WARREN J. PLATH

TRANSFORMATIONAL GRAMMAR AND TRANSFORMATIONAL PARSING IN THE REQUEST SYSTEM

1. INTRODUCTION

The REQUEST (Restricted English Question-answering) System is an experimental natural language query system currently under development at the IBM Thomas J. Watson Research Center. The general objective of this work is to explore the feasibility of using restricted subsets of natural language (in this case, English) as the basis for effective man-computer communication, with particular emphasis on the problem of making data-base-oriented services readily available to non-programmer users. Our initial implementation of REQUEST (described here and in a companion paper by S. R. PETRICK, in the I volume) addresses the data-base communication problem with reference to the specific world of business statistics, as exemplified by the summary data published annually in the "Fortune 500".

An essential feature of the language processing approach embodied in the REQUEST System is the employment of a very general transformational parsing algorithm and a large, explicitly formulated transformational grammar in order to produce semantically interpretable underlying structures of input sentences (questions and commands). REQUEST thus differs markedly from such well-known systems as those of F. B. THOMPSON (1973) (direct semantic interpretation of surface structures), W. A. WOODS (1972) (representation of grammatical information and parsing strategy in augmented transition networks), and T. WINOGRAD (1972) (direct incorporation of grammatical information within the parsing program). Furthermore, the underlying structures assigned by our current grammar do considerably more than merely capturing a few relatively superficial syntactic generalizations such as the relationship between active sentences and corresponding passives: they are significantly more abstract than the deep structures of N. CHOMSKY's *Aspects* (1965) and go a long way towards explicit representation

of the meanings of sentences in a notation that bears certain strong resemblances to the predicate calculus. Among the motivations for our choice of approach are 1) the advantages of a transformational model as a vehicle for capturing significant linguistic generalizations, 2) the relative case of interpretation of our abstract underlying structures, and 3) the perspicuity of a system organization which separates data from algorithms and represents linguistic rules directly as units, rather than as discontinuous elements that are distributed over networks or programs.

2. OVERALL SYSTEM ORGANIZATION

The current version of the REQUEST System consists of a set of programs written in LISP 1.5, together with an associated set of data files containing the lexicon, grammar, semantic interpretation rules, and data base. The system runs interactively on a System/360 Model 67 under CP/CMS in 768K of virtual core. As shown in Fig. 1, the transformational component of the system, whose function it is to analyze input word strings and compute their meanings (i.e., underlying structures) consists of two main parts: a preprocessor and a parser. The interpretive component of the system also has two parts: (i) a semantic interpreter, which translates those meanings into executable code; and (ii) a retrieval component consisting of data accessing and formatting functions invoked by the semantic interpreter in order to complete the question-answering process. The present paper deals predominantly with linguistic and computational aspects of the preprocessing and parsing phases, while Petrick's paper covers the details of semantic interpretation and retrieval.

The role of the preprocessor is to segment the input string into words and punctuation marks and then look up each segment in the lexicon, producing a *preprocessed string* of lexical trees which serves as input to the parser. Multi-word strings that function as lexical units are identified by lookup in a special phrase lexicon; while arabic numerals representing cardinals, ordinals, and dates are supplied with lexical trees algorithmically rather than by matching against the lexicon. In cases where the information in the preprocessed string is inadequate, due to the presence of misspellings, unknown words, ambiguous pronoun references, and the like, the preprocessor prompts the user to supply the required information.

