriptive case function, DESC, is invoked with the phrase "beside the window" as its argument. Since the descriptive case almost always involves a prepositional phrase modifying the noun phrase or prepositional phrase immediately before it, DESC first checks to see if "beside the window" is a possible descriptor of "the man."

Since we do not have a data base to check to see if there is a man beside a window, our check must be a general one. Most nouns have a size associated with them under the indicator OBJ-SIZE. This is a very crude breakdown of physical objects into eleven size categories. "The world" is size 10 and "a pin" is size 0. (These sizes should be able to be changed by classifiers, adjectives, or modifying phrases. A toy elephant is probably not the same size as an elephant. This feature is currently not implemented.) The check for "beside" is merely used to rule out things like "the pin beside Canada." Because abstract nouns have no size information, sentences like "He had a thought beside the ocean" are not ambiguous. In any event, "beside the window" is found to be a likely modifier of "the man", and DESC succeeds. Since "beside" is a locative preposition, DESC returns the structure:

```
(<-LOC- BESIDE (NP (N WINDOW (NUMBER SG) (<-DEFINATE- THE))))
```

A form is stacked which will put this structure into the main sentence structure if the rest of the sentence can be handled. Just where it is placed is determined by DESC. Since the prepositional phrase modifies "the man", it will be put in as follows:

```
N MAN
    NUMBER SG
    <-DEFINITE- THE
    <-LOC- BESIDE
        N WINDOW
            NUMBER SG
            <-DEFINITE- THE
```

so the prepositional phrase "beside the window" is flagged as completed, and the next unflagged phrase, "the piano", is picked up.

Here we run into problems. Where does "the piano" fit into the structure? What does it modify? What is its case? There are relatively few ways a noun phrase can be used at this point. It could be an example of the TIME case, as in "I came home <u>this morning</u>.", but "piano" fails the TIME-test. It could be a classifier, but the phrase following it would have to be a noun phrase for this to be the case. So failure has occurred. Something has gone wrong. IF must have chosen the wrong meaning of the verb. The program must back up.

All the parts of the sentence flagged as used are unflagged, and back-up occurs into IF again. Here it is found that there are no meanings of the verb left to try. One of the meanings that was rejected earlier must have been the correct one. So IF fails entirely, and the program enters the top-level back-up mechanism.

There are two possible reasons failure has occurred:

1) Either the program did not look back far enough in an attempt to resolve an anaphoric reference, or 2) The tests were too severe. (ie: the SHOULD-BE tests caused failure when they should not have.)

The anaphoric part of the system has not been explained yet, but as there were no pronouns in the sentence, the first reason can be ruled out. In order to weaken the tests, a flag is set to shut off the SHOULD-BE tests. That is, all SHOULD-BE tests will succeed in future. The process begins again with IF.

The beginning is the same, but this time the first invocation of AGENT will succeed, because the test (AND (SHOULD-BE MUSICIAN) (MUST-BE HUMAN)) succeeds. The structure it returns is put in the register AG. IF continues with the second triple of parameters, and the form (PATIENT (MUST-BE MUSICAL-INSTRUMENT)) is EVALed. Now, PATIENT is very similar to AGENT: it looks in the appropriate place in the sentence for the patient of the verb. It then applies its TEST to it. In an active sentence, such as our example, the candidate for PATIENT is the first noun phrase after the verb. "The piano" is found, and since it passes the test (MUST-BE MUSICAL-INSTRUMENT), PATIENT returns "the piano" as the patient of the sentence.

Once again it seems that the correct meaning of the verb has been found, therefore IF EVALs the BUILDQ associated with that meaning. The following structure is built:

```
<==>
    N MAN
        NUMBER SG
        <-DEFINITE- THE
    PAST
    <-- DO
    <-CAUSE-
        <==>
            N PIANO
                NUMBER SG
                <-DEFINITE- THE
            PAST
            <-- EMIT
                NP
                    N SOUND
```

It now remains to try to clean up the unflagged parts of the sentence. The first one, again, is "beside the window", and exactly the same thing is done as was done previously: it is decided that "beside the window" is a locative descriptor of "the man", and this decision is stacked for later action.

