American Journal of Computational Linguistics Microfiche 75

NEWSLETTER OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS

VOLUME 15 - NUMBER 2 JUNE 1978

AMERICAN JOURNAL OF COMPUTATIONAL LINGUISTICS is published by the Association for Computational Linguistics

SECRETARY-TREASURER Donald E Walker, SRI International, Menlo Park, California 94025

EDITOR David G Hays, 5048 Lakeshore Road, Hamburg, New York, 14075

ASSOCIATE EDITOR George E Heidorn, IBM Research Center, P O Box 218, Yorktown Heights, New York 10598

EDITORIAL ASSISTANT William Benzon

American Journal of Computational Linguistics

CONTENTS

Microfiche 75: 2

American Journal of Computational Linguistics *Microfiche 75: 3*

THE DERIVATION OF ANSWERS FROM LOGICAL FORMS IN A QUESTION ANSWERING SYSTEM

FRED J DAMERAU

IBM Corporation
Thomas J Watson Research Center
Yorktown Heights, New York

ABSTRACT

This paper describes how the process of generating a response given an underlying representation for an input question is accomplished in the Transformational Question Answering (TQA) system under development at IBM Research, a brief description of which is given.

The last formal level of representation in this system is called a logical form. The basic method of evaluation of logical forms is the generate and test" paradigm, used, for example in the LUNAR system (Woods, Kaplan and Nash-Webber, 1972), although the implementation must be fairly efficient in order to be practical on a moderate size data base. The basic idea is to keep track of the equivalence relationships between the variables in the logical form and associated constants, and use this information to derive from the data base the extensions of the predicates contained in the logical form. A similar proposal has been made by Reiter(1976). The logical forms and the process by which candidate sets are computed from these forms are described in considerable detail. We believe it should not be necessary for a computational linguistics project to describe operations beyond the last level of formal representation in order for an outsider to understand exactly how a system operates sufficiently well that he can predict its behavior. Although we have attempted to achieve that, we still have a considerable way to go.

INTRODUCTION

This paper describes how the process of generating a response given an underlying representaion for an input question is accomplished in the Transformational Question Answering (TQA) system under continuing development at IBM Research. TQA has been operational in a laboratory mode for several years. The system is now installed in the office of the planning department of a small city where it is used to access the file of land use for each parcel of land in the city, (about 10,000 parcels with 40 pieces of data for each parcel). The sysytem is undergoing modifications and improvement prior to a formal evaluation stage.

SYSTEM OVERVIEW

A generalized flow diagram of the TQA system is given in Figure 1. Input, from a display device or typewriter-like terminal, is fed to the preprocessor, which segments the input character string anto words and performs lexical lookup. The process of lookup is complicated somewhat by a provision for synonym and phrase replacement. Words like "car" and "automobile" are changed to "auto", and strings like "gas station" are frozen into single lexical units.

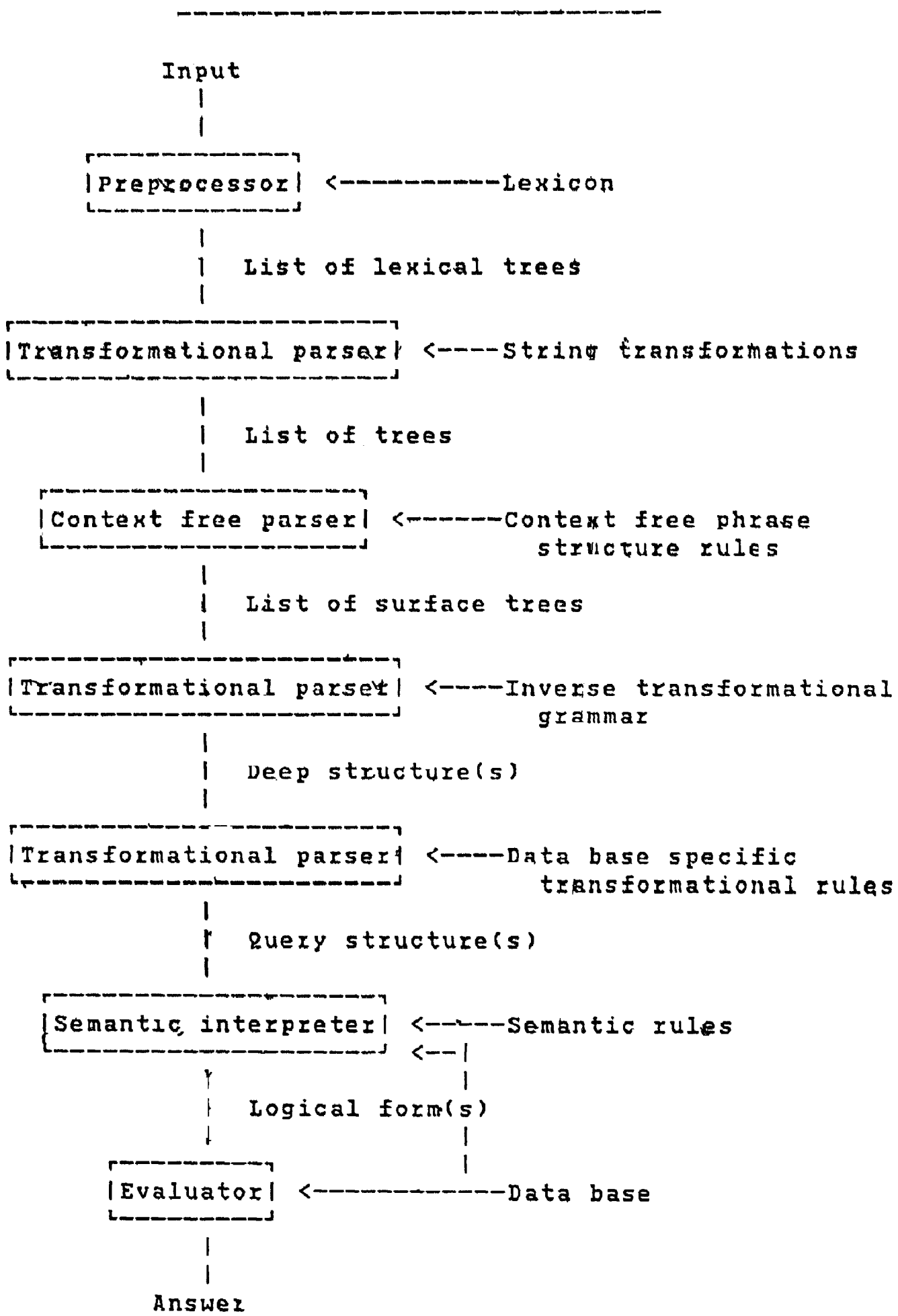

Input
Preprocessor
Lexicon
List of lexical trees
Transformational parser
String transformations
List of trees
Context free parser
Context free phrase structure rules
List of surface trees
Transformational parser
Inverse transformational grammar
Deep structure(s)
Transformational parser
Data base specific transformational rules
Query structure(s)
Semantic interpreter
Semantic rules
Logical form(s)
Evaluator
Data base
Answer

Figure 1

The output from the lexical lookup is a list of trees, each tree containing part of speech information, syntactic features and semantic features, as required. A description of the lexical component, now obsolete in its detail but still valid in main outline is given in Robinson(1973). The list of trees is input to a set of string transformations, described in Plath(1974). These transformations operate on adjacent lexical items to deal with patterns of classifiers, ordinal numbers, stranded prepositions, and the like. The effect of this phase is to reduce the number of surface parses and the amount of work done in the transformational cycle. The resulting list of trees is input to a context free parser, which produces a set of surface trees, each of which is fed to the transformational recognizer.

The recognizer attempts to find an underlying structure for each surface tree, Plath(1973). Typically only one of a set of surface trees will result in an underlying structure. This structure itself is input once again to the transformational recognizer, using a (small) set of grammar rules tailored to a specific data base to produce a query structure. Query structures are similar to underlying structures in form, but reflect the particular meaning constraints resulting from the format and content of a given data base. The query structure tree is processed by a Knuth-style semantic interpreter, Petrick (1977), producing a logical form. A logical form can best be thought of, in

our context, as a retrieval expression which is to be evaluated, producing an answer to the English input query. Since the major part of this paper is concerned with processing logical forms, discussion of their specifics will be deferred until later

The process of answer extraction from the data base is accomplished by a combination of LISP and PL/I programs, described below, and an experimental relational data base management system called Relational Storage System (RSS) (Astrahan, et al. 1976). The RSS provides the capability to generate a data base of n-ary relations, with indexes on any field of the relation, and low-level access commands like OPEN, NEXT, CLOSE, with appropriate parameters, to retrieve information from such a data base.

All the processing modules are under the control of a driver module, which maintains communication with the user, calls the processors in the correct sequence, and tests for errors. An example of the processing of a question, with the intermediate outputs, is given in Figure 2.

In this example, the numbers 2945, 6535, 6635, 6975 are the numbers of milliseconds of computer time used up to the point shown, on an IBM S/370 Model 168. The structures printed are a bracketted terminal string representation of structures which are stored and manipulated as trees by the

```
what are the heights of the drug stores ?

2945 SURFACE STRUCTURES:

   (((WH SOME) (THING X1)) BE (THE ((HEIGHT X4)
     (OF (THE ((DRUG_STORE 591) X7))))) ?)

6535 UNDERLYING STRUCTURES:

1. (BD IDENTICAL (THE (X4 (* BD HEIGHT X4
    (THE ((DRUG_STORE 591) X7)) BD *))) ((WH SOME)
    (THING X1)) BD)

6635 QUERY STRUCTURES:

1. (THE (X4 (* BD HEIGHT X4 (THE ((DRUG_STORE 591)
    X7)) BD *)))

6975 LOGICAL FORM:

(setx 'X4
  '(foratleast 1 'X44
    `(setx 'X7
       '(testfct
         '591
         '('LUC X7 '1976)
         '= ) )
      (testfct
       X4
       '('JSTOR X44 '1976)
       '= ) ) )

7995 ANSWERS:
```

	NUMBER STORIES
1976	
70590001610	1
80100000710	1
80100000811	2
90430000910	1

Figure 2

processing programs. The nonterminal nodes of the tree, together with their associated complex features, represent much additional information that is not shown here. The number 591 is a land use code which, in the data base, indicates a drug store, and the long numbers in the answer are the parcel identifiers, (ward-block-lot).

From this brief description, it should be apparent that the TQA system, considered as a black box, is similar to many others. In particular, there is a designated level of meaning representation, the logical form, which is the last formal construct in the system. The remaining processing necessary to derive an answer and to format it for presentation to a user is accomplished by an unstructured set of computer programs. Two separate issues arise as a result: how efficiently can the logical form be evaluated against a real data base, and to what extent do the processing functions further specify meaning, beyond that carried by the logical form?

EVALUATION OF LOGICAL FORMS

The basic method of evaluation of logical forms is the "generate and test" paradigm used, for example, in the LUNAR

system Woods Kaplan and Nash-Webber, 1972). The simple version of this paradigm, used by Woods and implemented in our early systems, involves checking pre-selected lists of objects or, in the worst case, all the objects known to the system, to see if they satisfy the query predicates. It is computationally impractical except for small data bases. Our current variant of this method is much more efficient. The basic idea is to keep track of the equivalence relationships between the variables in the logical form and associated constants, and use this information to derive the extensions of the predicates contained in the logical form from the data base. A similar proposal has been made by Reiter(1976) We do not however, make such extensive use of query transformations as Reiter outlined.

Logical forms

In order to describe the evaluation process, it is necessary to describe the logical form in somewhat more detail, referring for an example again to Figure 2. In the first place, except for the set-forming function setx, which takes as arguments a variable name and a proposition, all other well-formed formulas are composed of predicates and their arguments. Some of the predicates are perfectly ordinary like greaterthan. Some are quantifiers, like foratleast, which takes a limit argument n, an argument

which is a set, and a proposition p, and which is true just in case n or more elements of the specified set satisfy the proposition p. Others are special application predicates like parcel, which is true just in case its single argument is a parcel identifier.

The main data base related predicate is named testfct. Referring to Figure 2, it is seen that testfct has three arguments. The first is a constant or a variable which will be replaced by a constant before evaluation, the second argument is a list whose members determine a particular data base value, and the third is an operator specifying the relation which must hold between the first argument and the data base value determined by the second argument.

The data base can be thought of as a collection of binary relations, all sharing the same key. In our application, this is the parcel identification or account number, by which any piece of property can be identified. The list which is the second argument of testfct consists of the relation name and the key which identifies a value in the relation. The key actually has two parts. The second part is a year, now unused, although since the files in which we are currently interested are changed on a yearly basis, we anticipate maintaining and accessing historical data. The first part of the key is the account number mentioned above. In general, the second argument of testfct must be

sufficient to identify a unique binary relation and value in that relation.

If the logical form is itself a proposition the system will answer either "yes" or "no". If the logical form has a top level setx, the system will print the members of the set satisfying the specified proposition, perhaps along with some identifying information:

Simplifications

A number of simplifications can be, and in part have been, carried out on logical forms prior to evaluation. Some predicates, for example, are essentially empty for purposes of evaluation, in that they always evaluate to true. As an example, the predicate dollar, for information fields referring to taxes, is empty of meaning because the processor assumes that the contents of the taxes field are always dollars. A slightly less obvious example of a possible simplification can be seen in Figure 2. The set argument of the foratleast predicate contains no free variables. It is not necessary, therefore, to evaluate the inner setx function for each evaluation of the foratleast predicate. Instead, the setx function is evaluated as soon as the semantic interpreter has discovered that it has no free variables, using the standard evaluation mechanism, and the value, i.e., a set, is substituted for the setx

expression. Our system performs simplifications of this kind in its normal mode (although it can also delay all evaluations until a complete form has been built), so that the final logical form seen by the retrieval functions during processing is usually that shown in Figure 3, where the inner setx has been replaced by the satisfying set viz the parcel identifiers of the set of drug stores.[2] After all the applicable simplifications have been done, the resulting form is passed to the evaluation function, EVALU.

