AD-775 232

NATURAL COMMUNICATION WITH COMPUTERS IV

Bolt Beranek and Newman, Incorporated

Prepared for:

Advanced Research Projects Agency

January 1974

BOLT BERANEK AND NEWMAN INC

CONSULTING · DEVELOPMENT · RESEARCH

BBN Report No. 2721 January 1974

NATURAL COMMUNICATION WITH COMPUTERS IV

Quarterly Progress Report No. 13
1 October 1973 to 31 December 1973

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Advanced Research Projects Agency or the U.S. Government.

This research was supported by the Advanced Research Projects Agency under ARPA Order No. 1697; Contract No. DAHC15-71-C-0088.

TABLE OF CONTENTS

I. INTRODUCTION

We report the details of our progress on Natural Communications with Computers IV in the areas of;

1. Speech Understanding
2. Distributed Computation and TENEX Improvements
3. Language Research
4. Speech Compression

in this our thirteenth quarterly technical progress report.

During this past quarter, our speech understanding system was demonstrated to the executive committee of the ARPA speech steering committee. The system completed the analysis of a spoken input sentence during the demonstration. Both preparatory to this demo and as a result of things we learned from the demo, many of the components of our speech understanding system were substantially changed.

The acoustic segmentation and labeling programs have been improved in accuracy. We look to still further improvements in this module including a possible new implementation in another, more suitable programming language. We have improved our phonological analysis such that the number of word matches have increased by about a factor of 2 or 3 and the number of correct matches have also increased. The syntax module was substantially debugged and tested as well as improved in the sense that fewer duplicate

or possibly incorrect proposals are made by it. We have improved the interaction between the semantics and syntax modules. Some experiments were initiated this quarter with pragmatic testing of the theories proposed by the various modules. The lexical retrieval programs have been improved in accuracy and now include a package for gathering performance statistics.

This quarter our distributed computation project implemented a JSYS trap mechanism for TENEX which is used to trap monitor calls enabling a program to define its own virtual machine. We have begun work on several improvements to the TIP-RSEXEC system. We have also made several improvements in our RSEXEC system including; correcting some problems associated with the user interface to the distributed file system, expanding the flexibility in controlling jobs at multiple TENEX sites, logging various statistics about distributed file access, and refining the RSEXEC protocol in conjunction with other sites. We have also begun work on the design of an interprocess communication and synchronization facility appropriate to a distributed computer environment.

The reliability perceived by TIP/TENEX users has been improved by correcting a number of deficiencies in TENEX's NCP, and by providing the users with more helpful error diagnosis. The efficiency of TENEX operation has also been

substantially improved by installation of a new scheduler which reduces system overhead from 35% to 15%.

During this quarter computer languages research efforts were again focused on LISP improvements and automatic programming. LISP sysout files now take up much less file space and are easier to transmit across the network because of major format changes. An initial implementation of user-defined data types has been completed. We made visits to SRI and Xerox PARC to exchange ideas about LISP. Our automatic programming efforts have been directed at the discovery of new methods for program synthesis based on the analysis of human program synthesis behavior.

In our speech compression research we have worked mainly on the quantization properties of transmission parameters for the linear predictive vocoder. Several alternate sets of parameters that represent the linear predictor have been investigated as transmission parameters. Although each of these sets provides equivalent information about the linear predictor, their properties under quantization are different. The results of a comparative study of the various parameter sets have indicated that the reflection coefficients possess many desirable quantization properties. Using a spectral sensitivity measure, we have developed a method of optimally quantizing the reflection coefficients.

In particular it was found that the reflection coefficients exhibit a U-shaped sensitivity behavior under quantization: a high sensitivity near ± 1 and a low sensitivity near 0. This clearly indicates that a nonlinear quantization of the reflection coefficients is desirable. Finally, we have demonstrated that a linear quantization of the logarithms of the ratios of the area coefficients approximates the performance of the optimal quantization.