The only other part of the sentence to be handled is "for Mary." As with "beside", the cases associated with "for" are returned from the CASE-TABLE. They are: DURATION, BENEFICIARY, EXCHANGE, and IND-SUBJ. (IND-SUBJ has not been implemented yet.) Assuming that there have been no relevant previous sentences, the foregrounding of cases will have no effect on this ordering.

The DURATION case is tried first. DURATION is a particularly simple case. Basically it checks to see that the noun phrase in the prepositional phrase has the property TIME under the flag N-PROP. "Mary" fails this test, and DURATION is rejected.

The next case is BENEFICIARY. The only test for this case is that the noun phrase be animate. "Mary" passes this test since it has the SUPERSET WOMAN and WOMAN has the N-PROP ANIMATE. Therefore BENEFICIARY succeeds and returns: (<-BENEFICIARY- (NPR MARY)). Unlike "beside the window", this phrase is a case of the verb. Because all cases of the verb (but AGENT and PATIENT) are considered to be essentially parallel with respect to the verb, they are put into the structure at the same level, that of the verb symbol "<--", and their order is arbitrary. A form is stacked to put the above structure into the main sentence structure in the correct location.

Next the p-parse is checked for any unused phrases. None are found, and the program terminates by placing the two forms into the structure, which is returned as the "meaning" of the sentence:

```
<==>
    N MAN
        NUMBER SG
        <-DEFINITE- THE
        <-LOC- BESIDE
            N WINDOW
                NUMBER SG
                <-DEFINITE- THE
    PAST
    <-- DO
    <-CAUSE-
        <==>
            N PIANO
                NUMBER SG
                <-DEFINITE- THE
            PAST
            <-- EMIT
                NP
                    N SOUND
    <-BENEFICIARY-
        NPR MARY
```

A gloss of this structure might be: the man, who has location "beside the window", in the past did something which caused the piano to emit sound. The beneficiary of his action was Mary.

4. Other Examples

A few examples of sentences handled by the system are given here. Space constraints do not allow us to include parses, for all the sentences but the remainder are in Taylor[8].

The man with the wife who is bigger than he goes to Vienna with a woman who is smaller than he.

```
>    PARSE:
>    S
>         NP NIL
>              DET THE
>              N MAN
>                   NUMBER SG
>         PP NIL WITH
>              NP NIL
>                   DET THE
>                   N WIFE
>                        NUMBER SG
>         REL NIL
>              RELPRO WHO
>              VP NIL
>                   TNS
>                        PRESENT
>                        VOICE ACTIVE
>                   V BE
>              <-ADJ- NIL BIG
>                   COMP-SUP COMPARATIVE
>         THAN-PH NIL
>              NP NIL
>                   PRO HE
>         VP NIL
>              TNS
>                   PRESENT
>                   VOICE ACTIVE
>              V GO
>         PP NIL TO
```

```
>           NP NIL
>               NPR VIENNA
>       PP NIL WITH
>           NP NIL
>               DET A
>               N WOMAN
>                   NUMBER SG
>       REL NIL
>           RELPRO WHO
>           VP NIL
>               TNS
>                   PRESENT
>                   VOICE ACTIVE
>               V BE
>           <-ADJ- NIL SMALL
>               COMP-SUP COMPARATIVE
>       THAN-PH NIL
>           NP NIL
>               PRO HE
>   BY HE I ASSUME YOU MEAN THE MAN
>   IS THAT CORRECT?
* Y
>   <==>
>       N MAN
>           NUMBER,SG
>           <-DEFINITE- THE
>           <-DESC- WITH
>               N WIFE
>                   NUMBER SG
>                   <-DEFINITE- THE
>                   <==>
>                       N WIFE
>                           NUMBER SG
>                           <-DEFINITE- THE
>                       PRESENT
>                       <-- HAVE-PROP
>                           BIG
>                               <-COMPARED-TO-
>                                   N MAN
>                                       NUMBER SG
>                                       <-DEFINITE- THE
>       PRESENT
>       <-- MOVE
>       <-SOURCE- SOMEPLACE
>       <-DESTINATION-
>           NPR VIENNA
>       <-CO-AGENT-
>           N WOMAN
```

```
>                   NUMBER SG
>                   <-INDEFINITE- A
>                   <==>
>                       N WOMAN
>                           NUMBER SG
>                           <-INDEFINITE- A
>                       PRESENT
>                       <-- HAVE-PROP
>                           SMALL
>                               <-COMPARED-TO-
>                                   N MAN
>                                       NUMBER SG
>                                       <-DEFINITE- THE
>
>
```