The Pre-evaluator

It might seem that since the system has been written in LISP, it would only be necessary to define the appropriate functions and then call the regular LISP evaluator, instead of a special evaluator like EVALU. While this would be possible, the difficulty with such an approach can readily be seen by considering the embedded setx in Figure 2. The desired set of X7s is that set of parcel identifiers fo which the associated land use code is "591". testfct is a predicate which is true for the appropriate X7s, but what is the candidate set of X7s which should be tested? At worst, the system might consider the set of all objects it knows about. As a better choice, the system could infer from the syntax of testfct that the candidates are all members of the set of parcel identifiers, but still there are almost 10,000

```
what are the heights of the drug stores ?

2930 SURFACE STRUCTURES:

1. (((WH SOME) (THING X1)) BE (THE ((HEIGHT X4)
     (OF (THE ((DRUG_STORE 591) X7))))) ?)

6500 UNDERLYING STRUCTURES:

1. (BD IDENTICAL (THE (X4 (* BD HEIGHT X4 (THE
     ((DRUG_STORE 591) X7)) BD *))) ((WH SOME)
     (THING X1)) BD)

6599 QUERY STRUCTURES:

1  (THE (X4 (* BD HEIGHT X4 (THE ((DRUG_STORE 591)
     X7)) BD *)))

7176 LOGICAL FORM:

(setx 'X4
  '(foratleast 1 'X44
     (90430000910 80100000811 80100000710
      70590001610)
     (testfct
      X4
       ('JSTOR X44  1976)
      '= ) ) )

7385 ANSWERS:
```

	NUMBER STORIES
1976	
70590001610	1
80100000710	1
80100000811	2
90430000910	1

Figure 3

of those A much better approach is to attempt to compute the extension of those predicates for which the variable being sought is an argument Again referring to Figure 2, a

reasonable set (in fact the perfect set) of candidates for X7 can be found by looking in the data base for that set of identifiers for which the land use code is 591 If the data base is properly organized, such a search can be very fast Not all predicates are so simple however. The remainder of this section will describe in some detail how candidate sets for more complicated predicates are arrived at. Once candidate sets have been computed the EVALU function can invoke the LISP evaluator on the logical form. The alternative of including a candidate generator in the setx program and all the potential top level predicates and then applying the LISP EVAL function directly seems much less attractive.

As a preliminary, notice that we need only insure that candidate sets have been established for all the setx variables in a logical form. This is so because, while each quantifier has an associated variable, the domain of that quantifier is either given explicitly as a list of constants, or implicitly by a setx expression. Secondly, since the object of pre-evaluation is merely to find efficient, not necessarily optimal, candidate sets for the setx variables, we need not keep track of the structure of a complex predicate. As an example, consider Figure 4, which is the logical form for the question,

"What drug stores are located in ward 2?"

The predicate of the setx is "and", but for purposes of

```
(setx 'X2
  '(and
    (testfct
      '591
      '('LUC X2 '1976)
      '= )
    (testfct
      '2
      '('WARD X2 '1976)
      '= ) ) )
```

Figure 4

determining a candidate set we can consider each term of the "and" individually. Evaluation of the form with a given candidate set will ensure that a particular member satisfies both terms of the "and".

Operation of the pre-evaluation function. Pre-evaluation is accomplished by a function EVALUA, which takes a logical form, i.e., a setx expression or a proposition as its argument. It determines the type of form with which it is dealing and calls an appropriate specialist routine If as in the case of the "and" of Figure 4, the logical form being considered contains more than one component form, EVALUA calls itself recursively. Consequently, pre-evaluation is a depth-first, left-to-right process. The function always returns nil, all work being accomplished by changes to global variables. Among these are a LISP variable which

contains a list of all setx variables in the logical form, a LISP variable which lists each query variable for which a value has been found, and its value, and a LISP variable which keeps track of the equality relationships which have been discovered between query variables for which a value is yet to be found.

Operation of the algorithm can be better understood by considering somewhat more complicated examples than those seen previously. When EVALUA is given the logical form of

---

What parcels have an area exceeding 550000
square feet ?

7524 LOGICAL FORM:

```
(setx 'X2
  '(and
    (foratleast 1 'X39
      (setx 'X5
        '(testfct
          X5
          '('PARAREA X2 '1976)
          '= ) )
      '(greaterthan X39 '550000) )
    (parcel X2) ) )
```

Figure 5

Figure 5, it calls the setx specialist, which adds X2 to the (null) list of set variables and the (null) list of query variables, and calls EVALUA with the associated setx predicate, "and". As mentioned, this simply results in two

calls to EVALUA, the first of which causes the quantifier specialist to be invoked. (The second call, when made, will not cause any change to the global lists of candidate values for variables, since a candidate set of all parcel identifiers is not useful for purposes of retrieval.) X39 is added to the list of query variables, and the domain argument of the quantifier is inspected. When this is seen to be an instance of setx rather than a list of constants, two actions are taken. Notice that whatever the domain of X39 is, it is a subset (perhaps not a proper subset) of the domain of X5, i.e., the candidate set for X5 must include at least all of the elements of X39. Further, any restrictions which can be imposed on X39 can also be imposed on X5, since the proposition associated with the quantifier is the one to be satisfied, and any candidate not meeting this criterion would be superfluous. Therefore, we can 1) enter into the list of variable relationships the information that for purposes of the pre-evaluator, X39 and X5 are equivalen and 2) call EVALUA once more with the setx associated with X5 as an argument.

X5 is added to the list of set variables, and reinvocation of EVALUA with the setx predicate causes a call to the specialist for testfct. Since there are two variables in testfct, X5 and X2, for which values are unknown, a call to the data base cannot yet be made. The instance of testfct is placed on a list of pending data base calls,

preceded by the variables which require values. (Each time a value for a variable is found, that list is inspected, and any data base calls which can then be made are executed.) Return is made to the quantifier specialist, which calls EVALUA with the predicate over whose arguments quantification is made, viz., greaterthan.

The specialist for numeric predicates, finding that one argument is a variable and the other a constant, causes a change in the variable list to show that X39 and consequently X5 are greater than 550,000. A value like ">550,000" can be used by the data base component to narrow its search just as well as a constant or list of constants, and is therefore acceptable as the value of a candidate list. These changes to the variable lists cause the list of pending data base calls to be inspected and, since only one variable is now unknown in the stacked testfct, a call to the data base is made for those parcels with an area greater than 550,000 square feet.

The specialist for testfct instructs the data base search routine to return as a value a list corresponding to the remaining variable in the form, i.e., X2. In the present example, that is a list of parcel numbers, viz., those parcels which have an area exceeding 550,000 square feet. This list is then assigned as the value of the candidate set for X2.

The stack of recursive calls to EVALUA will now unwind, until a return is made to the evaluation function EVALU. This function determines that candidate lists for all the setx variables have been found, and creates a new list of variable-candidate set pairs for use by the setx function itself. Finally, EVALU can call the LISP evaluator, with the original logical form as an argument.

The case of negatives. The predicate "not", denoted in our system by not* to distinguish it from the LISP not, presents special problems for the kind of system outlined above. A simple example of the difficulty can be seen in

```
What drug stores are not in traffic
zone 6 ?

5651 LOGICAL FORM:

     (setx 'X3
       '(and
         (not*
           (testfct
             '6
             '('TRAFZ X3 '1976)
             '=) ) )
         (testfct
           '591
           '('LUC X3 '1976)
           '= ) ) )
```

Figure 6

Figure 6, which corresponds to the question

"What drug stores are not located in traffic zone 6?"

and variants thereof. When the testfct specialist is given the first half of the and in this form, along with information that there is a dominating not*, it could in principle generate a data base call, since there is only one unassigned variable. The effect would be the retrieval of all parcel identifiers of parcels not located in traffic zone 6. This is a substantial fraction of the data base, and would require inordinate amounts of time and storage space to handle Notice that the other half of the and will also provide a candidate list for the variable X3, presumably much smaller in size. It appears to be the case, from our so far limited experience, that questions containing only a single negated search clause hardly ever occur. The evaluator therefore puts a testfct call of this type on the stack mentioned earlier, indexed by the variable(s) corresponding to the parcel identifier. When the second half of the and of Figure 6 is processed, and a value found for X3, the deferred testfct will be unstacked, resulting in a data base call, and causing a retrieval based on that list of identifiers rather than on the negated value. This data base search is necessary, since we must find the traffic zones for the parcels contained in the candidate list.

This example is also an illustration of why, as was mentioned above, the logical form as a whole must in general be evaluated by the LISP evaluator. In this case, the candidate set for X3 derived from the second clause of the

and is a superset of the answer set which can only be derived by evaluating the whole conjunction. Some efficiencies could doubtless be gained by skipping evaluation in those cases where it is unnecessary, but that is purely an implementation decision

The not* of Figure 7 presents a different kind of problem

```
How many banks have a height not exceeding
 5 floors ?

7966 LOGICAL FORM:

(setx 'X1
 '(quantity X1
   (setx 'X3
     '(and
       (not*
         (foratleast 1 'X45
           (setx 'X6
             '(testfct
               X6
               '('JSTOR X3 '1976)
               '= )
             '(greaterthan X45 '5) ) )
       (testfct
          611 *
         '('LUC X3 '1976)
         '= ) ) ) ) )
```

Figure 7

---

from the previous example. Firstly, notice that the negative must be passed inside the quantifier since the alternative of finding all buildings greater than 5 stories in height and then getting the complement set with respect

to all buildings is extremely unattractive computationally. In the second place, a search qualifier of "<= 5" does not intuitively seem to be much worse than "> 5", at least in the absence of data base distributional statistics. One might, therefore, generate a search with such a qualifier. Our present system does this, although experience may show that all instances of testfct dominated by not* should be deferred, as are the cases of "¬=", for efficiency reasons.

Other specialists Most of the important specialist routines in the pre-evaluator have already been mentioned. There are a few others which should be noted. One is a generator function which, given a predicate, will produce its extension from a stored list. This feature was heavily used in our early system, which had a small data base, but is currently hardly used at all, though it remains available. In principle, one could, given a predicate like "SCHOOL(X)", generate a list of schools. In the present application, this would not be useful, but might in some other. The sole uses at present are a generator for the predicate RANK, for which a list of numbers from 1 to 100 is produced, and for the predicate YEAR, which produces a list of the numbers 1960 to 1985.

The proposition "(QUANTITY x s)" is true if x is equal to the cardinality of the set s. The associated specialist has the obvious function of determining x when s is an instance

of setx.

Equality between variables can be inferred where the domain of a quantified variable is given by an instance of setx, as was illustrated above. Certain predicates also allow this inference to be made. It is clear that predicates like "EQUAL", "SAMEREF", (for "same reference"), and "IDENTICAL" should belong to this class. Since variables can only refer to individuals, the predicate "MEMBER" also is in this class, e.g., given (MEMBER X3 (SETX ....)), a candidate set for X3 can be derived by evaluating the setx expression.

Further efficiency considerations. It has already been noted that generation from instances of testfct with an operator of "¬=" are deferred until enough information is available to execute the query using a list of parcel identifiers. Some other steps have also been taken to reduce data base access time and subsequent evaluation time For one thing, the semantic interpreter has a preferred ordering for instances of the predicate testfct. For example, the relation "WARD" divides the parcels of the city into 6 classes, while the relation "LUC" (Land Use Code) divides the parcels into several hundred classes. If there is no intrinsic reason for ordering the instances of testfct differently, the one with "LUC" will occur earlier in the logical form, (cf. Figure 4). The pre-evaluation specialist

for testfct makes use of this ordering in two ways. If a variable has been assigned a list of identifiers containing fewer members than some threshold x, (x is currently set to 25, but can easily be changed), then a retrieval will always be made using the list of identifiers rather than by a constant compared to data base values. In Figure 4, the second call to the testfct specialist will look up the ward of the four drug stores instead of finding the hundreds of parcels in ward 2. In some instances, particularly for relations like Land Use Code, this may result in more data base accesses than retrieving a new set of keys depending on value, but the improvement cannot be large. In many other instances, there is a big reduction in accesses.

If the candidate set is larger than 25, retrieval will be made using the constant, but the length of the current candidate list is used to limit the number of accesses. Thus, if the current candidate list is 50, the data base access program will terminate if it finds more than 50 identifiers with the value being used. A re-access is then made using the list of identifiers. Again, this may result in inefficiency in some cases where searches are ended just before normal termination, but it does provide a guarantee against excessively long retrievals.

Any number of other efficiency measures could be adopted, and more may be necessary than we now have. For the moment,

these seem to provide acceptable retrieval times.

The Evaluator

For the most part, evaluation of logical forms is quite straightforward. Hidden semantic effects are discussed in the next section; here we are mainly concerned with computation.

Each instance of setx searches the list of variable-candidate set pairs to find the candidate set associated with its own variable and substitutes the members of the set for the variable one by one into its associated predicate. Those members of the candidate set for which the predicate evaluates to true are placed in the solution set. Operation of the quantifier predicates is similar to that of setx, except that, as in Figure 5, it may be necessary to evaluate an instance of setx to find the domain of the quantification variable.