II. CONTINUOUS SPEECH UNDERSTANDING

A. Executive Committee Site Visit

During the past quarter, the speech understanding project at BBN assembled a demonstration system which attempts to analyze a continuous speech signal from its capture as an electrical signal at a microphone to a complete syntactic and semantic analysis. We did not however, include the generation of code to answer the requests from the data base since this capability has been previously demonstrated in the LUNAR system. This system was demonstrated to the executive committee of the Speech Steering Committee during November, and at that time it completed an analysis of the sentence: "How many samples contain Silicon?" after manual intervention to make two small changes in the segment lattice to simulate the effect of planned but not then incorporated capabilities. These two changes were as follows:

1. to bridge a pair of segments labeled [Z][S] with a single segment labeled [Z] to represent the acoustic phonetic fact that voicing can drop out earlier than the frication in a [Z] preceding an unvoiced consonant. (This rule could be incorporated as a phonological rule in the phonological rule component, but its proper place is in the segmenting and labeling program.)

2. to add to the segment lattice an [L] segment which was mislabeled by the acoustic segmenter and labeler as an [UH], [AO], or [OW], blocking the match of the word "silicon". This word would have matched well enough anyway using the phonetic similarity matrix, if it were not for other complications in the match. Specifically, the [L] exhibits a contextual influence on the preceding [IH] reducing it to something which looks more to the labeler like an [UH] or [AX]. This phenomenon could be dealt with either by a smarter segment labeler or by phonological rules, but no such rules were present in the system, and instead, the similarity matrix score was used. Neither the missing [L] nor the reduced [IH] alone would have rejected the word, but together they were sufficient to make the word match look unacceptable. This example illustrates the typical situation that any one of a number of possible improvements to the system -- all independently motivated -- could be used to overcome particular false decisions. In this case, either an improvement in [L] recognition, or the correct treatment of the pronunciation of the word "silicon" (either by generative phonological rules or by changing the dictionary spelling for the same effect) would have corrected this false decision.

Prior to this demonstration, and subsequent to it, much work was done on the various components of the system and is detailed in the following sections:

B. Segmentation and Labeling

Before the ARPA site visit in November, there were several improvements made in the acoustic segmentation and labeling programs. A general purpose dip-detector applied to the energy of the differenced signal now typically identifies three quarters of the previously missed boundaries.

An attempt was made to achieve more accurate recognition of vowels and liquids. Rough target formants are extracted from the formant tracks. The only speaker normalization is a scaling of these target formants according to the fundamental frequency. The resulting normalized formant targets are then compared to the target formants of 15 vowels and liquids using a formant distance/ratio scheme, to come up with an ordering of likelihood. The top 3 (sometimes 4) choices are taken as the answer. For those segments with correct boundaries, 95% accuracy is achieved. Over 50% of the first choices are correct.

A simple algorithm based on the value of the two-pole frequency (which basically reflects the peak in the spectrum) to distinguish place of articulation for strident fricatives has been implemented. The only classes of phonemes for which algorithms for determining place of articulation have not yet been included are nasals, plosives, and nonstrident fricatives.

In many cases, a second attempt is made at labeling a segment. Often, several phonemes are common to both the first and the second label. In an earlier version of the labeler, this resulted in duplication. This redundancy in the segment descriptions has been eliminated in the current version.

In order to aid in assessing performance, some new displays of the segment lattice were produced, both on the CRT and on the Calcomp plotter. It is felt that well designed displays will be of significant help in the overall problem. Also, to facilitate evaluation, each new feature added to the program has been added as a separate module, so that the effect of any one of them can be easily determined.

Since the demonstration, the entire structure of the segmentation and labeling module has been under careful re-evaluation. An attempt is being made to organize the program and the data so that context is more easily considered both in the segmentation and the labeling

procedures. An investigation into other programming languages than FORTRAN (most notably BCPL) has been a necessary part of this evaluation.

C. Phonological Analysis

During the past quarter, work in phonological analysis has involved the implementation of phonological rules in both the analytic and generative directions. In the generative direction, a system for marking the entries in the lexicon was devised to express a few of the most useful of the phonological rules which we have identified and formalized. This system marked certain segments with a number representing the likelihood of deletion of that segment in continuous speech. Several alternative dictionaries were made up. Some had lexical entries which were marked for segments that could be deleted as a result of factors confined only to the word itself. Others, in addition to such information, contained markings for segments that could be deleted due to the influence of contiguous words.