Fred loved the old woman before he came to Canada.

Many cases can appear as embedded sentences as well as prepositional phrases. Pronoun references within sentences can be resolved.

Fred played Jack tennis.

Some verbs allow the co-agent case to appear in this form.

The music played loudly from the small room.

```
>     PARSE:
>     S
>         NP NIL
>             DET THE
>             N MUSIC
```

```
>                 NUMBER SG
>         VP NIL
>             TNS
>                 PAST
>                 VOICE ACTIVE
>             V PLAY
>         <-ADV- NIL LOUD
>         PP NIL FROM
>             NP NIL
>                 DET THE
>                 N ROOM
>                     NUMBER SG
>                     <-ADJ- SMALL
>     <==> SOMEONE
>         PAST
>         <-- PLAY
>             N MUSIC
>                 NUMBER SG
>                 <-DEFINITE- THE
>         <-SOURCE-
>             N ROOM
>                 NUMBER SG
>                 <-ADJ- SMALL
>                 <-DEFINITE- THE
>         <-ADV- LOUD
>
>
```

It is idiotic that Fred went to India to play football.

```
>     PARSE:
>     S
>         NP NIL
>             PRO IT
>                 SUBJ
>                 OBJ
>                 NUMBER SG
>         VP NIL
>             TNS
>                 PRESENT
>                 VOICE ACTIVE
>             V BE
>         <-ADJ- NIL IDIOTIC
```

```
>        THAT-COMP NIL
>            <==>
>                NPR FRED
>                PAST
>                <-- MOVE
>                <-SOURCE- SOMEPLACE
>                <-DESTINATION-
>                    NPR INDIA
>                <-PURPOSE-
>                    <==>
>                        NPR FRED
>                        PRESENT
>                        <-- PLAY
>                            N FOOTBALL
>                                NUMBER SG
>    <==>
>        <==>
>            NPR FRED
>            PAST
>            <-- MOVE
>            <-SOURCE- SOMEPLACE
>            <-DESTINATION-
>                NPR INDIA
>            <-PURPOSE-
>                <==>
>                    NPR FRED
>                    PRESENT
>                    <-- PLAY
>                        N FOOTBALL
>                            NUMBER SG
>        PRESENT
>        <-- HAVE-PROP
>            IDIOTIC
>
>
```

There is a small pen in that box.

The "there is" construct is a special case of "to be."