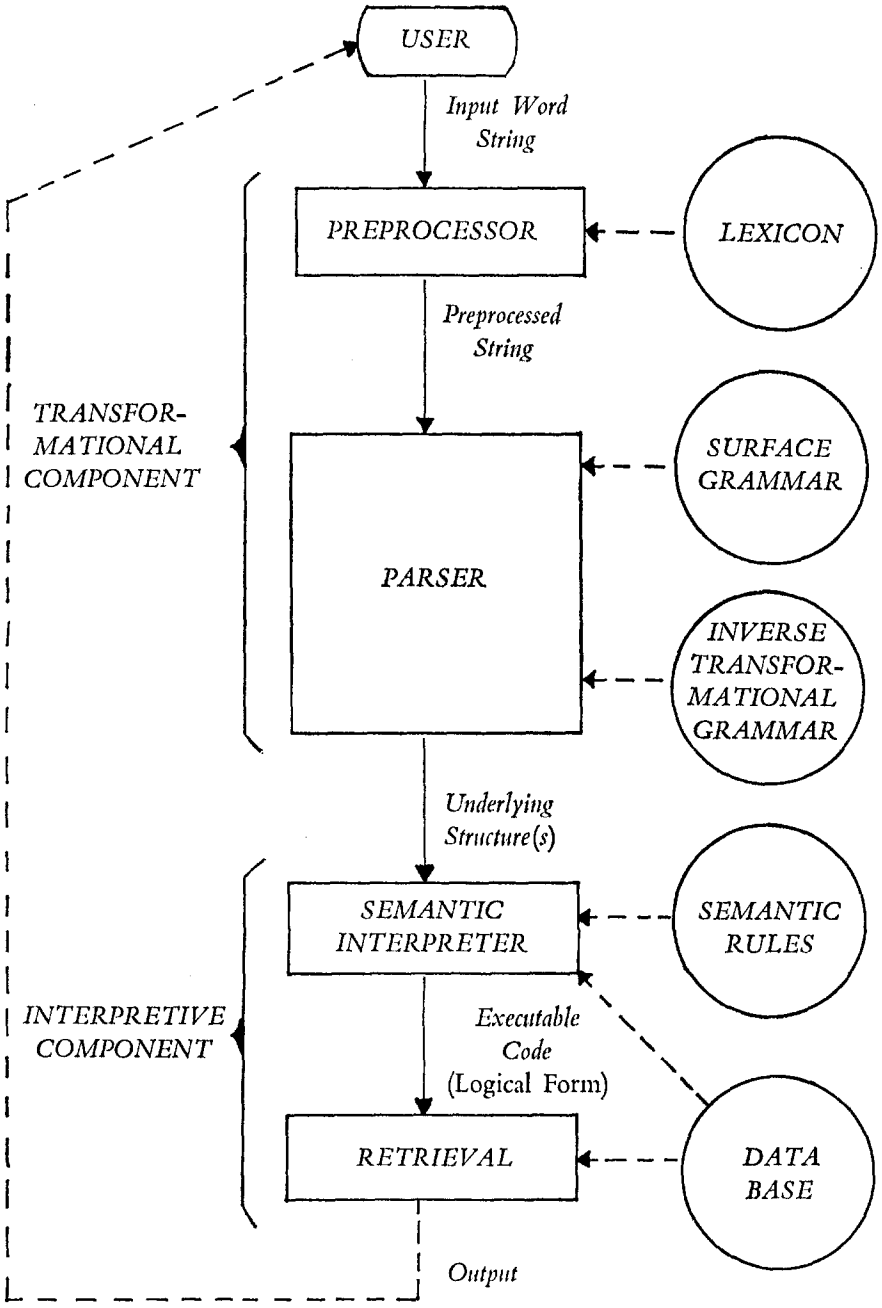


Fig. 1. *Overall System Organization.*

Operation of the parser[1] proceeds in two stages: first, the rules of a context-free surface structure grammar are applied to the preprocessed string by a phrase structure parser in order to compute the surface structure of the sentence. (Because of the well-known inadequacy of pure phrase-structure systems in providing unambiguous surface analyses of sentences (S. KUNO, A. G. OETTINGER, 1963), it is often the case that *several* structures are assigned). Next, the transformational parser processes each surface structure in turn, attempting to map it step-by-step into a corresponding underlying structure. In this process, the parser employs a set of transformational inverses which it applies in precisely the opposite order from that in which the "forward" counterparts of the same transformations would be employed in sentence generation: inverses of the postcyclic transformations are applied first, starting with the "latest" and ending with the "earliest"; then the inverses of the cyclic transformations are applied (also in last-to-first order) working down the tree from the least deeply embedded (main) clause to those that are most deeply embedded. The parser can check the validity of the inverse transformational derivation at each point by testing the corresponding forward transformation to make sure *a*) that (if obligatory) it does *not* apply to the current tree if its inverse failed to apply and *b*) that it *does* apply if the inverse applied (and in fact precisely undoes the effect of applying the inverse). This mode of operation is particularly useful for debugging purposes.

In the course of transformational parsing, most of the spurious surface structures are rejected very quickly by special blocking rules which employ transformational pattern matching to filter out ill-formed configurations not detectable by a context-free phrase structure mechanism. In the experiments we have carried out to date, it has almost uniformly been our experience that precisely one underlying structure is assigned to each sentence except in cases where *a*) there is genuine semantic ambiguity or *b*) a sentence outside the current coverage of the transformational grammar has been entered into the system. At least some of this initial success in disambiguation is due to a policy of making certain transformations sensitive to semantic as well as syntactic information.

---

[1] The original design and implementation of the parser is due to S. R. PETRICK (1973) More recently, it has been significantly revised and extended by M. Pivovonsky, who (with the help of E. O. Lippmann) has also been chiefly responsible for implementation of the preprocessor.