Evaluation of the other predicates consists simply of applying a corresponding LISP function to the arguments. Sometimes the final logical form to be evaluated bears no obvious relation to the input question, as in Figure 8. The usual reason is that a large amount of evaluation was done

```
Are there more than 25 parcels in the Carhart
 neighborhood ?

36229 LOGICAL FORM:

(greaterthan '176 '25)
```

Figure 8

---

during interpretation. because form contained no free variables. The full logical form corresponding to Figure 8

```
Are there more than 25 parcels in the Carhart neighborhood.?

15986 LOGICAL FORM:

(forall 'X115
  (setx 'X38
    '(quantity
      X38
      (setx 'X34
        '(and
          (testfct
            '9
            '('NEIGH X34 '1976)
            '= )
          (parcel X34) ) ) ) )
   (greaterthan X115 '25) )
```

Figure 9

---

is given in Figure 9.

The evaluation of the predicate testfct is not as obvious as that of the others One of the design goals in the project has been to make it relatively easy to move from one data base to another. As part of that effort, we have attempted to make the LISP programs, as contrasted to the PL/I programs, insensitive to the structure of the data base. Our approach to this has been to define a list structure, essentially nested binary relations, into which the real data structure is mapped. Restructuring is accomplished by the PL/I program which serves as the LISP - RSS interface. At the same time as the PL/I program returns values to the testfct specialist during the pre-evaluation phase, it formats the corresponding data base items into the standard structure and writes them onto a disk file, in effect creating a sub-data base for the particular query. Only the sub-data base is used during evaluation of logical forms, to find values corresponding to keys in the instances of testfct. In addition to isolating the LISP programs from the real data structure, this tactic makes it unnecessary for any programs called by the evaluator to re-access the full data base, with a consequent efficiency gain.

Creation of the standard LISP data base into which the real data is translated has meant that the set of ISP functions has undergone the least modification in our change of data base from business statistics to planning data. Except for improvements made to increase the efficiency of

programs, these routines are almost the same as they were before.

SEMANTIC EFFECTS OF EVALUATION

In principle the processes which will bw used to compute the answer to a query should be obvious at the level of either the query structure or the logical form. We have not, however, been completely successful in accomplishing this. In some cases, we can see how it might be done and have not gotten around to doing it because of more urgent concerns. In other cases, we can see how to do it, but not how to do it efficiently. In a few cases, it is not clear what to do.

Approximation. Consider the sentence and corresponding logical form shown in Figure 10. The precise system meaning of "about" is clearly hidden in the program corresponding to the operator APPROX. In the present implementation, APPROX of x and y is true if:

1) when $y<10$, $x>y-2$ and $x<y+2$,

2) when $10<y<40$, $x>y-3$ and $x<y+3$,

3) when $y\geq 40$, $x>y-.05y$ and $x<y+.05y$.

I.e., 6 and 8 are approximately equal to 7, 14 and 18 are approximately equal to 16, and 951 and 1049 are approximately equal to 1000. Whether this definition is

```
What parcels are assessed at about $ 1000000 ?

6168 LOGICAL FORM:

(setx 'X2
   (and
    (testfct
      '1000000
      '('VNCITY X2 '1976)
      'APPROX )
    (parcel X2) ) )

6373 ANSWERS:
                                    ASSESSMENT-
                                     CITY_$
  1976
   7059000260O                         977,750
   70430000609                       1,000,000
   70400000600                       1,033,425
   70310000602                         955,900
```

Figure 10

satisfactory or not clearly depends on a variety of contextual factors. It should also be clear that the semantic interpreter could produce a logical form in which this meaning was expressed directly. We have chosen to express the meaning in our processing programs primarily for convenience, i.e. it was easiest to do it in this way, and there was no obvious reason to do it elsewhere.

A similar but slightly different example is shown in Figure 11, where the output rather than the input is to be an approximation to the true value. In this instance, a function called FUZZUP is applied to a data base value to

```
About how many square feet do the drug
stores have ?

7227 LOGICAL FORM:

  (setx 'X4
    '(foratleast 1 'X52 '(18570 24814 8440
 5465) '(approx X52 X4)) )

7479 ANSWERS:

                                   PARCEL
                                    AREA-SQ_FT
  1976
   70590001614                        19,000
   80100000714                        25,000
   80100000814                         8,400
   90430000914                         5,500
```

Figure 11

find that number with the maximum number of trailing zeros which satisfies the APPROX relation. The fuzzed value rather than the true value becomes the output.

A more subtle case is illustrated by Figure 12. It seems clear that what is really wanted are those parcels with an area of a million square feet or more, rather than exactly 1,000,000 square feet. If the latter result is wanted, the question is better phrased "exactly 1,000,000", (and must be phrased in this or a similar way in our system.) On the other hand, a value like 1,000,205 seems to imply that exact equality is wanted. This intuition is captured in our system

```
what parcels have an area of
1,000,000 square feet?

8416 LOGICAL FORM:'

(setx 'X2
  '(and
    (foratleast 1 'X45
      (setx 'X5
        '(testfct
          X5
          '(PARAREA X2 '1976)
          '= ) )
      '(equal X45 '1000000) )
    (parcel X2) ) )

8789 ANSWERS:

1:  70880000900
2:  70790000100
3:  70790000100
          . .
22:  80300000101
MORE PARTICULARS DESIRED?
YES OR NO?
yes
EXPLANATIONS TO THE ANSWERS:
FOR 70880000900 MORE - 13590410
FOR 70790000100 MORE - 5977500
FOR 70790000100 MORE - 5583085

FOR 80300000101 ALMOST - 958320
```

Figure 12

by having the testfct predicate inspect its numeric arguments with a function called ROUNDNM, which is true if an argument is a round number, defined in our system to be a number greater than 99 in which at least the rightmost half of its digits are 0. In the case of round numbers, it seems reasonable to give as an answer the identifier of a parcel

whose area is only slightly less than 1,000,000 square feet, as well as greater.. In our implementation, we use the same lower limit as for APPROX, but this may be too low. In order to insure that the answer is correctly understood by the user, the system saves the exact values retrieved and displays them on request, as shown in Figure 12.

Equality of character values. A problem analagous to that of numerical approximations occurs also in comparing character string values. Consider the question and answer pair shown in Figure 13. The contents of the OWNER field

---

```
What parcels does Shell own ?

4244 LOGICAL FORM:

(setx 'X2
  '(and
    (testfct
      'SHELL
      '('OWNER X2 '1976)
      '= )
    (parcel X2) ) )

4432 ANSWERS:
                                  OWNER
  1976
   70600009501          SHELL OIL COMPANY
   80220003300          SHELL OIL CO
```

Figure 13

---

have not been standardized, so that parcels could be owned

by "Shell Oil", "Shell Oil Co.", etc. Fortunately, for names of persons, last names are listed first, so that the strategy of assuming equality if the input argument and the field value match up to a comma or a blank is generally successful. Problems do arise; for example, properties belong both to "The City of ..." and "City of ...", where the left match fails to find all the relevant data items. The opposite situation, i.e., over-generalization, can of

---

```
what parcels does Gluck own ?

4525 LOGICAL FORM:

(setx 'X2
  '(and
    (testfct
      'GLUCK
      '('OWNER X2 '1976)
      '= )
    (parcel X2) ) )

4742 ANSWERS:

                                OWNER

  1976
   90400000100          GLUCK, DE & ORS
   90410000900          GLUCK, CP
```

Figure 14

---

course also occur, cf. Figure 14. In any event, the decision as to what constitutes sameness of reference is buried in computer code in this instance in the PL/I program as well as in the LISP definition of the function

SAMEREF.

Definitions. The extensional definition of most predicates can be derived from the data base. A few predicates are defined by the system code. Examples are RANK and YEAR. which as mentioned above have associated generators An additional example is LASTYEAR which is defined to be the previous year. Many other definitions of this kind have been eliminated in the current version of the system.

Answers. It is not always obvious what constitutes the answer to a question. Consider the example in Figure 15. Both the English question in its literal reading and the logical form would seem to imply that the question would be answered by presenting only the numbers in the right hand column of the table which is actually printed as an answer. Yet it is quite clear that a simple list would generally be useless without the parcel identifiers printed on the left, and indeed that identification would be expected by the person entering such a question. The example of Figure 16

---

what is the gross floor area of the drug stores ?

7245 LOGICAL FORM:

```
(setx 'X4
   (foratleast 1 'X44
     (90430000910 80100000811 80100000710
      70590001610)
     (testfct
      X4
      '('PGFAREA X44 '1976)
      '= ) ) )
```

7465 ANSWERS:

	GROUND-FLOOR AREA-SQ_FT
1976	
70590001610	7,078
80100000710	10,125
80100000811	6,500
90430000910	1,800

Figure 15

---

is less clear. An enumeration of the three wards in which the four drug stores were located might have been a sufficient answer. The answer given would be correct for

"In What ward is each drug store located?"

Moreover, given the question

"What are the wards which have drug stores?"

it is clear that only a list of wards should be the output, and given

"What is the combined floor area of the drug stores?"

only a single number representing the total is the desired

```
In what wards are the drug stores located ?

9403 LOGICAL FORM:

(setx 'X3
  '(foratleast 1 'X64
    '(90430000910 80100000811 80100000710
      70590001610)
    '(testfct
      X3
      '('WARD X64 '1976)
      '= ) ) )

9597 ANSWERS:
                                    WARD
  1976
   70590001610                                2
   80100000710                                3
   80100000811                                3
   90430000910                                5
```

Figure 16

answer. (Our system does not as yet answer this question or its analogues, although this is planned for later in the year.) Since the ambiguity exhibited by the question of Figure 14 is so pervasive in an application of this kind, we have chosen to present a maximally general answer, including identifications, when we are unable to resolve the ambiguity directly. An exchange with the user could be devised to elicit the information for resolution, but would apidly become tedious for questions of this type. For yes/no questions, and for questions in which there is only one object in the answer set, this problem naturally does not

arise, and the appropriate answer is easily produced..

CONCLUSIONS

We have not yet concerned ourselves with adding an English response generator to the TQA system. In the applications envisioned at present, such a capability does not seem to be critical. We are able to manage with short answers from the data base and with canned information and error messages. In spite of this omission, it should also be apparent that our computational component has a considerable amount of linguistic knowledge embedded in it, more than we would like. Whether it is possible to achieve a level of formal representation which would make this unnecessary is still unclear. Moreover, even if it were possible, it is not clear whether such a solution would be efficient enough, or even if it would be more perspicuous than the current system We intend to proceed as far as we are able in this direction, out of conviction that practically useful systems must be easily adaptable to new applications, and that such adaptation is much more difficult when computer code, even high-level computer code, must be changed, rather than tables. This is not to imply that we regard modification of a table whose size is on the order of a grammar as trivial; quite the contrary. Nonetheless, we believe it is easier to change a grammar or

a semantic interpreter expressed in table form than it is to change a special parser or a special interpreter. In essence, we believe it should not be necessary for a computational linguistics project to describe operations beyond the last level of formal representation in order for an outsider to understand exactly how a system operates.

FOOTNOTES

This system was formerly called REQUEST.

The form of Figure 3 is, in fact, subject to another syntactic transformation prior to execution. Normally, foratleast needs to be executed once for each potential value of the setx variable. However, in the case where the quantificational range of foratleast 1 is a constant, repeated evaluation of the quantifier is quite inefficient. Instead, a special retrieval function called MAPFIELD, which can accept a list of arguments, replaces forms like those of Figure 3. In this example the replacement takes the form

(MAPFIELD 'x77 'JSTOR '(5043... ...00) '1976 ')

Although this transformation arises quite often in practice, it is sufficiently non-general that we have not augmented our inventory of logical forms by including MAPFIELD. Instead, we look on it as an implementation measure only.

References

Astrahan, M.M.; Blasgen, M.W.; Chamberlin, D.D.; Eswaran, K.P.; Gray, J.N.; Griffiths, P.P.; King, W.F.; Lorie, R.A.; McJones, J.; Mehl, J.W.; Putzolu, G.R.; Traiger, I.L.; Wade, B.W., Watson, V.(1976). System R: Relational Approach to Database Management. ACM Transactions on Database Systems, Vol. 1, No. 21 June, 1976, pp. 97-137.

Petrick Stanley R.(1977). Semantic Interpretation in the Request System. In in Cõmputational and Mathematical Lingustics, Proceedings of the Internamtional Conference on Computational Linguistics, Pisa, 27/VIII-1/IX 1973, pp. 585-610.

Plath, Warren J.(1973). Transformational Grammar and Transformational Parsing in the Request System. IBM Research Report RC 4396. Thomas J. Watson Research Center, Yorktown Heights, N.Y.

Plath, Warren J.(1974). String Transformations in the REQUEST System American Journal of Computational Linguistics, Microfiche 8.

Reiter, Raymond(1976). Query Optimization for Question-Answering Systems. In: COLING 76, Proceedings.

Robinson, Jane J.(1973). An Inverse Transformational Lexicon. In Natural Language Processing. Randall Rustin, ed. Algorithmics Press, Inc., New York, N.Y., 1973 pp. 43-60.

Woods, W.A.; Kaplan, R.M.; Nash-Webber, B.(1972). The Lunar Sciences Natural Language Information System: Final Report. BBN Report No. 2378. Bolt Beranek and Newman, Inc., Cambridge, Massachusetts. June 15, 1972.