Further phonological research was pursued with emphasis upon features of voicing and aspiration and their interplay with lexical and syntactic stress patterns.

Implementation of Analytic Phonological Rules

As briefly mentioned in the previous quarterly progress report, the segmenter/labeler component of the SPEECHLIS system produces as its output a lattice of labeled segments representing its hypotheses as to the phonetic structure of the utterance. An analytic phonological rule component then augments the lattice with possible underlying segments and segment sequences which could have resulted in the observed acoustic sequences. Each added segment may have associated with it a predicate function which is later used by the word matcher to check the applicability of the given rule based on the specific word spelling and the necessary context. The augmented segment lattice then becomes the input to both the lexical retrieval and the word matching components of the system.

There are two types of processes which are representable by phonological rules. Generative rules transform underlying or base forms to observable pronunciations of words or word sequences. Analytic rules, on the other hand, transform observed phonetic sequences to more basic or "more correct" forms.

Each segment added to the lattice by these analytic phonological rules should be considered conditional upon some requirement imposed by the predicate (or filter) function associated with it. When the word matcher finds a

path through the lattice which is an acceptable word match to a particular lexical entry, it examines the segments in that path for predicate function pointers. For each such pointer that it encounters, it calls the predicate function, giving as arguments the word spelling of that lexical entry, the position within that spelling, and a pointer to the segment in the lattice. The predicate function, which can be an arbitrary piece of code, performs a computation on these arguments and returns true if it accepts the use of that segment in that word match or false if it rejects it. (A possible generalization of this is for the predicate function to return a confidence or probability. The evaluation mechanism in the current word matcher does not seem sophisticated enough to warrant this.)

Although a rule which adds a branch to the segment lattice, based on existing structure, is analytic, the condition imposed by the predicate function associated with the branch is a function of the underlying form in the lexicon, giving the applied-rule-plus-predicate a generative flavor as well. These predicate functions can be used in three ways:

1. To check a context condition not checked in the "analytic" application of the rule. Many relevant factors are or may not be available in the segment lattice. These factors include:
    a. Stress

b. Place of articulation

c. Position of segment with respect to word boundary

2. To compensate for "sloppiness" in the context of the "analytic" application of the rule. For example, if the rule were:

and the segment lattice were labeled:

where x is some set of labels which does not fit the description a, then if the segment c were to be added, an unwanted path x-c would exist in the augmented lattice. One way to eliminate this would be to bridge the entire context by a two-segment branch consisting of a followed by c. This partial copying can become quite complex in general and it can result in duplication of much of the lattice. Instead, the segment c is added anyway, but any word matches using the unwanted path are summarily rejected by the predicate function.

3. A rule of general usefulness may fail to apply for a few exceptional words. Such exceptions may be detected in a predicate function.

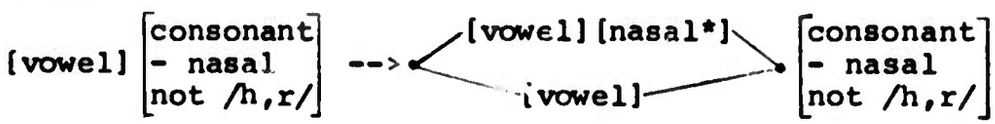
Example: Nasal Deletion Rule

Generative form:

$$\begin{bmatrix}\text{consonant}\\ \text{+ nasal}\\ \alpha\ \text{place}\end{bmatrix} \text{ --> } \emptyset \ / \ [\text{vowel}] \_\_ \begin{bmatrix}\text{consonant}\\ \text{- nasal}\\ \alpha\ \text{place}\\ \text{not /h,r/}\end{bmatrix}$$

"A nasal consonant is deleted if it occurs immediately after a vowel and immediately before a nonnasal consonant with the same place of articulation which is not /h/ or /r/."

Analytic form:



* Predicate function requires:
1. Nasal not word-initial.
2. Preceding segment must be a vowel
3. Nasal may be word-final (if it is, predicate has no way of checking the following segment)
   OR
   Following segment must be a nonnasal consonant (not /h/ or /r/) with same place of articulation as the nasal.