3. AN EXAMPLE

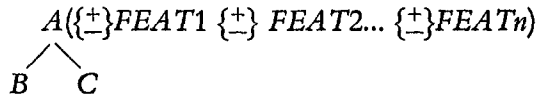
In order to illustrate how the REQUEST System uses transformational parsing to compute the underlying structures of input sentences, let us follow the processing of a typical query step by step. At the start of the session (Fig. 2) the system prompts the user by typing out the message " QUESTION? ", and the user responds by typing in " What was GM's gross income for 1970? ". At this point, the preprocessor segments the input string into a sequence of words and punctuation marks which is checked against the phrase lexicon to detect the presence of any multiword lexical strings that should be treated as units. The result of this process – consisting of a serially-numbered list comprising words, multi-word units, and punctuation marks – is typed out at the terminal to confirm receipt of the query. As can be seen from Fig. 2, the input sequence " gross income " has been identified as a single unit " GROSS-INCOME " on the basis of lookup in the phrase lexicon. (Had the user typed in " General Motors " or " G. M. " instead of " GM ", these strings would also have been handled in the same fashion.)

```
QUESTION?
What was GM's gross income for 1970?
   WHAT(1) WAS(2) GM(3)'S (4) GROSS_INCOME(5) FOR(6) 1970(7) ?(8)

   PREPROCESSING OUTPUT:

((WHAT (OR (VADJ (+ ADJ + QUANT) WH SOME)
           (NP (NOM (V (+ ADJ + QUANT) WH SOME)
                    (NOM (NOUN (- HUMAN + SG) (V THING)
                                              (INDEX (- CONST) X1 )))))))

(WAS (AUX (+ PAST + SG) BE))
(GM    (NPROPNOM (NOUN (- HUMAN + SG) (INDEX (+ CONST + CO) GM))))
('S    (GENAF 'S)
(GROSS-INCOME
       (NMNL (- HUMAN + SG + Q NOUN + ARG1 + PERIODIC + NMNL)
          (V (+ POSS2) AMOUNT MONEY GROSS)
          (INDEX (- CONST) X5)))
(FOR   (PREP FOR)
(1970) (PROPNOM (NOUN (- HUMAN + SG + TIME) (INDEX (+ CONST + YEAR) 1970))))
(?     (PUNCT (+ QUES) ? )) )
```

Fig. 2. *Input and Processing Output for a Typical Query.*

The next action of the preprocessor is to look up each of the numbered items in the lexicon. Absence of a given item from the lexicon may stem either from an actual gap in lexical coverage or from an input typing error on the part of the user. In such cases the user is prompted to retype the offending item, but can also drop the entire question and start again if he so chooses. Since there are no errors or missing words in our current example, however, the preprocessor is able to complete the entire lookup successfully without calling upon the user for corrective action.

The final output of the preprocessing phase (Fig. 2) is an ordered list of dictionary entries, each consisting of an entry key paired with an associated tree (or disjunction of trees) represented as a parenthesized string. The basic notational convention used in our tree representation is that an expression of the form $(A(\{^{+}_{-}\}FEAT1\ \{^{+}_{-}\}FEAT2...\{^{+}_{-}\}FEATn)\ B\ C)$ stands for a tree of the form:



where A, B and C are nodes and $FEAT1$ ... $FEATn$ are syntactic or semantic features.

As an examination of the preprocessing output of Fig. 2 shows, the lexicon is designed in such a way that the process of substituting lexical trees for corresponding input items results in making numerous local changes in the input string – changes which take it as far as possible, at this early stage, along the way towards underlying structure (J. J. ROBINSON, 1973). Thus, on the one hand, all inflected items are replaced by stems whose part-of-speech nodes carry the corresponding tense and number information in the form of syntactic features, along with a variety of semantic features. Furthermore, common nouns such as "gross income" are already interpreted as combinations of underlying predicates (V) and variables (INDEX (-CONST)) in anticipation of the way they are represented in underlying structure. (The specific values of index variables, such as the "X1" and "X5" in Fig. 2, are used to keep track of matters of reference in more complicated sentences by having the preprocessor assign identical variables to pronouns and their antecedents. No such complexity arises here, however, and the preprocessor simply employs the word number of each common noun in manufacturing a unique

variable name.) In addition, proper nouns, such as "1970" and "GM", are treated as logical constants (INDEX (+ CONST)) which belong to particular semantic classes – in this case those of year names (+ YEAR) and company names (+ CO).

Once specification of the preprocessed string is complete, control passes to the parser, whose first task is to attempt to assign a surface structure to the preprocessed string. The parser makes use of a context-free surface grammar and standard phrase structure parsing techniques in attempting to connect the sequence of lexical trees into a coherent surface structure tree (actually as many such trees as the rules will allow). In this case three distinct surface structures are produced, as indicated by the bracketed terminal strings displayed at the top of Fig. 3. The only difference in the three surface analyses lies in the treatment of the string "for 1970": In analysis (1) it is (correctly) treated as a postmodifier of "gross earnings" – now represented in standardized form as ((AMOUNT MONEY GROSS) X5); in (2) the string is broken up, with "for" treated as a stranded preposition postmodifying "gross earnings" and "1970" treated as a major noun phrase constituent of the sentence (e.g., as in "What year are those the figures for – 1970?"); while the third analysis again treats "for 1970" as a prepositional phrase unit, but this time as a major constituent of the sentence rather than as a postmodifier of "gross earnings".