American Journal of Computational Linguistics *Microfiche 75: 43*

ONE MORE STEP TOWARD C O M P U T E R L E X I C O M E T P Y

NICHOLAS V. FINDLER AND SHU-HWA LEE

Department of Computer Science
State University of New York at Buffalo
4226 Ridge Lea Road
Amherst, New York 14226

ABSTRACT

We describe the continuation of an earlier work on the problem of lexical coverage. The objective is to prove experimentally certain mathematical conjectures concerning the relationship between the sizes of the covering and covered sets of words, and the maximum length of dictionary definitions. The data base on which the experiments are carried out has been also extended to the full contents of an existing dictionary of computer terminology. The results of the previous and present work lay the foundations for quantitative studies on lexical valence and its relation to the frequency of usage and other principles of dictionary selection.

Besides the inherent interest in these investigations, the concepts dealt with and the methods of quantifying dictionary variables may eventually lead to more efficient dictionaries with respect to precision, compactness, and computer time and memory needed for processing.

Supported by NSF Grant MCS 76-24278.

INTRODUCTION

First, we shall introduce the problem define some basic terms and provide a brief historical account of past results. In order to render this paper fairly self-sufficient, a brief summary of the previous work, Findler VIII (1974), will also have to be given.

A monolingual dictionary may be considered economical and efficient if a small set of words are used to define a relatively large set of entries. Quantitative information as to what size vocabulary is needed to cover a given number of entries is very scarce and may be characterized by two "data points":

The New Method English Dictionary published by M.P. West and J.G. Endicott in 1961 uses 1,490 self-defined basic words to explain some 18,000 words and 6,000 idioms, i.e. about 24,000 expressions. Thus, the size ratio is 0.062.

Ogden's Basic English, published in 1933, involves 850 English words and 50 "international" words to define 20,000 English words. The ratio of the covering and covered set sizes is 0.045.

The basis of selection was the "usefulness" of the words employed in the definitions, as opposed to the frequency of their occurrence in some standard texts. However, neither this concept nor other principles of selection suggested by other researchers have ever been quantitatively analyzed and made use of. We shall discuss these issues later on.

In order to approach the problem in definite terms, Findler (1970) considered three basic variables:

(i) the covering set, R, of size v_R,

(ii) the covered set, S, of size v_S,

(iii) the maximum definition length, N, such that each word in S can be defined by at most N ordered words from R.

The task was formulated to find

(a) v_R as a function of v_S at different parametric values of N, and

(b) v_R as a function of N at different parametric values of v_S

Calling $\Delta v_R/\Delta v_S$ increment ratio and v_R/v_S size ratio, the following conjectures were made concerning the first task:

(a1) The increment ratio is, in general, less than one.

(a2) The increment ratio, in general, decreases as v_S increases.

(a3) For large constant values or N, v_R approaches a limiting value asymptotically as v_S increases.

(a4) The increment ratio never exceeds the size ratio.

Two points need to be noted in this connection. An exception to rules (a1) and (a2) would occur in a dictionary system, which does not treat polysemous words or homonyms as individual entries, every time a new word with many meanings or homonyms is introduced into the covered set. Second, the cited case is an exception to rule (a1) but not to (a4). When N=1, the covering and the covered sets are of the same size, i.e. both the increment ratio and the size ratio equal one. However, not every word is defined by itself only. If a new word is introduced that al-

ready has a synonym in the covering set, it will be defined by that synonym. In this case, the increment ratio is 0 and the size ratio becomes less than 1. (This will be clear with the description of the data base construction on page 11.)

For the second general task, (b) the following conjectures were also made:

(b1) $\underline{v_R}$ monotonically decreases as $\underline{N}$ increases.

(b2) For any fixed value of $\underline{v_S}$, $\underline{v_R}$ asymptotically approaches a lower limit as $\underline{N}$ increases without bound.

It seems reasonable to state in a qualitative sense that in the process of generating a dictionary smaller $\underline{v_R}$ values mean smaller storage requirements whereas smaller $\underline{N}$ values tend to reduce processing time and output volume. In order to answer the question "What are the optimum values of $\underline{v_R}$ and $\underline{N}$ for a given $\underline{v_S}$ for a certain (family of) computer applications on a machine with a given cost structure?' one has to consider the interrelation of the above three basic variables and to compute three entities: the semantic index (roughly, the number of different meanings) of the elements in the covered set, the lexical valence (roughly the capability of being substituted for another word) of the elements in the covering set, and the frequency of occurence of the elements of both sets. Quantitative investigations of the last three dictionary variables are planned to follow the present, second stage of our study.

THE DATA BASE AND THE PROGRAM

We have extended the data base used in our previous work,

Findler and Viil (1974). The whole contents of the dictionary on computer technology, Chandor (1970), is now included in the present study. Its structure, rather simple and uniform, is described below. First, some general principles of data base construction are outlined.

Every element of the covered set is considered a single lexical item, regardless of the number of words the original dictionary entry consists of. Also, each word is coded as a string of at most 10 characters (containable in one CDC Cyber computer word). The abbreviations are still easy to read with relatively short practice.

Only the dominant meaning of polysemous terms was dealt with. Each entry has thus one meaning and one definition.

Terms in the definitions (elements of the covering set) are also considered lexical items, i.e. even multiword entities appear as a single unit and are represented by at most 10 characters.

The basic vocabulary, that is the covering set, consists of elements that also appear in the covered set. In our particular case, they are non-technical words used to define the technical terms of the computer dictionary. A definite distinction was made between content words and function words (also called operators). The latter were not included in the covering set nor were they counted in determining the length of definitions. Hence, the covering set consists only of content words.

The function words indicate grammatical and logical relationships between the words contributing to the content.

They belong to 11 categories:

1) prepositions, e.g. of, in, to;

2) conjunctions, e.g. and, or, if;

3) the relative pronoun which;

4) combinations of preposition and relative pronoun, e.g. in which, to which, by which;

5) present participles equivalent to a preposition, e.g. using, containing, representing;

6) combinations of participle and preposition, e.g. consisting of, opposed to, applied to;

7) combinations of adjective and preposition, e.g. capable of, exclusive of, equal to;

8) combinations of noun and preposition, e.g. part of, set of, number of;

9) combinations of preposition, noun, and preposition, e.g. in terms of, by means of, in the form of;

10) prepositional phrases associated with a following infinitive, e.g. used to, necessary to, in order to;

11) other frequently used purely functional expressions, e.g. for example, namely, known as.

Actually, the function words were replaced by code numbers in the dictionary. The code numbers were assigned consecutively as the function words were needed during the construction of the data base so that the order is purely random. A complete list of the 121 function words used, together with their code numbers, is given in Table I.

- -

INSERT TABLE I ABOUT HERE

- -

The original definitions were somewhat simplified and standardized. In this process, articles were omitted (many languages do very well without them). On the other hand, implicit relationships were made explicit. Nouns are represented in singular, thus avoiding another dictionary entry for plural or, what would be worse, programming a "grammar". Likewise, finite verb forms are represented in third person plural present indicative active. Avoiding the third person singular eliminates another dictionary entry, and avoiding the passive voice eliminates a great many participles, which otherwise would have had to be entered. Of course, present and past participles (the former identical to gerund in form) could not always be avoided and had to be entered in the dictionary where needed. Auxiliary verbs were automatically eliminated by avoiding compound tenses and the passive voice. Finally, "to do" associated with negation was simply omitted.

Some examples will make the encoding process clear.

Original dictionary entry:

<u>aberration</u> A defect in the electronic lens system of a cathode ray tube.

Definition in the data base:

DEFECT (in) SYSTEM (of) ELECTRONIC LENS (of) CATHRAYTUB

1. is equivalent to
2. of
3. in
4. in terms of
5. using
6. and
7. which
8. in which
9. between
10. to
11. or
12. from
13. used to
14. necessary to
15. part of
16. consisting of
17. containing
18. capable of
19. by means of
20. opposed to
21. when
22. on
23. so that
24. in order to
25. exclusive of
26. for
27. pertaining to
28. under
29. as
30. such as
62. if
63. among
64. by
65. namely
66. related to
67. concerned with
68. based on
69. constituting
70. resulting from
71. set of
72. including
73. followed by
74. provided by
75. developed by
76. assigned to
77. referred to
78. on which
79. used as
80. in the form of
81. from which
82. into which
83. number of
84. less
85. defining
86. known as
87. performing
88. performed by
89. independent of
90. chosen by
91. for which

31. equal to
32. into
33. with
34. according to
35. applied to
36. depending on
37. to which
38. whose
39. obtained by
40. inherent in
41. through
42. during
43. where
44. during which
45. out of
46. at
47. by which
48. used in
49. without
50. caused by
51. over
52. not
53. but
54. extended to
55. so as to
56. for example
57. represented by
58. along which
59. representing
60. against which
61. similar to
92. at which
93. whether
94. used by
95. about
96. before
97. per
98. having
99. formed by
100. around
101. after
102. since
103. against
104. until
105. whereupon
106. except
107. determined by
108. over which
109. in relation to
110. belonging to
111. corresponding to
112. due to
113. required for
114. type of
115. across
116. because
117. designed
118. indicating
119. produced by
120. outside
121. towards

TABLE I
List of Function Words

Note that "electronic lens system" (should be: electronic-lens system) means "system of electronic lens" (as opposed to "electronic system of lens"), and this relationship is made explicit. Note also that "cathode ray tube" is a single lexical item.

Original dictionary entry:

absolute coding Program instructions which have been written in absolute code, and do not require further processing before being intelligible to the computer.

Data-base entry: ABSOCODING

Definition:

PROGRAM INSTRUCTIO (which) ONE WRITE (in) ABSOLUCODE (and which not) REQUIRE FURTHER PROCESSING (before) INTELIGIBL (to) COMPUTER

Note that the first predicate in the relative clause, third person plural perfect indicative passive, is represented by the singular indefinite pronoun "one" as subject, followed by the standard plural active verb. The auxiliary "do" has been omitted and the negation is represented by a function word. The virtually redundant "being" has also been left out. In general, the copula is omitted (some languages do very well without it).

Original dictionary entry:

analytical function generator A function generator in which the function is a physical law. Also known as natural law function generator, natural function generator.

Data-base entry: ANLYTFNGEN

Definition:

FUNCGENRTR (in which) FUNCTION PHYSICAL LAW

Note also the omission of the gloss "Also known as . . ."

The stylized definitions are easily understandable even to human readers as the printout of the dictionary demonstrates.

The data base was constructed by selecting the first entry, then entering all the lexical items in its definition, subsequently entering all the lexical items in the definitions of these, etc. Words that were not defined in the original dictionary were entered and defined by themselves; they constitute the basic vocabulary. This procedure was continued until everything was defined, i.e. until all the terms in the covering set were also in the covered set. Then the next entry was selected from the dictionary, and the above process was repeated.

The dictionary was arranged in the form of a SLIP list, Findler et al. (1971). Every entry (element of the covered set) occupies four cells in this list: (1) entry word (as character data, using FORTRAN format specification A10), (2) definition length (an integer), (3) type of entry (an integer), (4) sublist name.

Three types of entries were distinguished for programming convenience:

1) code 0 indicates that the entry itself is not used in any definition i.e. it occurs only in the covered set and not in the covering set;

2) code 1 indicates that the entry occurs in both sets and is not an element of the basic vocabulary;

3) code 2 indicates that the entry is defined by itself, i.e. it belongs to the basic vocabulary.

The sublist the name of which is in the fourth cell for every entry in the main list, contains the definition. This arrangement conveniently separates the entry words from those in the definitions.

A cell in this second level contains either a word (in A10 format), i.e. an element of the covering set, or a sublist name. The codes for function words (integers) are contained in the cells in the third level. This arrangement is convenient for bypassing the function words in processing when they are not needed. The general dictionary entry and an example thereof are illustrated in Figure 1.