"If there exists a path through the lattice such that a vowel segment is followed by a nonnasal consonant (not /h/ or /r/), then bridge the vowel segment by a two-segment branch consisting of the vowel followed by a nasal. Attach a predicate (described above) to the nasal segment." (If such a branch bridging the vowel already exists, then no new branch need be added.)

The phonological rules component is implemented as a set of BCPL functions which live in the same fork as the word proposer. The segment descriptions are static tables listing sets of phoneme labels. The rules themselves are also elementary data structures describing the necessary context for the rule to apply and each segment of the new branch to be added to the lattice. The properties of these new segments can be expressed absolutely (i.e., duration = 30 msec) or relative to some segment in the context (i.e., duration = 80% of the first segment of the context, or stress = 1 less than that of the third segment). The predicate functions may be arbitrary, but in practice they mainly call a small set of functions which check segment descriptions and vowel stress.

The actual program fragment which specifies the Nasal Deletion Rule is given below. There is a set of phoneme clusters, which are used to describe segments, the rule, and its predicate. The notation for expressing the rule is far from a linguist's notation, but it is quite straightforward. The example is illustrative, not exhaustive.

```
//Definitions of phoneme clusters, used in the rules
//in the predicates.
static
{ VOWEL←table Ø,12,UW,UH,OW,AO,AA,AH,AE,EH,IH,IY,AX,EY
  CONSONANTNOTNASALHR←table Ø,14,P,T,K,B,D,G,F,V,TH,DH,S,Z,SH,ZH
  NASAL←table Ø,3,M,N,NX
  phM←table Ø,1,M
  phN←table Ø,1,N
  phNX←table Ø,1,NX
  LABIALNONNASAL←table Ø,4,P,B,F,V
  DENTPALNONNASAL←table Ø,8,T,D,S,Z,SH,ZH,TH,DH
  VELARNONNASAL←table Ø,2,K,G
}

//The Deleted Nasal Rule itself consists of 3 parts:
//  Description of the necessary context
//  Description of the new branch to be added
//  A string giving the name of the rule
let xDeletedNasal←list
  (list 2,                          //The context has 2 segments,
    OPERAND,,VOWEL,                    //a vowel followed by
    CONTEXT,,CONSONANTNOTNASALHR), //a nonnasal consonant
  (list 2,                          //The new branch has 2 segments:
    (list PHINTERSECTION,,1,           //intersection with the VOWEL
      RDURATION,,8Ø,                   //duration=8Ø% of the VOWEL
      RCONFIDENCE+1,,1ØØ,              //confidence=1ØØ% of the VOWEL
      RSTRESS+1,,Ø,                    //stress=same as the VOWEL
      RBCONFIDENCE,,6Ø,                //right bdry confidence=6Ø
      ENDLIST),
    (list PH,,NASAL,                   //The 2nd segment is a nasal
      CONFIDENCE,,1ØØ,                 //1ØØ means exact match only
      STRING,,"DeNasal",
      PREDICATE,,DeletedNasalPred, //predicate on this segment
      ENDLIST)),
  "DeletedNasal"                    //String giving rule name

//The predicate function for this rule:
and DeletedNasalPred(spellingindex,segptr)←valof
{ let oldrsw←rightsw
  predspx←spellingindex
//check that preceding segment is a vowel, and that
//nasal and following consonant have same place of articulation
  let yesno←check(-1,VOWEL,false)
        ((check(Ø,phM)check(1,LABIALNONNASAL,true))
         (check(Ø,phN)check(1,DENTPALNONNASAL,true))
         (check(Ø,phNX)check(1,VELARNONNASAL,true)))
  if traceflag do tracepred(yesno,segptr,"DeletedNasal")
  rightsw←oldrsw
  let cnt←lv DeletedNasal](yesno=>NACCEPT,NREJECT)
  rv cnt←1+rv cnt
  resultis yesno
}
```

The phonological rule applier takes as input an ordered list of the rules. Each rule is applied from left to right across the lattice before proceeding to the next rule, but rule repetition may be accomplished by including a rule name in the list more than once. Statistics are accumulated on how many times each rule is applied and on how many times its predicate function returns true and false. If a trace flag is enabled, each rule application and each predicate function execution is described on an output file, which may be the user's teletype.