Control now passes to the transformational parser, which takes each surface structure in turn (in last-to-first order) and attempts to map it step by step into a corresponding underlying structure through the systematic application of inverse transformations. As shown in Fig. 3, analyses (3) and (2) are rapidly eliminated through the application of the blocking transformations TCMPDBLK ("time compound blocking") and PRPBLOCK ("(stranded) preposition blocking"), respectively. TCMPDBLK exemplifies the interaction of syntactic and semantic information in the transformational component of the REQUEST System, in that it filters out a variety of otherwise acceptable structures in which a noun phrase or prepositional phrase with head noun marked (+ TIME) is adjacent to a noun phrase with head noun marked (+ PERIODIC), but where the former is not analyzed as a modifier of the latter. PRPBLOCK simply eliminates analyses where a putative stranded preposition immediately precedes a nominal expression.

The parser now proceeds to work on analysis (1), starting out by applying inverse postcyclic transformations that eliminate terminal punctuation (continuation 1.1) and genitive affixes (continuation 1.2),

QUESTION?
What was GM's gross income for 1970?
WHAT(1) WAS(2) GM(3) 'S(4) GROSS_INCOME(5) FOR(6) 1970(7) ?(8)

SURFACE STRUCTURES:
1. (((WH SOME) (THING X1)) BE ((GM 'S) (((AMOUNT MONEY GROSS) X5) (FOR 1970))) ?)
2. (((WH SOME) (THING X1)) BE ((GM 'S) (((AMOUNT MONEY GROSS) X5) FOR)) 1970 ?)
3. (((WH SOME) (THING X1)) BE ((GM 'S) ((AMOUNT MONEY GROSS) X5)) (FOR 1970) ?)

SENTENCE 3:
(((WH SOME) (THING X1)) ((GM' S) ((AMOUNT MONEY GROSS) X5)) (FOR 1970) ?)
FORWARD TCMPDBLK APPLICABLE. NO CONTINUATION

SENTENCE 2:
(((WH SOME) (THING X1)) BE ((GM 'S) (((AMOUNT MONEY GROSS) X5) FOR)) 1970 ?)
FORWARD PRPBLOCK APPLICABLE. NO CONTINUATION

SENTENCE 1:
(((WH SOME) (THING X1)) BE ((GM 'S) (((AMOUNT MONEY GROSS) X5) (FOR 1970))) ?)
SPNCTINS APPLIED. GO TO CONTINUATION 1.1

CONTINUATION 1.1:
(((WH SOME) (THING X1)) BE ((GM 'S) (((AMOUNT MONEY GROSS) X5) (FOR 1970))))
GENINFL APPLIED. GO TO CONTINUATION 1.2

CONTINUATION 1.2:
(((WH SOME) (THING X1)) BE (GM (((AMOUNT MONEY GROSS) X5) (FOR 1970))))
WHERASE APPLIED. GO TO CONTINUATION 1.3

CONTINUATION 1.3:
(((WH SOME) (THING X1)) BE (GM (((AMOUNT MONEY GROSS) X5) (FOR 1970))))
INFOR APPLIED. GO TO CONTINUATION 1.4

CONTINUATION 1.4:
(((WH SOME) (THING X1)) BE (GM (((AMOUNT MONEY GROSS) X5) (IN 1970))))
WHATFRNT APPLIED. GO TO CONTINUATION 1.5

CONTINUATION 1.5
(BE (GM (((AMOUNT MONEY GROSS) X5) (IN 1970))) ((WH SOME) (THING X1)))
PREPINS APPLIED. GO TO CONTINUATION 1.6

CONTINUATION 1.6:
(BE (GM (((AMOUNT MONEY GROSS) X5) 1970)) ((WH SOME) (THING X1)))
WHATFORM APPLIED. GO TO CONTINUATION 1.7

CONTINUATION 1.7:
(BE (GM (((AMOUNT MONEY GROSS) X5) 1970)) ((WH SOME) (AMOUNT X1)))
WHATNUMA APPLIED. GO TO CONTINUATION 1.8

CONTINUATION 1.8:
(BE (GM (((AMOUNT MONEY GROSS) X5) 1970)) (A (((WH SOME) LARGE) (AMOUNT X1))))
QUWHMARK APPLIED. GO TO CONTINUATION 1.9