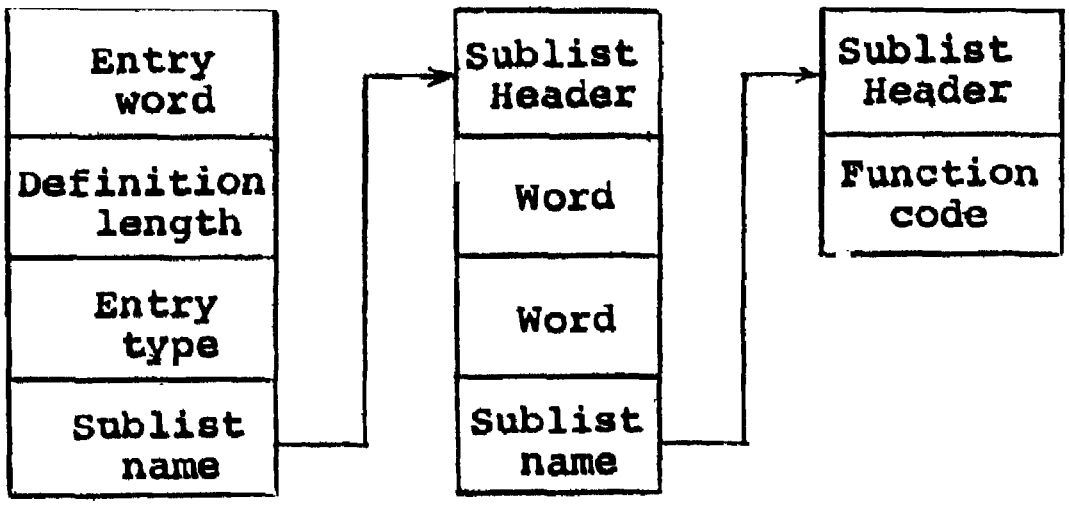
- -

INSERT FIGURE 1 ABOUT HERE

- -

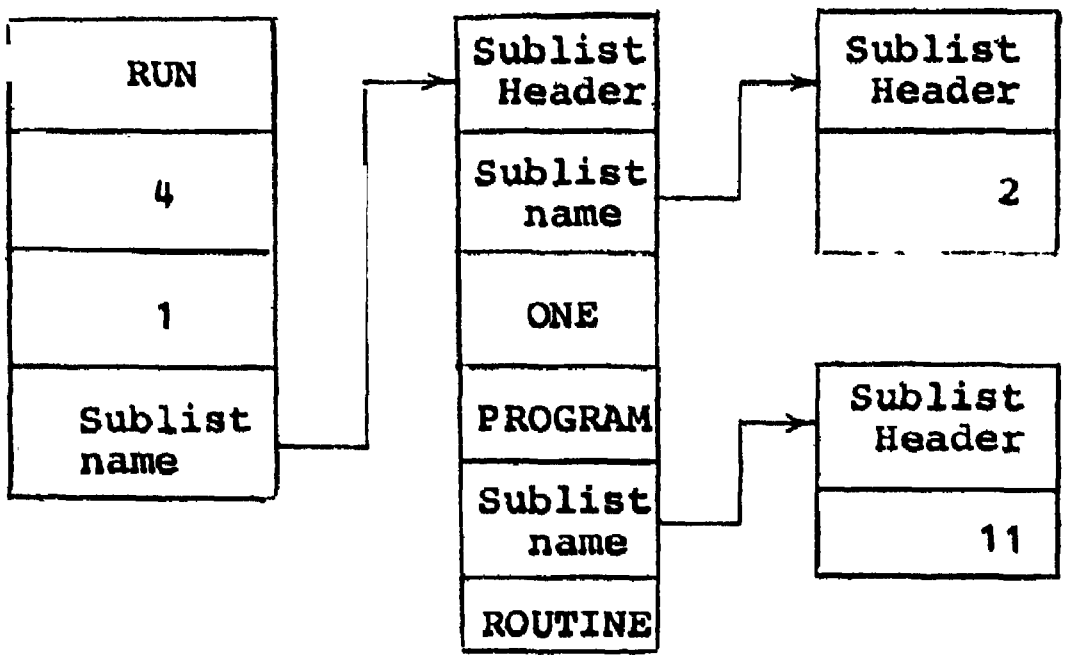
The fact that every dictionary entry owns a sublist is practical in another respect: useful information about the entry can be collected and deposited in a description list associated with the sublist. For example, if it were desired to evaluate the definition component of the lexical valence of each lexical item, a program could be developed that counts how many times a particular item occurs in the definition of other items and stores this information in the description list created for that item. Investigations of this nature will be done subsequently.

The task is to establish experimentally the relationship between $\underline{N}$ and $\underline{v_R}$ for fixed values of $\underline{v_S}$. The program starts out with the values of some fixed data point obtained in the previous



Data Structure for a Dictionary Entry

FIGURE 1a



An Exemplary Dictionary Entry

RUN =: PERFORMANCE of (=2) ONE PROGRAM or (=11) ROUTINE
Definition length: 4; Entry type: 1.

FIGURE 1b

study, Findler and Viil (1974), or one calculated for the extended data base The size of the covering set, v_R is then systematically reduced. Code 1 type words are replaced by their definitions of length 1, 2, 3, 4, etc. (Recall, code 1 means that such entries are not defined by themselves and occur both in the covering and the covered set.) After the substitutions are made in all definitions and the words are counted out of v_k, the corresponding N values are ascertained. The process is repeated for different size covered sets, i.e. v_S is kept at different constant levels, for each run. (We note that a quantitatively more satisfactory refinement could have been added to the dictionary-reduction program. Each basic word would be compared with all the remaining definitions, and those which do not appear in any definition are to be eliminated. Thus a basic word would occur in the dictionary only if it is needed in a definition, which is the case in the unreduced dictionary. This way, a more natural proportion between the basic words and others could be restored. However, in the present preliminary work, we did not wish to pay the considerably higher price for such refinement.)

The program is very complex for two basic reasons. First, the definitions of words to be replaced may themselves contain one or more words to be replaced. Therefore, as many as necessary iterations of replacement have to be carried out in the process. Second, the huge data base representing the whole dictionary had to be subdivided into 9 files only one of which can be dealt with by the program at a time. The intermediate results of one run have to be transferred to the subsequent run, which requires some tricky programming. A brief description of

the multi-file handling is given in the Appendix.

Figure 2, summarizes the results for four different levels of the covered set. Although the procedure followed (leaving one and then two files out of the nine, and adjusting for the bias introduced) leads to quantitative inaccuracies, the conjectures listed in the Introduction are fully corroborated.

- -

INSERT FIGURE 2 ABOUT HERE

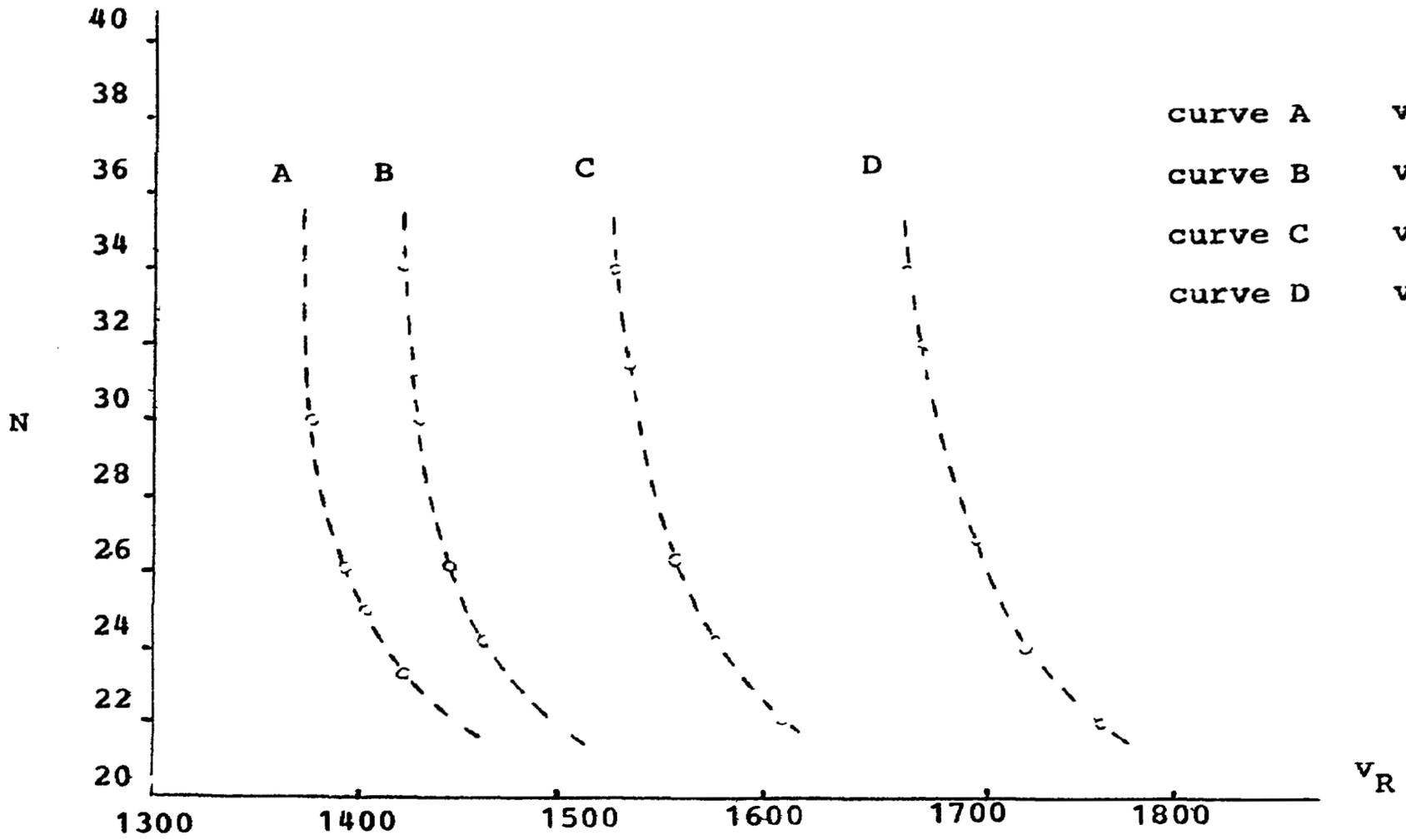
- -

FINAL COMMENTS

The data base encoded, some of the programs used and, most of all, the experience gained in dealing with dictionaries and their characteristic variables will be useful in attacking the next set of problems. The latter relate to the question on what size vocabulary is needed to cover a given number of dictionary entries (without the ubiquitous circular definitions). The answer should be given as a function of storage requirements and processing time so that an optimum solution can be obtained for a family of applications on a machine with a given cost structure. Such study will involve the semantic index of the elements of the covered set, the lexical valence of the elements of the covering set, and the frequency of occurrence of the elements of both sets.

ACKNOWLEDGEMENTS

We thank H. Viil, who co-authored with one of us (N.V.F.) the first phase of this work, for many ideas and stimulating discussions. We are also indebted to Penguin Books for their



Variation of Maximum Definition Length with the Size of Covering Set

FIGURE 2

permission to use one of their publications as our data base

REFERENCES

Chandor, A. (1970). A Dictionary of Computers. Penguin Books. Harmondsworth, England.

Findler, N.V. (1970). Some conjectures in computational linguistics. Linguistics, No. 64, pp. 5-9.

Findler, N.V., J.L. Pfaltz and H.J. Bernstein (1972). Four High-Level Extensions of FORTRAN IV: SLIP, AMPPL-II, TREETRAN and SYMBOLANG. Spartan Books: New York.

Findler, N.V. and H. Vill (1974). A few steps toward computer lexicometry. Am. J. Comp. Ling., 1, 1, 4.

APPENDIX

In the following, we give a brief description of the way multi-file handling has been organized.

It was noted before that the whole dictionary could not be fitted in the core memory at one time and, therefore, the data base had to be subdivided into 9 files to be processed separately. There was a need, however, for some flow of information between runs dealing with the different files. This was arranged by additional files constructed during processing time as well as a few control variable values being read from cards at the beginning of runs subsequent to the first one.

The variable KNTPPT indicates the section of the dictionary currently under study. The variable INCONT is set to 0 for the very first run for each N value. This tells the program to set up new lists for Covered List, Covering List, and so-called Waiting List. In all subsequent runs, its value is 1 which indicates that the program must bring these lists in from an additional, external file.

The program examines the current section of the dictionary, entry by entry. If the entry is an element of the basic vocabulary (type 2), the program bypasses it when it deals with the unreduced dictionary (it is bound to be processed as part of a definition later). Otherwise, this type of word is immediately added to both the Covered List and the Covering List (such word always covers itself), since the definitions in which they occur may have been eliminated.

If a word is not found on the Covered List, it is put there and the appropriate counter is incremented. Then all the words in the definition of the word in question are put on the Waiting List, which is subsequently processed. This is necessary because of the adopted principle that all the covering words must themselves be covered. (Tabulated data are meaningful only if this condition is satisfied.)

The program eventually examines the Waiting List word by word. If the current word is already on the Covered List (it may have occurred earlier in the dictionary), the program checks if it is also on the Covering List (it may not be because it has not yet occurred in the definition of another word). If not, it is put there and the appropriate counter is increased. All words on

the Waiting List come from definitions and must therefore be added to the Covering List. After a word has been processed, it is deleted from the Waiting List (but its processing may have caused new entries to appear on the Waiting List).

If the current word is not on the Covered List, it must, of course, be put there. First, however, the program tests if the word occurs in the section of the dictionary currently in core memory (its "numerical value" is between those of the first and the last word of the section). If the word is not there, its processing is postponed and the next word on the Waiting List is examined because it is more economical to process first all the words available in the dictionary section present than to read in other sections of the dictionary as the words dictate it (memory swapping is expensive).

When the bottom of a non-empty Waiting List is reached, the words remaining there must be in other sections of the dictionary. Subsequent dictionary sections are brought in, to replace the current one, in a cyclic manner until all processing is completed.

COMPUTATION IN DEPARTMENTS OF LINGUISTICS

RICHARD FRITZSON

Department of Linguistics
State University of New York at Buffalo
Buffalo, New York 14261

That computers and linguists meet, for the most part, only in the still somewhat exotic field of computational linguistics is a sad statement about the state of ordinary linguistic research The time when computers were to be considered only the tool of the natural scientist or the statistically minded social scientist is long past, 'word processing technology' is now the specialty of a growing number of computer companies Not only can this technology be of great value in reducing the clerical burden of the linguist and linguistics student, but, linguists, as specialists who have been studying and manipulating language for years, are in a position to be contributing to this field In fact, in many areas of linguistic research the analysis of particular languages, the search for linguistic universals, the analysis of discourse and text, computer technology can be of help to the linguist, and, in many subfields of computer science automated language processing, the design of human/machine interfaces, the structuring of data bases, linguistics has much to offer the computer scientist, yet up until now, relatively few such cross contributions have been made Computer scientists have been slow to discover the value of linguistics to their work, the time has come for linguists to take the initiative and to train themselves (and their students) to make use of and contribute to the field of computer science.