The house with the piano in it was given to Fred by his wife.

```
>    PARSE:
>    S
>        NP NIL
>            DET THE
>            N HOUSE
>                NUMBER SG
>        PP NIL WITH
>            NP NIL
>                DET THE
>                N PIANO
>                    NUMBER SG
>        PP NIL IN
>            NP NIL
>                PRO IT
>                    SUBJ
>                    OBJ
>                    NUMBER SG
>        VP NIL
>            TNS
>                PAST
>                VOICE PASSIVE
>            V GIVE
>        PP NIL TO
>            NP NIL
>                NPR FRED
>        PP NIL BY
>            NP NIL
>                DET
>                    POSSPRO HIS
>                N WIFE
>                    NUMBER SG
>    BY HIS I ASSUME YOU MEAN FRED
>    IS THAT CORRECT?
* Y
>    <==>
>        N WIFE
>            NUMBER SG
>            <-POSS-BY- FRED
>        PAST
>        <-- TRANSFER
>            N HOUSE
>                NUMBER SG
>                <-DEFINITE- THE
>                <==>
>                    N HOUSE
>                        NUMBER SG
>                        <-DEFINITE- THE
>                    PRESENT
```

```
>                         <-- CONTAIN
>                             N PIANO
>                                 NUMBER SG
>                                 <-DEFINITE- THE
>         <-RECIPIENT-
>             NPR FRED
>         <-SOURCE-
>             N WIFE
>                 NUMBER SG
>                 <-POSS-BY- FRED
>
>
```

5. Anaphoric References

Anaphoric references are resolved in the case analysis part of the system. As the system is developed around a specific domain or data base, these routines will be modified to give them more power. Currently they work solely by looking at the previous sentences.

Resolution of anaphoric references fits very well into a case system. Since a pronoun is only encountered in a search for a particular case, this gives the anaphoric routines a great deal of information about what kind of referent to look for. Here we will give just a brief outline of a fairly intricate procedure.

When a pronoun is found in the sentence, it triggers a call to the function ANAPHORIC. ANAPHORIC takes four arguments:

1. A list of cases to look for.
2. A test that the referent must pass.
3. A number indicating how far back in the history to look.
4. The pronoun referenced.

The search is breadth first, in that the program tries very hard to find the referent in the earliest possible sentence. The test is an arbitrary form. SHOULD-BE and MUST-BE elements of the test are shut off on failure as they are in the rest of the back-up procedure.

Say, for instance, that the system is given the sentence:

(12) He played the piano.

The call to AGENT would be the form:
(AGENT (AND (SHOULD-BE MUSICIAN) (MUST-BE HUMAN))). Since the obvious candidate for the agent is a pronoun, ANAPHORIC would be invoked. Its TEST would be:

(AND (SHOULD-BE MUSICIAN) (MUST-BE HUMAN)).

ANAPHORIC would look back through the parses and p-parses of the recent sentences which are kept as global variables, looking for a noun phrase that will pass this test. As it becomes more and more desperate it will make the test less strict. Since "he" is the pronoun, ANAPHORIC is smart enough to insist that the referent be male.

Most pronoun references within a sentence itself can also be resolved. For instance:

(13) <u>Fred</u> went to London so <u>he</u> could visit the queen.
(14) <u>Jack</u> took you up in <u>his</u> airplane.

References to events and places can also be handled:

(15) It was unfortunate that the children were killed.

(16) I went to France. Fred lives there.

The resolution of locational references ("here" and "there") is a difficult problem. By treating "there" as a pronoun whose referent must be a location, "there" is handled fairly well by the system. "Here" is much more difficult, since its resolution is highly context dependent.

Another difficult problem is illustrated by sentence (17).

(17) Mary was aboard the Titanic when she sank.

This sentence is ambiguous: Mary could have sunk in a swimming pool while she was on the Titanic, but this is probably not the intended meaning. If "to sink" is defined with a test like

(SHOULD-BE BOAT)

then the system will pick up "Titanic" correctly. Its first choice as candidate is "Mary", however; thus if the test does not rule "Mary" out, the system will choose her as its initial guess.

This illustrates a difficulty with the current system's anaphoric routines. The first candidate found which passes the test is chosen, rather than all of the candidates being looked at, and the most likely accepted.

6. Conclusion

In summary, then, what we have implemented is a powerful parser for English sentences. It employs case frames to discover the intended meaning of the verb, then continues to use case in its analysis of the rest of the sentence. Each case has one or more tests associated with it, and each verb can add further tests to the cases in its case frames. These tests are gradually weakened on failure, giving the careful user complete control over the back-up.