There are now implemented 11 rules, of which 4 are "real" phonological rules (such as the Deleted Nasal Rule described above), and 7 account for other phenomena which are more appropriate to the segmenter/labeler component, but which can be expressed in the same format as the phonological rules (e.g., if the first segment in the lattice is a vowel, it may have been preceded by a P, B, D, G, F, or TH which was missed due to its low energy). We are just beginning to evaluate the effects of various rules, and detailed statistics have not yet been measured. With these 11 rules, we have seen the segment lattice increase in size by factors of 2 to 3. The number of total word matches has increased by about the same factor. The number of correct words matched has also generally increased.

D. Syntax

During the past quarter, the primary emphasis in Syntax was to shake out bugs and test the syntactic component on as many cases as possible in conjunction with the rest of the system. However, several modifications were made to syntax:

(1) Inflectional endings, which used to be noted by Syntax and treated as a special type of event, now appear when appropriate in the theories given to Syntax so that a separate event type is no longer needed.

(2) The number of monitors, event notices, and proposals made by Syntax has been reduced by two methods: first, one-word islands are not syntactically processed (unless they are within 2 segments of another island) because they produce too many partial paths resulting in a large number of monitors, proposals, and notices which are based on very little information. Secondly, careful checking is done to eliminate duplication. For example, if a proposal has been made for determiners starting at a given point with lexical word quality greater than 70, a similar proposal will not be made with a quality of 80.

(3) Considerable attention has been given to the scoring of syntactic proposals, monitors, and event notices so that those likely to provide the most information will be

processed first. The scores are arbitrary and have been (and will continue to be) adjusted as experience indicates. The current scores are as follows:

Proposals for specific words:

70 if word is proposed to fill a gap between islands

90 if proposed at the end of a one word island

80 otherwise

Proposals for syntactic classes:

95 if there is already a member of the class in the word lattice at the relevant place

70 if proposed to fill a gap between islands

80 otherwise

[The above proposal scores reflect the minimum required lexical score for a word to be accepted by the proposal and as such reflect the inverse of the likelihood of the word being there based solely on context. If the context strongly predicts a word, then it can be accepted with a lower lexical score than if the context permits the word but other possibilities are also likely. These proposal scores are always overridden in the case where a proposal of any type is being made to fill the last gap in a theory. When a theory is so close to being completed that virtually any word will be accepted regardless of lexical score, the score for the proposal is set to 0.]

Event notices:

40 for constituent notices

30 for notices which fill a gap between islands

15 for content words at the end of an island of more than one word

10 for non-content words at the end of an island of more than one word

5 for content words at the end of a one word island

0 for non-content words at the end of a one word island

[The scores for event notices are the scores that the control component will use to decide which events to pursue. The higher the score, the more likely the control component is to devote attention to the event.]

Monitors:

15 for word monitor at the end of an island of more than one word

10 for category monitor at the end of an island of more than one word

5 for word monitor at the end of a one word island

0 for category monitor at the end of a one word island

0 for test monitors

[The scores for monitors reflect Syntax's estimate of the relative importance of the thing being monitored to a theory. It is combined with the lexical score of a word that triggers it to determine the score of the resulting event.]

(4) A rudimentary statistics gathering package was included to provide automatic records for each call to Syntax (and

cumulative totals over all calls) of the number of each type of monitor, proposal, and event notice created, the number of constituents found in the WFST, and the number of constituents newly constructed (subclassified by whether they were complete parsings or wnether they were usable or unusable by the theory which caused their construction). This information should help to locate the areas in which further refinement is most needed.

(5) a new grammar was constr ed which is about the same size as the previous small speech grammar (32 states, but which is more error free than the old grammar in terms of the partial paths which may be followed and the structure of the constituents which are built. There is a patch file which may be added to the grammar to allow the inclusion of some relative and reduced relative clauses.