Specialized training in the use of the computer within a particular discipline is not new Students in many social sciences now find themselves facing increasing pressure and mandatory requirements to take computer training within their department, linguistics is, in fact, unusual in not having such requirements or even oppurtunities At a time when graduating linguistics students are facing a shrinking job market, the oppurtunity to be trained in a 'commercially useful application of linguistics ought to be attractive to many students

Today, in most universities, computing is available to linguistics

departments only through the use of a large, central university computer which is expected to be of service to all university departments. But, as computer costs continue to fall, and, as large computing centers continue to be unresponsive to the needs of their new users, it will not be uncommon to find more and more departments purchasing their own computing facilities and buying or developing their own software This is already happening today, both by externally funded individual researchers and by entire departments in need of specialized computing facilities What kinds of computing equipment are available for a linguistics department trying to equip itself today?

My answer is structured, to some extent, by the organization of language It is widely understood, even by non-stratificational linguists, that the faculty of language is based on a stack of structured systems, each one building a large number of units above from a smaller number below, i.e a handful of phonetic features combine to form less than fifty phonemic segments which combine to form thousands of morphemes,tens or hundreds of thousands of words, an infinite number of sentences and texts expressing countless ideas and concepts It will not be surprising to find that as one climbs this stack, from phonology upward, the amount of computing power needed to perform useful tasks and research increases in proportion to the increasing number of units and the complexity of their structuring I will concern myself, mostly, with the possibilities available for the study of the lower levels This is because the type of linguistic work being done in the study of the semantic and cognitive levels is still primarily research and the people involved are more likely to already know their needs and options as far as computing goes Also, since the cost of computing in these areas is somewhat higher, it is less likely that departments will be doing their own purchasing for these purposes

HARDWARE FOR THE PHONOLOGIST

The student of phonology, morphology and linguistic field analysis is concerned primarily with the manipulation of linguistic text, expressed as a series of phonemic symbols or blocks of phonetic features The task is to identify identical or similar substrings, correlate their appearance with a particular meaning, and segment the text into these identified substrings As new substrings are identified, the text is often rewritten with a new organization based on new understandings, so as to improve the chances of finding new substrings, field workers often use index cards for this purpose Problem

after problem is solved in this way, with a not insignificant amount of time being spent in the reorganizing and recopying stages It is a tedious business because it is very mechanical In fact, efficient computer algorithms for doing much of the job already exist and have been implemented on nearly all computers in the form of text editors. The task is relatively simple and even the smallest computer available can do an adequate job

A linguistics department interested in providing its students with training in the use of computers for this kind of work (and they will become standard tools for the purpose very soon) would do well to purchase as many (one or more) identical, small (hobbyist size) computers as it can afford For educational purposes, the very smallest microcomputers, equipped with modest mass storage devices, such as tape cassettes or floppy discs, are just fine Assignments in classes can be distributed on departmentally owned or student owned tapes or discs (less that $10 each) These can be automatically duplicated just as assignments are now mimeographed, they are reuseable and usually contain enough room to store several assignments, including the partial results from day to da and final solutions For larger, research sized projects, involving alot of text, or more complicated analyses, such as automated analysis of phonological tactics, the fastest microcomputers, with larger mass storage devices, might be more appropriate

(Imlicit in the discussion of these types of machines is the fact that student use of them is via an interactive terminal Microcomputers are not typically operated in 'batch mode', and no benefit could be derived from doing linguistic analysis in any but an interactive mode of operation)

While microcomputers and associated memories are relatively inexpensive, linguists have a genuine need for sophisticated input and output devices which are somewhat more expensive Standard computer terminals generally provide all and only the characters available on a typewriter keyboard, some provide only upper case letters What is needed is a terminal with the same capabilities as the selectric style typewriter one with changeable type fonts, including the standard phonetic symbol alphabet, with diacritics CRT terminals (cathode ray tube terminals) can provide this type of operation more cheaply, more reliably, and more flexibly than printing terminals (there is no need to stop and change type fonts) CRT terminals which support user designed type fonts are available, and in fact, may be the only ones on which the standard phonetic alphabet can be currently supplied These terminals are somewhat expensive

(several thousand dollars each), but since they are very flexible and often support some degree of computer graphics display as well as having the potential to display texts written in any language, they are valuable educational tools

If all or most of the terminals in a department are CRT type terminals, it will be necessary to provide some means of producing 'hard copy' output on paper While most interactions with a computer can take place on a screen, some record of the results of a session will be needed for study and evaluation Printers which can handle the flexible type fonts needed by linguists are available They are fast they operate in the same way that copying machines work and simply transfer the contents of the CRT screen to the paper (including graphic materials) They are expensive However, a small department might well find that only one of these printers is necessary to meet their needs, the results of work done on any of the small microcomputers could be moved (either over communication lines or carried on a disc or tape) to the printer with little or no delay

HARDWARE FOR THE GRAMMARIAN

Syntax is, perhaps, the most widely studied subject in linguistics today Given that this is so, there is a real need for linguists, both professional and student, to understand the extreme difficulty of the task of writing a grammar for a language That attempts are made to do this without the aid of a computer is perhaps all the evidence one needs to see that the difficulties are not well understood. A formal grammar, particularly one written in the notations commonly used today, is very much like a computer program It is a list of instructions for generating a list of strings, a computer program is a list of instructions for performing some process (which might be generating a list of strings) Both need to be precise, both are very complex, both suffer from the fact that a change in one part of the ordered list may cause an unanticipated change in the effect of another part It would be very surprising to find that linguists were better at producing untested, yet correct, formal characterizations of complex processes than computer programmers I expect that testing a newly written grammar will be as enlightening an experience for a linguistics student as debugging a new complex program is for a computer science student.

Furthermore, just as the computer is of use in studying phonology and

morphology, it can also offer data organization services to aid in the study of syntax Automated tactic analysis of syntax is still a research project, the software necessary for it is not likely to be produced by a software house. But the research is probably best performed in a linguistics department

Having established a need, we must now recall a warning made earlier Useful contributions to the study of syntax by computers requires more computing power than is needed for similar contributions to the study of phonology and morphology While the need for sophisticated type fonts and input/output devices is lower (not necessarily a good eductional syntax program would permit the manipulation of syntactic trees on a graphics screen), there is a real need for faster processors and increased memory capacity To purchase the necessary computing power, a department would have to step up from the hobbyist microcomputer size machines to the scientific research minicomputer (e g the middle range PDP-11 series) These machines cost an order of magnitude more than the microcomputer and yet, when the subject is syntax, will probably only serve a few students at a time

An alternative, available to some departments, is to use the university's central computing facility. Money could be spent on the best available terminals and the needed communications equipment Grammar testers have been written by university researchers for typical university size computers (Friedman 1971, for transformational grammars, Kehler 1976, for ATN grammars) and are available at little or no cost

As I mentioned in the beginning, the use of the computer in the study of semantics and cognition is still very much a research topic and little, if any of the work being done currently can be performed on small computers I will not describe the requirements of such work since they vary widely depending on the nature of the work.

SOFTWARE FOR THE LINGUIST

What is missing from the computing facilities described so far is software, programs which are of use in solving linguistics problems The small computers are sold with a minimum of very traditional computer software, none of it of any use to the nonprogramming linguist In fact, at no level of computing power is there currently available commercial software which is of use to nonprogramming linguists For large computers, as mentioned above,

some of the results of university research work is available for some purposes However, for the types of machines that departments are likely to purchase, there is essentially nothing

This problem can be overcome in two ways The standard method is for a department to hire a student programmer to design and write the needed software This has several advantages it is relatively cheap (especially when university assistantships are available for the purpose), it is personal - the student can be instructed to write exactly the kind of program that is needed The disadvantages of this method are in the quality and durability of the systems produced in this way Student programmers are, in fact, students learning to program Often their work is lacking in the 'ease-of-use' or 'human engineering' features found in well written, commercially produced programs, and, it is just these features which are very important to users not familiar with or comfortable with computers Furthermore, programs produced by student programmers are not well known for their reliability, maintenance of them is difficult and usually restricted to the period of time that the original programmer is still available Again, to the user unfamiliar with computers, reliability is a very important feature It is very discouraging to try to do anything with semi-operational programs

An alternative is to create sufficient demand for this type of educational software so that a commercial software house or a well funded university programming group would consider the investment of its time and money profitable With linguists and linguistic educators providing input at the design level, very useful and reasonably priced software could be produced in this way The catch, however, lies in generating sufficient demand

A final comment about one other potential use of computers within a linguistics department The search for language universals (cross linguistic research) requires very large collections of information A collection of partial and complete grammars along with sample texts for a large representative sample of human languages is a formidable amount of information The kinds of questions posed by linguists using this information do not require immediate interactive response In fact, they traditionally require weeks or months of library research for answers It is therefore not unreasonable to consider the storage of this information on a small, even hobbyist size, computer equipped with large mass storage devices The task is a difficult one, but

of potential value to both linguists and computer scientists

Linguists need easier access to this information A computerized database, structured according to the needs of linguists, would be a very valuable tool which could be distributed to any department willing to make the necessary investment in hardware The database is large, but unlike many other large databases, it is one about whose structure a great deal is known Computer scientists are still looking for ways to effectively and efficiently organize databases, and linguists, with their intimate knowledge of the structure of language, have an opportunity here to provide an example of how to use the structure of a body of information in storing it on a computer effectively It is a task which requires the expert knowledge of several linguistic disciplines and it is a research project ideally suited to a department of linguistics

American Journal of Computational Linguistics

MANIFESTO THE PRESS AT TWIN WILLOWS

DAVID G HAYS, PUBLISHER
5048 Lakeshore Road
Hamburg, New York 14075

An Idea and a Problem

Contrary to a famous opinion, linearity came in with speech, printing just let us see what we had been saying all the time But thought is nonlinear, and conversation flows as print cannot With electronic publication, we will be able to move through a permanent record of collective knowledge with some of the flexibility that conversation has always allowed

But why a permanent record? Science is forever changing, and so is art Kuhnian revolutions, small or large, are frequent Electronic media do not impose artificial stasis on the flux of ideas that gradually eliminates errors from science and yields pleasure in art

However, none of us have much experience in the new modes of communication Since all need help, we must--in the famous phrase--explain to each other what none of us understand

A Method

THE PRESS at Twin Willows is mostly a method

The method is to use printed paper, familiar to us all, and microfiches, familiar to many, in shifting combination with the unfamiliar electronic media

A computer will be installed in the office of THE PRESS, and used from the beginning for administration and text preparation Editors of books and journals that come to THE PRESS can submit on floppy disk, on casette, or by telephone, but they can also submit on paper Publication can be by photoreproduction, rapid printing, high-quality offset, magnetic recording to drive a computer, or through the telephone net Microfiches will be suggested for many publications

As editors and readers gradually become familiar with the new systems, teaching each other as they learn, we can expect the contents of publications to become more and more suitable to the new media, and less and less suitable to the old

Services

THE PRESS at Twin Willows will offer services at every step from the author's conceptualization through advertising of the finished work.

Editorial. For its clients, THE PRESS will help if necessary to find expert readers who can submit opinions and suggestions about the content of proposed articles and books. THE PRESS will provide counsel on readability. THE PRESS will mark up copy for typographic form, lay out pages, and otherwise give traditional redactory services.

Administrative. For its clients, THE PRESS will maintain tickler files and issue reminders to contributors and readers when their submissions are due. It will prepare budgets and keep accounts. It will maintain mailing lists, membership lists, and consultation lists. It will conduct membership surveys and elections of officers.

Bibliographic As support can be obtained, THE PRESS will draw on existing collections and add its own classifications and subject labels to make bibliography available to clients. Thus the preparation of a bibliography for a work in progress can be assigned to THE PRESS, and a book buyer can follow up references or ask for selective dissemination.

Educational. THE PRESS will shortly begin publication of a newsletter for clients and prospects: Services and how to use them, the competition, new products in hardware and software, publications and courses for authors and editors, and personal notes from the field of electronic publication.

Conferences, workshops, and courses will be organized as the field needs them and can support them.

Handbooks, manuals, and other materials for editors will be written or collected as feasible, catalogued, and offered for sale or gift.

Pricing Policy

Methods and materials will be designed for each client initially; later, a catalogue of components of the publication process will be prepared so that the client can do the design work.

Beyond the direct cost of labor performed and materials consumed at THE PRESS and of services purchased for the client, the equipment used will be billed at a rate intended to give rapid amortization, and a management fee of 15% added.

This policy should bring the cost of information--books, journals, and electronic access--within the limits of anyone's purse.

R E V I E W S: MICRO HARDWARE, SOFTWARE

P U B L I S H E R: THE PRESS AT TWIN WILLOWS

May 23, 1978

To help hobbyists, householders, businesses, and government keep up with the countless vendors who offer hardware and software in the microcomputer market, THE PRESS at Twin Willows will begin immediately to collect and publish evaluative, analytic reviews, according to David G. Hays, Publisher.

"When the computing market was dominated by just a few big companies," Hays says, "it was fairly easy to decide how to handle a computing problem. Once a buyer had settled on a computing budget, the market might offer only two or three main frames big enough and cheap enough to do the job. Now the buyer can design a machine to fit a purpose, and has to choose components out of lists that run up to dozens of alternatives. The worst part is, no one publishes the list!"

THE PRESS intends to correct part of the problem by making useful information about the market available in easy language and inexpensive format. "Before long," Hays expects the hardware and software reviews will be accessible online for clients to dial in."

Where will the reviews come from? THE PRESS invites any user of any microhardware or software to write it up; the

editors at THE PRESS will rewrite if necessary, make sure that the evaluations are not illegally harsh, and eliminate the most obvious errors. No fees are offered to reviewers at present, but a change is contemplated. "Everyone who helps should be paid," as Hays puts it.

Manufacturers and software houses can send their lists and item descriptions to be included with the evaluations.

THE PRESS, which will also publish original material in whatever technical fields need its services, is "mostly a method," Hays says. Its purpose is to teach information users how to cooperate with each other, making central publishing less relevant.