CONTINUATION 1.9:
(BE (GM (((AMOUNT MONEY GROSS) X5) 1970)) (A (((WH SOME) LARGE) (AMOUNT X1))))
ERASEBDS APPLIED. GO TO CONTINUATION 1.10

CONTINUATION 1.10:
(BD BE (GM (((AMOUNT MONEY GROSS) X5) 1970)) (A (((WH SOME) LARGE) (AMOUNT X1))) BD)
IDEQUDEL APPLIED. GO TO CONTINUATION 1.11

CONTINUATION 1.11:
(BD BE EQUAL (GM (((AMOUNT MONEY GROSS) X5) 1970)) (A (((WH SOME) LARGE) (AMOUNT X1))) BD)
AUXINUMA APPLIED. GO TO CONTINUATION 1.12

```
 CONTINUATION 1.12:
(BD EQUAL (GM (((AMOUNT MONEY GROSS) X5) 1970)) (A (((WH SOME) LARGE) (AMOUNT
X1))) BD)
 PRUNES APPLIED. GO TO CONTINUATION 1.13

 CONTINUATION 1.13:
(BD EQUAL (GM (((AMOUNT MONEY GROSS) X5) (* 1970 *))) (A (((WH SOME )LARGE) (AMOUNT
X1))) BD)
 NPPREPOS APPLIED. GO TO CONTINUATION 1.14

 CONTINUATION 1.14:
(BD EQUAL (THE (((AMOUNT MONEY GROSS) X5) (* GM 1970 *))) (A (((WH SOME) LARGE)
(AMOUNT X1))) BD)
 GENOFMRK APPLIED. GO TO CONTINUATION 1.15

 CONTINUATION 1.15:
(BD EQUAL (THE (((AMOUNT MONEY GROSS) X5) (* GM 1970 *))) (A (((WH SOME) LARGE)
(AMOUNT X1))) BD)
 ORDNOUNF APPLIED. GO TO CONTINUATION 1.16

 CONTINUATION 1.16:
(BD EQUAL (THE (X5 (* (AMOUNT MONEY GROSS) X5 GM 1970 *))) (A (((WH SOME) LARGE)
(AMOUNT X1))) BD)
 SEARCHING FOR EMBEDDED CLAUSES

 SENTENCE 4:
((AMOUNT MONEY GROSS) X5 GM 1970))
 ERASEBODS APPLIED. GO TO CONTINUATION 4.1

 CONTINUATION 4.1:
(BD (AMOUNT MONEY GROSS) X5 GM 1970 BD)
 LOC2FEAT APPLIED. GO TO CONTINUATION 4.2

 CONTINUATION 4.2:
(BD (AMOUNT MONEY GROSS) X5 GM 1970 BD)
 PREPFNP2 APPLIED. GO TO CONTINUATION 4.3

 CONTINUATION 4.3:
(BD (AMOUNT MONEY GROSS) X5 GM 1970 BD)
 SEARCHING FOR EMBEDDED CLAUSES

 A STRUCTURAL DESCRIPTION OF SENTENCE 4:
(BD (AMOUNT MONEY GROSS) X5 GM 1970 BD)

 A STRUCTURAL DESCRIPTION OF SENTENCE 1:
(BD EQUAL (THE (X5 (* BD (AMOUNT MONEY GROSS) X5 GM 1970 BD *))) (A (((WH SOME) LARGE)
(AMOUNT X1))) BD)

 UNDERLYING STRUCTURES:
1. (BD EQUAL (THE (X5 (* BD (AMOUNT MONEY GROSS) X5 GM 1970 BD *))) (A (((WH SOME)
LARGE) (AMOUNT X1))) BD)

 LOGICAL FORM:
(SIZEOF
  (SETX
    (QUOTE X1)
    (QUOTE (AND (EQUAL (QUOTE 18752354000) X1 (AMOUNT X1))) ) )

 ANSWERS:
1. $18752354000
NEXT QUESTION?
```

Fig. 3. *On-line Trace of the Processing of a Typical Query.*

substituting in their stead the features (+ QUES) and (+ GEN), respectively, on higher nodes which do not show up in the trace displayed in Fig. 3. Inverse WHERASE applies next, inserting the feature (+ WH) on the NP node that dominates the structure ((WH SOME) (THING X1)) – an action once again not visible in the trace (continuation 1.3). At this point, the inverse transformation INFOR recognizes the preposition " for " as a surface variant of " in " within a (+ TIME) postmodifier of a (+ NMNL) noun (continuation 1.4), following which inverse WHATFRNT (one of five variants of WH-movement implemented in the current grammar) effects a major reordering of clause components by sister-adjoining the NP dominating ((WH SOME) (THING X1)) to the right of the NP following the auxiliary (continuation 1.5).