E. Semantics

In a previous Quarterly Progress Report, we discussed a procedure for comparing a representation of the semantic structure of a set of word matches (in the form of nested case frame tokens) and a representation of their syntactic structure (as a parse tree). This procedure was employed in checking the consistency of Semantics' hypotheses with the possible syntactic realizations of the set. At that time, it was necessary for Semantics to have had already postulated the interconnection

between all the content words that appeared in the parse tree before the procedure could work correctly. Thus, Syntax could not include in a syntactic structure a content word that Semantics had not yet seen and have this verification process work correctly. Instead, all content words would have to be added to a theory by Semantics for things to work out.

In order to permit more flexible interaction between Syntax and Semantics, we have begun to relax this restriction and to extend the comparison procedure into a general one for using the semantic network to evaluate the meaningfulness of an arbitrary syntactic tree. As a first step, we started on the implementation of three functions which would communicate across forks and answer the questions: does the referent of this syntactic tree represent a member of a specified semantic class (MEM), does it represent a synonym for a semantic concept (EQU), does the tree represent a possible restriction on some other semantic concept (RMOD).

The first two questions assume that the syntactic construction is the realization of a semantic complex concept and can therefore be associated with at least one case frame. For a given case frame then, they will allow one, when completely implemented, to compute all the possible assignments of syntactic subtrees to semantic cases. The third question assumes one has already established the semantic concepts that the remainder of the syntactic tree instantiates, and currently only works when

questioning the appropriateness of a prepositional phrase as a restrictive modifier. Much more work will be done in this area in the next quarter.

F. Pragmatics

During the past quarter, we tentatively included some pragmatic tests into the procedures for evaluating theories and events. These tests involved checking, for theories and events containing a hypothesized main verb, whether the tense, aspect, voice, and mood of the verb were likely ones in the context of lunar geology. This likelihood would then be reflected in the score of the theory or event. For example, stative verbs like "contain" and "have" (verbs denoting states or attributes) rarely appear in the past tense, while non-stative verbs like "analyze" and "find" (verbs denoting actions or events) rarely appear in the present. It is also unlikely for the progressive aspect to appear with either tense or verb type, as in "Which samples were being analyzed for uranium?". In addition, since we have not been concerned with the particular scientists who have investigated the lunar samples, verbs which allow agent deletion in the passive voice, such as "discover" and "analyze" are more likley to appear that way than in the active voice, where the agent of the action must be specified. (e.g. "Which new minerals were discovered in the lunar breccias?" as opposed to "Which new minerals did the investigators discover in the lunar breccias?".) With respect to mood (declarative, emphatic,

interrogative, imperative), there are certain verbs like "contain" and "analyze" which, either because they are stative or alternatively because they denote actions which the computer could not perform, would never appear as imperatives (e.g. "Contain silicon." or "Don't analyze sample S1ØØØ5.") Emphatic constructions such as "Which samples do contain silicon?", while grammatical, are also unlikely.

These pragmatics tests on tense, aspect, mood and voice could be based solely on the linear ordering of the word matches in the theory or the event, and thus could be done before any syntactic processing. For example, if either the initial auxiliary or the one immediately preceding the main verb were a form of "be", the voice of the utterance would necessarily be passive. If neither were, the voice would be active. This made them attractive as presumably low cost investments with high returns. They required, however, that any auxiliary verb forms which were to be part of the verb phrase of the hypothesized utterance (i.e. forms of "be", "do" and "have") be found and associated with the verb. This resulted in a mushrooming in the number of theories, one for each set consisting of a verb and matching preceding or initial auxiliary. That is, if "found" matched in the word lattice starting at segment 14, and there were word matches for "was", "is" and "has", ending at segment 14, and also ones starting at Ø, there would be seven separate theories, one for "found", one for "was found", etc. instead of merely one for "found". This would also cause the number of

events involving "found" to grow by almost the same number.

In running through examples, we found we were losing more by this proliferation of theories than we were gaining by using pragmatic likelihood information as soon as possible. There seem to be three options open to us now: we could allow Syntax to bring the auxiliaries into a theory, and only use the pragmatic strategies to guide its choice. Implementation of this decision would have to wait until the parser is changed to allow register checking activities on its first pass through a theory. Alternatively, we could wait until a theory is about to be passed to Syntax before spawning sons containing possible auxiliaries. Or thirdly, we could use the crisp cluster mechanism described in the section on Control to group forms of "be", "have" and "do" together, to decrease the magnitude of the proliferation. Though we favor the first option in the long run, we shall experiment with the other two to see what they can buy us.