Hays, who is setting up THE PRESS, is a professor of linguistics and of computer science in the State University of New York at Buffalo. He moved to Buffalo from The RAND Corporation in 1968 after 13 years of research on language and computing. Hays is honorary member of the International Committee on Computational Linguistics, editor (1974-78) of the American Journal of Computational Linguistics, and former chairman of NSF's Social Science Advisory Committee.

THE PRESS offers no free literature, but is preparing to issue a Newsletter. A $1 deposit will bring the first few issues, including more about the hardware reviews. THE PRESS is located at Twin Willows, 5048 Lake Shore Road, Hamburg, New York 14075; the telephone number is 716-627-5571.

PUBLISHING AJCL

DAVID G. HAYS
THE PRESS at Twin Willows
May 20, 1978

A letter to: ACL Executive Committee, AJCL Editorial Board

Dear Colleague:

My term as Editor expires, by my definition, at the end of the present calendar year. The Association will choose a new Editor; at the same time, I think that some changes in operations are appropriate.

In the 1960s, I proposed the use of ultramicrofiches for library development; but I said, if I did not write, that photographic storage had a time limit; and the predicted time is now up.

To supplement my University salary, I am organizing The Press at Twin Willows. The enclosure describes the earliest form of the venture; I hope for rapid evolution.

It would be to my commercial advantage to act as publisher for AJCL. I believe that if ACL adopts the word-processing and lexicographic businesses as areas of applied computational linguistics the Association can grow and serve a significant role in improvement of the common weal; and for The Press to help would be very pleasant and profitable.

As Editor, I have contributed the use of space and equipment that I paid for myself, and some small help that the University gave. The new Editor may have more to offer, making The Press redundant; in that case, I should like to open negotiations for secondary publications extracted from AJCL.

The Press cannot offer quite so much; it will be necessary to bill the Association for machine time and personnel costs. But only out-of-pocket costs will appear on invoices if the Association decides to deal with The Press.

As for member services, we can continue microfiches; offer hard copy; move up quickly or slowly to typographic quality; issue newsletters along with quarterly journal; and give online access to computer files. Most of that can be done immediately, but some of it may have to wait a few months.

It is up to the Association to say what it needs, if anything

Sincerely

A S I S: 41ST ANNUAL MEETING

NOVEMBER 13 - 17

NEW YORK CITY

THEME: THE INFORMATION AGE IN PERSPECTIVE

36 Technical Sessions in three general areas:

COLLECTION, GENERATION, AND ANALYSIS OF INFORMATION

DISSEMINATION OF INFORMATION

INFORMATION FOR DECISION-MAKING AND CONTROL

FOR MORE INFORMATION CONTACT: ASIS
1155 16th Street, N.W.
Washington, D.C. 20036
202 - 659-3644

12TH ANNUAL HAWAII INTERNATIONAL CONFERENCE ON SYSTEMS SCIENCES

JANUARY 4 - 5

HONOLULU

SPONSORSHIP: College of Business Administration
Department of Electrical Engineering
Department of Information and Computer Sciences
UNIVERSITY OF HAWAII
Association for Computing Machinery

Sessions on MEDICAL INFORMATION PROCESSING will be included in the conference. For more information contact:

Dr. Bruce D. Shriver or
Dr. Terry M. Walker
HICSS-12/Medical Information Processing
University of Southwestern Louisiana
Box 44330
Lafayette, LA 70504

LINGUISTIC STRUCTURES PROCESSING:

STUDIES IN LINGUISTICS; COMPUTATIONAL LINGUISTICS, AND ARTIFICIAL INTELLIGENCE

ANTONIO ZAMPOLLI, EDITOR
Director of the Linguistics Division, CNUCE
The Institute of the Italian National Research Council (CNR)

FUNDAMENTAL STUDIES IN COMPUTER SCIENCE, Volume 5

NORTH HOLLAND, AMSTERDAM & NEW YORK, XVI + 586 PP., 1977
ISBN 0-444-85017, US $44.95/DFL. 110.00

NATURAL LANGUAGE IN INFORMATION SCIENCE

PERSPECTIVES AND DIRECTIONS FOR RESEARCH

DONALD E. WALKER, HANS KARLGREN, MARTIN KAY, EDS.

SKRIPTOR, Stockholm, Sweden, 1977

FID Publication 551

This book presents the results of a Workshop on Linguistics and Information Science organized by the Committee on Linguistics in Documentation of the International Federation for Documentation (FID/LD) and by the KVAL Institute for Information Science. It contains a series of papers that provide perspectives on linguistics and information science from the vantage points of information science (F. W. Lancaster, University of Illinois), library science (Derek Austin, The British Library), quantitative linguistics (Wolf Moskovich, Hebrew University of Jerusalem), computational linguistics (Naomi Sager, New York University), linguistics (Petr Sgall, Charles University), complex semantic information processing (Teun A. van Dijk, University of Amsterdam), and terminology (J. Goetschalckz, Commission of the European Communities). The book also features a challenge paper on the linguistics of information science (Hans Karlgren, KVAL Institute for Information Science) that delineates major issues in this area. These papers are bracketed by an overview of the Workshop (Donald E. Walker, SRI International) and by a review of the field (Karen Sparck Jones, Cambridge University, and Martin Kay, Xerox Palo Alto Research Center) that updates the book Linguistics in Information Science, a comprehensive survey prepared several years ago by Sparck Jones and Kay under the auspices of FID/LD (Academic Press, New York, 1973).

Natural Language in Information Science will be of interest to specialists in the areas referenced above and to anyone who wants to know more about the potential of natural language processing for information science. The price is $10.00 (U.S.) plus postage and handling. Order as follows:

North and South America

Roberts Information Services
8305-G Merrifield Avenue
Fairfax, Virginia 22030, USA

Europe, Asia, Africa, and Australia

Skriptor
S-104 65 Stockholm 15, Sweden

A J C L: JOURNAL OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS

COMPUTATIONAL LINGUISTICS DEALS WITH SYSTEMS OR COMPONENTS OF SYSTEMS FOR:

RESEARCH ON LANGUAGE	Lexicology Dialectology Grammar Discourse Understanding	Phonology Language Change Semantics Universals
LABORATORY EXPERIMENTATION:	Psychology Sociology	Phonetics Neurophysiology
PRACTICAL APPLICATION:	Translation Instruction Robotics	Documentation Lexicography Speech Recognition
SCHOLARLY INVESTIGATION:	Stylistics Text Comparison	Content Analysis

CONTENT

ORIGINAL CONTRIBUTIONS: Algorithms, programs, system designs, experimental results. theoretical analyses

REVIEWS AND SURVEYS

ANNOUNCEMENTS: Symposia, conferences, publications, courses, grants

ABSTRACTS OF PUBLICATIONS: Wide coverage of journals, books, and technical reports

RESEARCH IN PROGRESS

RESOURCES: A perpetual inventory of files of text, computer programs, dictionaries, grammars, and other materials available to researchers

ADVERTISEMENTS: Announcements of books, equipment, services

FORMAT

The AMERICAN JOURNAL OF COMPUTATIONAL LINGUISTICS is published on 4" by 6" units, each an index card or a microfiche. For each original contribution, two units are supplied: an index card bearing an extended summary, and a microfiche containing full text, illustrations, and related materials. Abstracts, announcements, advertisements, and resources may appear on cards or on microfiche. The microfiche standard is MIC-9, reduction 24x, maximum 98 pages per fiche. Each unite supplied carries at the top a heading characterizing its content. The Journal is mailed in quarterly numbers; 15 to 25 fiche are issued each year.

SUBSCRIPTION

Subscriptions to the AMERICAN JOURNAL OF COMPUTATIONAL LINGUISTICS are available through membership in the ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. For the year 1978, dues for individuals are $15; dues for institutions are $30. A supplementary charge for first class mailing (U.S.) is $2; for foreign subscriptions, the air printed charge is $4. Volumes of the AJCL for 1974, 1975, and 1976 are available at rates of $10 individual and $25 institutional per year; the rates for the 1977 volume are $15 individual and $30 institutional for first class or air delivery, add $2 or $4 per year as appropriate.

Send dues, payable to the ASSOCIATION FOR COMPUTATIONAL LINGUISTICS (or ACL), or requests for information to:

Dr. Donald E. Walker, ACL
SRI International
Menlo Park, California 94025, USA

Alexander D Roth Director
Washington Office

Pender M McCarter Editor

afips Washington Report 79

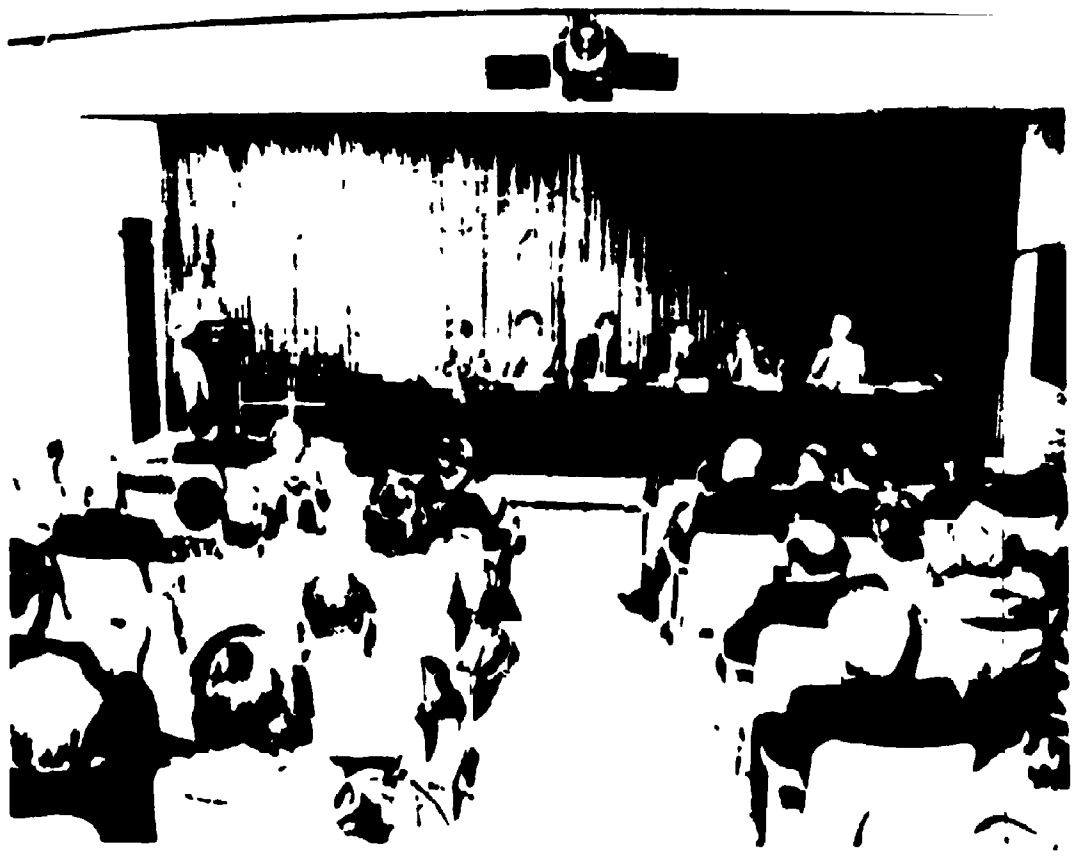
American Federation of Information Processing Societies Inc Washington Office 1815 North Lynn Street Suite 805 Arlington, Virginia 22209 703 243-3000

Vol. IV, No. 6 June, 1978

AFIPS IN WASHINGTON

AFIPS CONVENES CONFERENCE ON WHITE HOUSE, CONGRESSIONAL INFORMATION SYSTEMS; SENIOR GOVERNMENT OFFICIALS ADDRESS AFIPS AUDIENCE

High-level Government officials last month addressed senior members of AFIPS in a special Washington briefing on White House and Congressional information systems. The AFIPS Conference on White House and Congressional Information systems, held May 2nd in the Presidential Press Conference Room of the Old Executive Office Building, was attended by: Mr. Richard Harden, special assistant to the President; Mr. Carl Calo, assistant director for Information Systems, Office of Administration, Executive Office of the President (EOP); Mr. Edward Zimmerman, special assistant to the director, Office of Administration, EOP; Rep. Charlie Rose (D-N.C.), chairman, House Policy Group on Information and Computers; Mr. John Swearingen, director of Information Systems, U.S. Senate; Mr. Neal Gregory, staff director, House Policy Group on Information and Computers; and Mr. Boyd Alexander, director, House Information Systems. About 70 AFIPS' individuals attended the special briefing, including officers, members of the Board of Directors, presidents of the constituent societies, and committee chairmen.



WHITE HOUSE, CONGRESSIONAL OFFICIALS ADDRESS AFIPS AUDIENCE (AFIPS/T.C. White)

Special Assistant to the President Richard Harden told the audience that there are "long-range plans" for developing computer communications between the White House and Capitol Hill. (Previous reports have indicated that such communications could include sharing of budget information.) A minicomputer in every Congressman's office (or at least clusters of minis shared by Congressmen) were possibilities discussed by Congressional participants in the conference.

Opening the White House presentation, Mr. Harden outlined the need for improved information management within the EOP. He noted that the typical Presidential adviser may be considering 30 to 40 issues with five to 10 issues of major importance being considered at a given moment. For each issue, Harden said, several Federal agencies may be involved, as well as a number of Congressional committees, and other groups.