Processing the structure against the remainder of the postcyclic transformations successively deletes the preposition " in " in favor of the features (+ LOC2) and (+ IN2) (continuation 1.6), replaces " thing " by the more specific noun " amount " (continuation 1.7), expands ((WH SOME) (AMOUNT X1)) into a (A (((WH SOME) LARGE) (AMOUNT X1))) (continuation 1.8), and finally eliminates the (+ WH) feature on the NP dominating the latter (continuation 1.9). (It should be noted here in passing that the WHATNUMA transformation not only accounts in part for the underlying equivalence of sentence pairs like " *What* were GM's earnings? " and " *How large* were GM's earnings? ", but also for the equivalence of pairs beginning

" *How large* $\left\{\begin{smallmatrix}\text{an amount}\\ \text{a number}\end{smallmatrix}\right\}$ of ... " and

" *What* $\left\{\begin{smallmatrix}\text{number}\\ \text{amount}\end{smallmatrix}\right\}$ of ... ".)

The application of inverse cyclical transformations begins with the insertion of sentence boundaries (BD) on the highest clause by the inverse ERASEBDS transformation (continuation 1.10). This is followed by insertion of the main predicate " EQUAL " by inverse IDEQUDEL (continuation 1.11). IDEQUDEL is one of a set of four related transformations that supply one of the five underlying predicates IDENTICAL, EQUAL, RANK, LOCATED, and MEMBER in clauses with a BE auxiliary and no predicate head (verb or adjective), the choice depending on syntactic and semantic features of the main NP's in the clause. Examples of each of the five types of copulative clauses covered by these transformations are:

" Is Jones XYZ's president? ", " What were GM's 1970 earnings? ", " What company was fifth in 1969 sales? ", " Is the headquarters of IBM in Pittsburgh? ", and " Is a city a place? ", respectively.

After the main predicate " EQUAL " has been inserted, the auxiliary is deleted by inverse AUXINUMA, whose forward counterpart combines the functions of auxiliary insertion and number agreement (continuation 1.12). Next (continuation 1.13), inverse S-pruning (PRUNES) turns the postmodifying structure " 1970 " into a subordinate clause fragment dominated by a sentence node S1. (This shows up in the bracketed terminal string as insertion of " (* *) " surrounding " 1970 ".) Inverse noun phrase preposing (NPPREPOS), which relates such surface pairs as " GM's sales " and " the sales of GM ", then applies (continuation 1.14) (i) moving the NP node dominating " GM " into initial position in the subordinate clause fragment, (ii) marking that NP with the features (+ PREP) and (+ OF), and (iii) replacing the original copy of that NP by " THE ". The features (+ PREP) and (+ OF) are then erased by inverse GENOFMRK, which also replaces the feature (+ GEN) by the feature (+ POSS2), which indicates inalienable possession.

The final transformation that applies in the first cycle is inverse noun formation (ORDNOUNF), which moves the V dominating (AMOUNT MONEY GROSS) into the subordinate clause fragment as main predicate along with an NP dominating a copy of the variable X5, with the original copy of X5 remaining outside the subordinate clause as the " binding " instance of that variable. (The original version of the noun formation transformation – and, more generally, the overall treatment of variables and constants in the grammar – are due to Paul Postal.)

Application of the inverse cyclical transformations to the subordinate clause (now labelled " SENTENCE 4 " by the program) is relatively uneventful. The only transformations that apply merely insert sentence boundaries (continuation 4.1) and delete the features (+ IN2) and (+ POSS2) from the NP nodes dominating " 1970 " and " GM ", respectively. Since there are no further embeddings, the transformational parsing procedure terminates, producing a unique underlying structure for the sentence. As described by S. R. PETRICK, the underlying structure is then mapped by a Knuth-style semantic interpreter into a corresponding logical form. Finally, the logical form is evaluated interpretively to yield the answer " $18, 752, 354, 000 ".

4. CURRENT STATUS OF THE REQUEST GRAMMAR

The immediately preceding example provides a partial illustration of approximately one-quarter of the transformational apparatus of the current REQUEST grammar, which includes 18 blocking rules and 63 pairs of transformations (forward and corresponding inverse). The surface grammar currently comprises 261 context-free rules which, through the use of rule-factoring techniques, effectively represent a set of rules more than half as large again. The grammar also contains 18 base rules and 222 auxiliary phrase structure rules, which are used by the parser in checking the well-formedness of underlying structures and transformationally-derived structures, respectively.