The system is carefully structured to allow easy extension or modification. As more world knowledge is added to the system, the tests on the cases, and in the case frames can be made to employ this knowledge, thus making them more selective.

The structure building routines are completely general, allowing the user to return any structure he desires within the constraints of the general knowledge he puts into the system.

We feel that this system illustrates the simplicity, flexibility, and expressive power of case in applications in computational linguistics.

BIBLIOGRAPHY

1. Bruce, Bertram. "Case Systems for Natural Language." Computer Science Department, Rutgers, CBM-TT-31, 1974

2. Celce-Murcia, M. "Paradigms for Sentence Recognition", in System Developement Corp. Final Report No. HRT-15092/7907. 1972

3. Fillmore, Charles. "The Case for Case" in Bach and Harms(eds.), Universals in Liquisitic Theory. New York: Holt, Rinehart, and Winston, 1968. 1-90.

4. Martin, William A. "Translation of English into MAPL Using Winograd's Syntax, State Transition Networks, and a Semantic Case Grammar" Automatic Programming Group Internal Memo 11, MIT Project MAC, 1973

5. Schank, R. C. "Conceptual Dependancy: A Theory of Natural Language Understanding" Cognitive Psychology 3,4 (1972) 552-631

6. Schank R. C., "Identification of Conceptualizations Underlying Natural Language" in Schank and Colby(eds.), Conputer Models of Thought and Language. San Fransico: W. H. Freeman & Co. 1973. 187-247

7. Simmons, R. F. "Semantic Networks: Their Computation and Use for Understanding English Sentences" in Schank and Colby 1973, 63-113

8. Taylor, Brock H. "A Case-Driven Parser", unpublished Masters thesis, University of British Columbia, Vancouver, 1975

9. Winograd, T. Understanding Natural Language. New York: Academic press, 1972

10. Winograd, Terry "A Procedural Model of Language Understanding" in Schank and Colby, 1973, 152-186

11. Woods, W. A. "Transition Network Grammars for Natural Language Analysis" Comm. ACM Vol 13 (Oct. 1970) 591-606

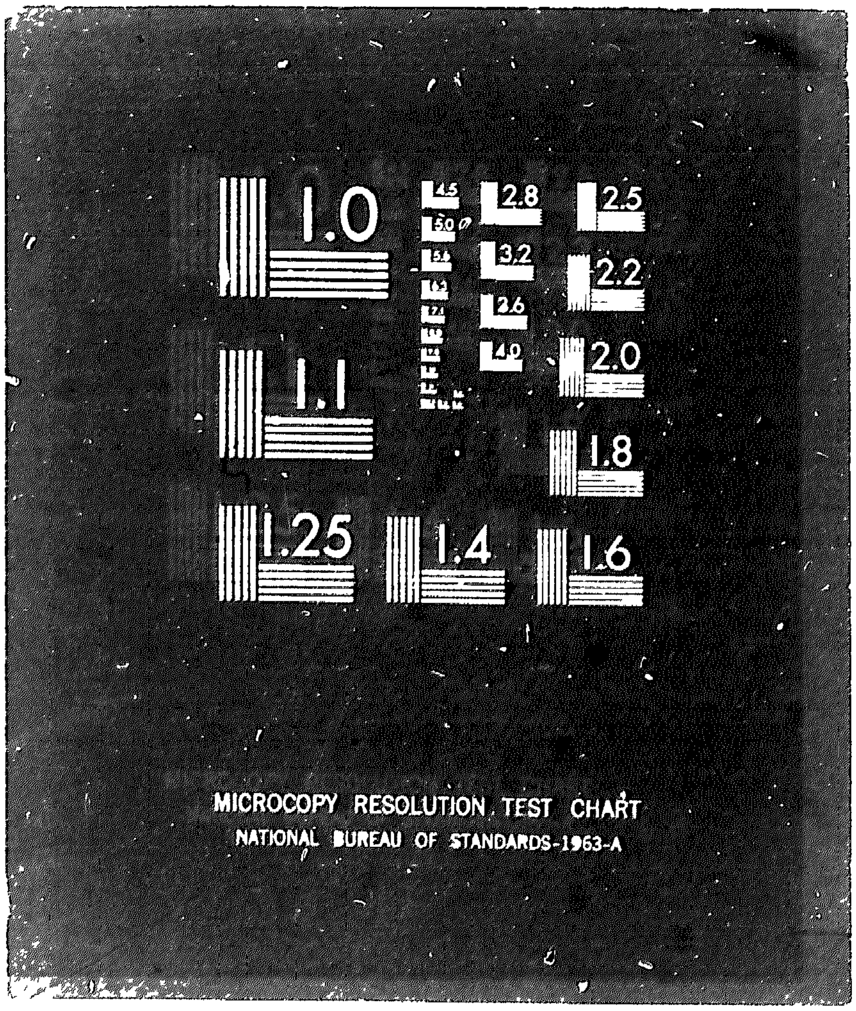
12. Woods, W. A. "An Experimental Parsing System for Transition Network Grammars", in Rustin (ed.) Natural Language Processing New York: Algorithmics Press, 1973. 111-154