Carl Calo outlined plans for the Executive Office systems as follows:

Access to various nonconfidential systems, both public and private. These might include (1) publicly available information from Executive Branch systems, such as: FAPRS, the Federal Assistance Program Retrieval System, developed by the Department of Agriculture; and the Department of Justice's JURIS, the Justice Retrieval System; (2) Capitol Hill systems, such as: LEGIS, the Legislative Information and Status System; SOPAD, Summary of Proceedings and Debate; and SCORPIO, Subject/Content-Oriented Retriever for Processing Information On-Line; and (3) commercial resources, such as: the New York Times Information Bank, Lockheed's Dialog, and Wharton Econometric Forecasting Associates' (EFA) economic modeling and data services.

A series of information processing utilities available to all users. These might include correspondence control, word processing and text editing, project tracking, and a document filing and retrieval system.

Various special-purpose systems to meet specific needs of individual offices. Examples of present systems are: the Office of Management and Budget's (OMB) Budget Preparation System; the White House's Congressional Vote Analyses System; and the Office of the Vice-President's Time Analysis System.

A request for proposal for development of this system is expected to be issued early in July, following review by the General Services Administration, it was announced at the AFIPS Conference. Until July, Mr. Calo said that a "temporary upgrade" would be accomplished "with little or no increase in present expenditures." He stressed that the upgrade would be replaced at the time of the final procurement. [*Ed.:* At press time, it was learned that Interdata, Inc., Oceanport, New Jersey, performed the temporary upgrade.]

Ed Zimmerman noted that in developing the plans for the White House systems, the EOP has talked with the National Telecommunications and Information Administration (NTIA), the new unit within the Commerce Department; and is considering the recommendations of the Paperwork Commission. Mr. Zimmerman also announced that a demonstration of an advanced communications information retrieval system, accessing demographic information from the Census Bureau, is scheduled in June on Capitol Hill, and in the Old Executive Office Building.

In the question-and-answer session on White House information systems that followed, several individuals in the audience asked how privacy requirements will be met in the new systems. While there will be direct access of public data, Harden noted that private information will only be available to the White House in summarized form, and will not include individual records. Other questioners asked how the quality of information retrieved by the systems would be safeguarded. Zimmerman said that EOP would be selective in using data, and would constantly monitor its quality, as is done in maintaining the quality of a good library, he said.

In response to a question about the use of the systems at the very highest levels of the White House, Harden replied that the President might eventually use a CRT screen in his office.

Opening the presentation on Congressional information systems, Rep. Charlie Rose noted that two-way cable has already been installed in all Congressmen's offices and will permit video as well as data communication. (The House has recently authorized members to purchase, out of their office budgets, color televisions which could be used as display terminals.) In addition, Mr. Rose cited the improved communications with constituents through the use of word processing equipment. He also discussed the importance of computerized mailing lists for Congressmen in countering inaccurate mailings by lobby groups.

Neal Gregory stated that some 230 Congressmen now use terminals to access LEGIS, SOPAD, SCORPIO and JURIS. LEGIS provides information on bill status in both the House and Senate; SOPAD gives an on-going account of proceedings in both Houses; and JURIS contains numerous Justice Department legal briefs. (Some 300 members will have terminals by the end of the year.) Mr. Gregory cited the need for even more advanced word processing equipment to handle at least some of the eight million letters received each month in the House.

Boyd Alexander noted that a detailed, three-month study of members is being initiated to determine the need for additional information systems in the House. He announced that an Amdahl 470V5 had just been purchased to expand the scope of information services. [*Ed.*: An Amdahl spokesman said delivery was expected May 15th.] According to Mr. Alexander, a list of members' recorded votes will be added to LEGIS around July.

John Swearingen announced a new Senate study released in May, entitled *Information Systems for the United States* (#). Mr. Swearingen also noted the need for separate House and Senate groups to oversee information systems, stating that the situation in the two Houses is comparable to different companies with varying rules and procedures. He added that computer usage in the Senate is less than that in the House. According to Swearingen, the Senate receives up to two million letters per month, or 600 letters per week that could be handled (at least in part) by word processing equipment.

In the ensuing question and answer session on Congressional systems, Mr. Rose announced that the House Administration Committee is close to adopting a rule forbidding the use of members' computerized mailing lists by campaign committees. (An ethics rule of the Senate incorporates a similar provision.)

AFIPS President Dr. Theodore J. Williams introduced the participants and moderated the discussions. Washington Activities Committee Chairman Keith W. Uncapher complimented AFIPS volunteers for objectivity in providing technical information to the Government. Mr. Uncapher noted that differing views can be extremely valuable to high-level policymakers who must consider all options. The Washington Activities Committee chairman also introduced Alexander D. Roth, recently named to head the AFIPS Washington Office.

WASHINGTON DEVELOPMENTS

APPEALS COURT ORDERS AT&T, FCC TO IMPLEMENT PREVIOUS EXECUNET RULING, REFUSES TO RECONSIDER DECISION; AT&T SEEKS STAY WHILE ASKING SUPREME COURT TO PONDER CASE

The Bell system operating companies have begun processing requests by MCI Communications Corp., a Washington-based specialized carrier, for local telephone connections allowing MCI to expand its long-distance phone service, Execunet, to 12 additional cities. In April, the U.S. Court of Appeals in Washington ordered AT&T and the Federal Communications Commission (FCC) to implement the court's July, 1977, ruling which authorized the Execunet service. In May, the court refused to reconsider its earlier decision as requested by AT&T and the FCC. At press time, AT&T is seeking a stay while it asks the Supreme Court to consider the case.

Despite the 1977 appeals court ruling (which the Supreme Court would not overrule last January), the Federal Communications Commission, in February (see *Washington Report*, 4/78, p. 3), held that AT&T was not required to make the additional local connections required to implement Execunet. At that time, only Commissioner Joseph Fogarty dissented from the FCC, ruling, stating that the commission's action nullified the 1977 appeals court ruling. In its April ruling the court agreed, arguing that "MCI is in effect no better off than it was during the entire course of the litigation in this court. Notwithstanding our favorable decision, it is unable to expand Execunet."

The appeals court contended that AT&T and the FCC "twisted the issues we contemplated in this case beyond recognition," AT&T had argued that it would have to raise long-distance telephone rates if competition was introduced by MCI into densely populated areas with Execunet. The FCC held that the local connections should be denied, contingent on its study into the effects of competition on AT&T.

The Execunet service, which provides voice and data communications, involves calling a local number, then giving a code number to be connected through MCI's network with another telephone in one of 18 cities now served by Execunet. The appeals court decision is also expected to affect Southern Pacific Communications Co.'s plans to market a service similar to Execunet, called Sprint.

FEDERAL RESERVE BOARD ISSUES FINAL APPROVAL FOR NATIONWIDE ACH INTERCONNECTION

The Federal Reserve Board has issued final approval for a nationwide interconnection of automated (check) clearing houses (ACHs) which, by the end of this year, could permit the Fed's corporate customers to debit or credit their private customers' accounts using the Federal Reserve Communications System (FRCS) (see *Washington Report*, 3/78, p. 8). In the past, the Fed has provided ACH check processing, check settlement, and check delivery services on a strictly regional basis.

The April decision follows a 1976 pilot program undertaken by the Fed which was criticized by a former White House Office of Telecommunications Policy (OTP) official as a "surreptitious development of an on-line capability." In the interregional pilot program, some of the Fed's corporate customers filed debit or credit instructions on magnetic tape with their local ACHs. These instructions were then transmitted with FCRS to other regional ACHs and eventually to the corporate customers' banks.

In January, seeking comment on the proposed nationwide program, the Fed's Board of Governors said that "the probable long-run efficiencies resulting from interconnection of all operating ACH facilities justify the Board's action at this time to provide these services . . . Moreover, the Board regards its action to interconnect the current regional ACH facilities as a research and development program that will provide technical data and experience in the operation of the nationwide ACH facilities. The Federal Reserve System intends to make this information available to those in the private sector interested in the development of alternative systems."

The Fed also cited recommendations of the National Commission on Electronic Fund Transfers (NCEFT) (see *Washington Report*, 11/77, p. 2) which urged the Fed to continue development of "ACH-like services," while also encouraging private sector development in the same area. However, the Privacy Protection Study Commission (see *Washington Report*, 8/77, p. 2) recommended that "no Government entity be allowed to own, operate, or otherwise manage any part of an electronic payments mechanism that involves transactions among private parties." The Fed has recently implemented procedures which mandate removal of most all individual names held in a data base after 30 days.

According to the Fed, 95 out of 121 sets of comments received since last January endorsed the interregional ACH connection. Among those critical of the program, the Department of Justice noted that Federal Reserve involvement would discourage the private sector from developing similar systems because the Fed does not charge for its program.

In its January announcement, the Fed added that provision of the "interbank service" should also "enhance the opportunities open to depository institutions for developing improved 'retail' payments services for the public." Although not provided for in this nationwide ACH interconnection, a point-of-sale (POS) switch could conceivably link consumers and retailers with ACHs and the Fed. The NCEFT urged the Federal government not to become involved "operationally" in POS switches "at present or in the foreseeable future."

POLICYMAKING REPORTEDLY BEING CENTRALIZED IN WHITE HOUSE AS GELLER HEARING HELD ON NEW ASSISTANT SECRETARY OF COMMERCE NOMINATION

Generally recognized as the Carter Administration's chief potential spokesman on telecommunication policy, Henry Geller appeared before the Senate Committee on Commerce, Science and Transportation to answer questions about his nomination by the President as Assistant Secretary of Commerce for Communications and Information. Although receiving a friendly welcome from the Senate committee, Geller's appearance on April 14th was overshadowed by a controversy over the failure of Barry Jagoda, special assistant to the President for Media and Public Affairs, to appear before the committee as requested on the same day.

Presidential Adviser Said to Exceed Role in Telecommunications. Sen. Ernest F. Hollings (D-S.C.) invited Jagoda to appear before the committee to respond to allegations that Mr. Carter's special assistant might be exceeding his authority as adviser to the President, thus detracting from Geller's presumed status as chief spokesman for telecommunications policy. In declining to appear, Jagoda wrote Hollings that (as special assistant) his role is "advisory, and I have no decisionmaking authority in telecommunications policy." It appears, at press time, that until Jagoda's status is resolved to the committee's satisfaction, the Geller nomination will be delayed.

HOLLINGS, COMMERCE COMMITTEE GIVE FRIENDLY RECEPTION TO GELLER (AFIPS/P. McCarter)

Policymaking Said Being Centralized in White House. Although the President's reorganization of computer-related bodies stressed the need for combining the functions of the White House Office of Telecommunications Policy with the Commerce Department's Office of Telecommunications in order to strengthen Cabinet government, recent developments (including the Jagoda controversy) indicate that the President may be centralizing policymaking in the White House.

Mr. Carter's aides and Cabinet met in April at Camp David reportedly to determine procedures for centralizing long-range decisionmaking in the White House. The apparent shift in emphasis from Cabinet government to an increase in White House responsibility is further dramatized by the recent appointment of Anne Wexler as special assistant to the President. Ms. Wexler was formerly deputy undersecretary for Regional Affairs in the Department of Commerce.

Geller Describes Working Relationship With Commerce Secretary, President. At his Senate confirmation hearing, Assistant Secretary of Commerce-designate Geller described his relationship with the Secretary of Commerce, Juanita M. Kreps, and to the President. According to the nominee, he would bring "important decisions" such as those concerned with common carriers and Execunet to Secretary Kreps, with whom Geller says he has "ready access." Prior to meeting with Carter, Geller indicated he would talk first with Mrs. Kreps. Asked how he would react on a disagreement with the Secretary, Geller replied, simply: "She wins."

NTIA to Formulate Position on Bell Bill.' According to the nominee, the new National Telecommunications and Information Administration (NTIA), which Geller will head at Commerce, is formulating a position on the *Consumer and Communications Reform Act*, the "Bell Bill," which F called the "most important issue in telecommunications." Geller said failure to study this issue would make NTIA "an advocate." He added that NTIA is beginning its own studies on subsidies in the Bell System (*i.e.*, whether revenues flow primarily from the private line services to the public line services, as Bell claims, or vice versa). Geller also said that NTIA will participate in the Federal Communications Commission (FCC) rulemaking on message toll service (MTS) and wide-area toll service (WATS).

Electronic Mail. Privacy Issues, Transborder Data Flow Take Precedence Over EFTS. In the nominee's March interview with the AFIPS Washington Office (see *Washington Report*, 4/78, Supplement), Geller noted that NTIA is studying electronic mail, *e.g.*, "Should the U.S. Postal Service go into electronic mail? . . . Are you going to give them [a] monopoly, not likely. Will there be an advantage if they start competing with Bell or Satellite Business Systems?" According to the Assistant Secretary of Commerce-designate, electronic mail, privacy issues and transborder data flow are "proceeding in a faster track" than electronic funds transfer (EFTS). He told AFIPS Research Associate Pender M. McCarter: "We have those ahead of EFTS. We are doing electronic mail right now, looking at what should be done. We are deeply in the midst of privacy and will continue. And, we have made a commitment of resources to the international transborder data flow issue." Geller described NTIA as a "focal point" on transborder data flow, saying: "We ought to be doing the digging and supplying the information to the State Department, to the Congress, and others, as may be necessary."

Geller's Nomination Endorsed. Also appearing before the Senate committee, in support of Geller's nomination, were: Rep. Herbert E. Harris II (D-Va.); Ms. Valeri Byrd, staff director, National Black Media Coalition; and Mr. Paul G. Zurkowski, president, Information Industry Association. Following Mr. Zurkowski's presentation, Sen. Hollings solicited "help from your organization and others, on the convergence of computer and communications."