The grammatical coverage provided by the current REQUEST grammar is relatively extensive, but tends to be heavily concentrated in a limited number of areas, with certain important construction types not covered yet at all. This pattern of coverage is not a product of accident or oversight, but has arisen quite naturally from two basic considerations. The first involves one of the more obvious strengths of a transformational model of natural language (particularly one with relatively " deep " underlying structures): its capacity for relating wide ranges of synonymous surface structures to common underlying forms. Within the context of a man-machine interaction situation, it is highly desirable to capitalize on this strength as a means of enhancing the naturalness of the interaction language for the user. To this end, we have been attempting to build up our grammatical coverage in such a way as to allow the user great latitude of grammatical formulation in expressing each of the limited number of semantic relationships provided for in the system. With such " locally full " coverage, we hope to minimize what the user must learn about constructions to be avoided and how to avoid them.

The second consideration is that for REQUEST, as for any natural language understanding system, it makes sense to concentrate on grammatical constructions of importance in expressing the central relationships in the " world " of the data base in question. A case in point in our current grammar is the extensive coverage of constructions involving notions of rank and ordinality – concepts which assume a central role in the world of the " Fortune 500 ".

An example of what has been achieved so far in striving towards the goal of locally full coverage is the range of relative clause structures

handled by the current grammar. To begin with, the basic relative clause patterns covered (1) include counterparts of all the declarative main clause patterns handled by the system – among them patterns involving actives (1. a, d, g, h, k, l, m), passives (1. b, c, e, f, i, j), datives (1. d, e, f, g, h, i, j), and optional time and place adverbials (1. k, 1, m), as well as the interaction of all of these with clausal negation. In addition, the relative clause types covered not only share with *wh*-questions the option of preposition stranding (1. c, f, h, j), but also include provisions for optional substitution of " that " for the relative pronouns " which ", " who ", " whom ", " when ", and " where " under appropriate circumstances (1. a, c, d, f, h, j, k, l) as well as for optional deletion of those pronouns under still more restricted circumstances (1. d, f, h, j, k, l).

(1)

a. companies {which / that} {} {did not / didn't} {made / make} computers

b. companies by which computers {were / were not / weren't} made

c. companies {which / that / ø} computers {were / were not / weren't} made by

d. companies {which / that} {} {did not / didn't} {sold / sell} {computers to XYZ / XYZ computers}

e. companies by which { computers {were / were not / weren't} sold to XYZ ; XYZ {was / was not / wasn't} sold computers }

f. companies {which / that / ø} { computers {were / were not / weren't} sold to XYZ ; XYZ {was / was not / wasn't} sold computers } by

g. companies to which ABC {} {did not / didn't} {sold / sell} computers

h. companies {which / that / ø} ABC {sold / {did not / didn't} sell} computers to

i. companies to which computers {were / were not / weren't} sold by ABC

j. companies {which / that / ø} computers {were / were not / weren't} sold to by ABC

k. the computers {which / that / ø} ABC {sold / {did not / didn't} sell} (to) XYZ

in New York in 1969

l. the year {in which / when / that / ø} ABC {sold / {did not / didn't} sell}

{computers to XYZ / XYZ computers} in New York

m. the city {in which / where / *that / *ø} ABC {sold / {did not / didn't} sell}

{computers to XYZ / XYZ computers} in 1969

In addition to the surface patterns exemplified in (1), a variety of other relative clause constructions are covered. In (2), we see examples of possessive relatives, which appear in surface structures either as the preposed genitive form " whose " or as the postmodifying prepositional phrase " of which ". As can be observed from (2.b-f), in producing such structures, the *wh*-movement rules of the current grammar can either: (i) " pied pipe " a larger noun phrase containing a relative pronoun to clause-initial position (2.b, d); (ii) front only a relative pronoun which is the object of a preposition, leaving the remainder of the larger NP behind along with a stranded preposition (2.c, f); or (iii) do a combination of (i) and (ii) (2.e).

(2)

a. companies { whose headquarters / the headquarters of which } are (located) in New York

b. companies { whose assets / the assets of which } Chile expropriated

c. companies { which / that / ø } Chile expropriated the assets of

d. companies { whose subsidiaries' earnings / the earnings of whose subsidiaries / the earnings of the subsidiaries of which } the government impounded

e. companies { whose subsidiaries / the subsidiaries of which } the government impounded the earnings of

f. companies { which / that / ø } the government impounded the earnings of the subsidiaries of

Another important part of relative clause handling in the current grammar is the coverage of a variety of patterns of relative clause reduction (3). In the case of adjectival predicates such as "profitable", the (optional) reduction process involves deletion of the relative pronoun and suppression of auxiliary insertion, producing reduced surface clauses like (3.b) from the same structures that underly full clauses like (3.a). If the results of clause reduction contain only an adjectival constituent (e.g., if no phrase like "in 1970" is present), it is obligatorily preposed to yield a structure like (3.c). In cases such as (3.d), where the adjectival predicate itself is independently subject to deletion, one of the possible outcomes (3.e) of the reduction process is a prepositional phrase, with or without a preceding negative. In cases where the predicate in the full relative clause is nonadjectival (3.f, h, j), there is a similar reduction process, again involving deletion of the relative pronoun and suppression of the auxiliary. Here, however, an additional process is involved: replacement of the predicate by the corresponding *-ing* form (3.g., i, k). The final possibilities of relative clause reduction shown in (3) have to do with more restricted processes in which relative clauses with the predicate "have" (3.j, l) optionally reduce to prepositional phrases with "with" or "of" (3.k, m).