<u>Appendix</u>: The Dictionary Entry for "to play"

```
(PLAY V
      S-ED
      V-MEAN
      (IF ((AGENT (AND (SHOULD-BE MUSICIAN) (MUST-BE HUMAN)))
            AG
            (OPT (GETR PASSIVE) 'SOMEONE)
            (PATIENT (MUST-BE MUSICAL-INSTRUMENT))
            PA
            (COND ((AND (NOT FAIL-TEST)
                        (DEFAULT 'PATIENT (NOUN-GET (GETR AG)))))
                  (T 'OBL)))
          (BUILDQ ("<==>" ? "+" ("<--" DO)
                                ("<-CAUSE-"
                                 ("<==>" ? +
                                  ("<--" EMIT (NP (N SOUND))))))
                  AG TNS PA TNS)
          ((AGENT (AND (SHOULD-BE MUSICIAN) (MUST-BE HUMAN)))
            AG
            (OPT (GETR PASSIVE) 'SOMEONE)
            (PATIENT (MUST-BE MUSIC))
            PA
            OBL)
          (BUILDQ ("<==>" ? "+" ("<--" PLAY ?)) AG TNS PA)
          ((AGENT (AND (MUST-BE HUMAN) (SHOULD-BE SPORTS-MAN)))
            AG
            (OPT (GETR PASSIVE) 'SOMEONE)
            (IND-OBJ (MUST-BE HUMAN))
            CO-A
            OPT
            (PATIENT (MUST-BE SPORT))
            PA
            (COND ((NOT FAIL-TEST)
                   (DEFAULT 'PATIENT (NOUN-GET (GETR AG))))
                  (T 'OBL)))
          (BUILDQ (@ ("<==>") (?) ("+") (("<--" PLAY ?)) #)
                  AG TNS PA
                  (PROG (TEMP)
                        (RETURN
                         (COND ((SETQ TEMP (GETR CO-A))
                                (LIST (LIST '"<-CO-AGENT-"
                                            (SOFT-NOUN-LIST-GET
                                             (NP-BUILD TEMP))))))))
          ((AGENT (MUST-BE MUSIC)) PA OBL)
          (BUILDQ ("<==>" SOMEONE + ("<--" PLAY ?)) TNS PA)
          ((AGENT (MUST-BE ANIMATE)) AG OBL)
          (BUILDQ ("<==>" ?
```

```
         "+"
         ("<--" DO)
         ("<-CAUSE-"  ("<==>" ? "+"
                       ("<--" HAVE-PROP
                              ENTERTAINED))))
AG TNS AG TNS)))
```

END


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A