(3)

a. companies { which / that } { were / were not / weren't } profitable in 1970

b. companies (not) profitable in 1970

c. (un)profitable companies

d. headquarters { which / that } { are / are not / aren't } { located / ø } in New York

e. headquarters (not) { located / ø } in New York

f. 1969 earnings { which / that } { exceeded / did not / didn't } { exceed } $1, 000, 000

g. 1969 earnings (not) exceeding $1, 000, 000

h. the company { which / that } { ranked / was } { second (highest) / in second place } in 1969 sales

i. the company (ranking) { second (highest) / in second place } in 1969 sales

j. companies { which / that } have { d. / e. / f. / g. }

k. companies { having / with } { d. / e. / f. / g. }

l. the subsidiaries { which / that / ø } ABC has

m. { the subsidiaries of ABC / ABC's subsidiaries }

In concluding our discussion of the current status of the REQUEST Grammar, let us turn from our sampling of current coverage to a brief enumeration of important constructions not now handled by the system. At this writing, major known gaps include:

(a) absence of any provisions for conjunction or disjunction;

(b) need to augment coverage of quantifiers (currently limited to ordinal and cardinal numeral quantifiers) to include a much broader range of logical, numerical, and temporal quantifiers, e.g., " each ", " every ", " all ", " any ", " some ", " no ", " none", " more than *n*", " less than *n*", " exactly *n* ", " at most *n* ", " at least *n* ", " once ", " twice ", " *n* times ", " always ", " ever ", " never ", " more than *n* times ", " at least *n* times ", " *n* times in succession ", etc.;

(c) absence of provisions for handling arithmetic predicates such as " total ", " average ", " ratio ", " rate ", and " per ";

(d) inadequate coverage of comparatives (currently limited to a few constructions with " exceed ") and superlatives (currently limited to clauses with underlying predicate " rank ").

The four areas of deficiency just cited are all of considerable importance in relation to the provision of facilities for asking and answering semantically complex questions that seek information only implicitly stored in a collection of numerical data of the sort we are dealing with. They also represent areas of significant independent linguistic interest. Our present hope is that a reasonable range of the phenomena under (a) - (c) will prove tractable when dealt with under a presently contemplated extension of the parser which will make transformational power available at a point immediately prior to surface parsing: but the extent to which that hope is realized remains to be seen. With regard to comparatives and superlatives, a great deal of work lies ahead of us, particularly when it comes to handling the changes in clause inclusion relationships that seem to be involved in deriving sentence pairs like " XYZ earned more than ABC " and " XYZ's earnings exceeded ABC's " from a common underlying source.

In spite of the large amount of linguistic work remaining to be done in order to achieve truly satisfactory coverage, I believe we have progressed sufficiently far at this point to have demonstrated both the experimental viability and the future promise of a transformationally-based approach to natural language question answering.

REFERENCES

N. CHOMSKY, *Aspects of the Theory of Syntax*, Cambridge (Mass.), 1965.

B. HENISZ-DOSTERT, F. B. THOMPSON, *The REL Project*, in " Quarterly Progress Report ", California Institute of Technology, Pasadena, April 1, 1973.

S. KUNO, A. G. OETTINGER, *Syntactic Structure and Ambiguity of English*, in *Proceedings of the 1963 Fall Joint Computer Conference*, Baltimore, 1963.

S. R. PETRICK, *Transformational Analysis*, in R. RUSTIN (ed.), *Natural Language Processing*, New York, 1973, pp. 27-41.

S. R. PETRICK, *Semantic Interpretation in the REQUEST System*, in the I volume of this book.

J. J. ROBINSON, *An Inverse Transformational Lexicon*, in R. RUSTIN (ed.), *Natural Language Processing*, New York, 1973, pp. 43-60.

T. WINOGRAD, *Understanding Natural Language*, in « Cognitive Psychology », III (January 1972) 1, pp. 1-191.

W. A. WOODS, R. M. KAPLAN, B. NASH-WEBBER, *The Lunar Sciences Natural Language Information System*, Final Report, BBN Report No. 2378, Cambridge (Mass.), June 15, 1972.