G. Lexical Retrieval

The lexical retrieval programs have been modified in the past quarter to:

1. be more precise about the use of duration information when an adjacent segment in the word match is postulated as missing or extra.
2. not be arbitrary about selecting only one match of a particular word from a set of matches that have the same

quality. The set of equivalent word matches are now retrieved.

3. facilitate the development of programs which transform the segment lattice with analytic phonological rules. Functions were written and embedded in LISP to select options in the phonological programs, which reside with the lexical retrieval programs in a BCPL fork and share pages with LISP.

4. include a package for gathering performance statistics for the lexical retrieval programs as they operate with and without phonological rules. This package takes as input the name of an utterance, and its transcription as a list of words and where (in time) each begins in the utterance. Information is gathered about the word matches which are found in a lexical retrieval scan of the utterance: how many word matches are found; the distribution of word match scores; how many correct word matches are found; the distribution of word lengths; statistics about clustering of word matches at segment boundaries; statistics about the effect of duration cues on the word match scores.

H. The Control Framework

In the area of Control organization, we are coming to grips with a number of problems related to the size and complexity of the system that we are constructing. These include techniques for coping with the inherent combinatorics of the problem as well

as methods of providing for the space required for programs, grammars, dictionaries, semantic knowledge, and temporary working storage within the framework of forks and shared pages which we have available in TENEX. The following sections detail some of the work that has been done in these areas during the past quarter.

1. Organizing a Complex Software System

One major area to which we have devoted attention is the problem of complexity. As the system and its data structures become more complex, the amount of detail that requires consideration before changes can be made increases. A level of organization which is unnecessary in a simpler system here becomes necessary. One way to relieve the current weight of detail in SPEECHLIS is to separate the representation (as a LISP data structure) of each data object from its logical structure (as a list of its components). To this end, we have developed a set of conventions in LISP for describing our data objects and defining the functions which create, access and modify them. We are currently using these conventions to define the data objects which are shared by Syntax, Semantics, and Control, and to amend the programs which use them.

2. Address Space

One aspect of the problem of system size is the 18-bit limitation on address space in LISP. Currently, Control and part of Semantics live together in one LISP fork, with Syntax and the remainder of Semantics in another. From a functional point of view, the two LISP forks should really be merged; they share large data structures and code modules, interact frequently, and should be organized to facilitate experimentation with strategies for such interaction. Until the overlays for compiled code in LISP become available, we will continue to simulate co-habitation with a combination of shared pages and a new facility (currently being implemented) for passing arbitrary S-expressions for evaluation between the LISP forks. The current LISP forks are individually large enough now to require the use of TINYLISP, a stripped down version of INTERLISP which makes more space available to users.

3. Combinatorics

One way of avoiding an explosion in the number of theories as the size of the word lattice increases is by treating collections of word matches together until some knowledge source finds a distinction among them. Earlier, we introduced the concept of a fuzzy word match as a way of finessing our uncertainty about the precise location of a word match. In the past quarter, we introduced two new notions:

(1) Inflectional endings are no longer treated as separate word matches in a theory, but rather go into making up a new kind of entity, an Inflected Word Match . Inflected word matches resemble simple ones in having a left boundary, right boundary, and quality, but the "word" component of the word match is actually a list of the word matches for the uninflected word and the inflection. E.g.

(I ((16 sample 7 14 (1Ø0 1Ø0) ...)(17 -Z 14 15 (1Ø0 1Ø0) ...)) 7 15 (11Ø 11Ø) ...)

Inflected word matches are grouped with the simple word match from which they derive into a single fuzzy word match, since Semantics does not distinguish between the inflected and uninflected forms of a word. Like other fuzzy word matches, these remain intact until Syntax, or the inclusion of a new word match into the theory, requires that they be broken up. As yet, Syntax does not handle fuzzy word matches, so the theory seen by Syntax in the lower fork has its fuzzy word matches replaced by their longest, best matching member.

(2) We have begun to do some very rudimentary semantic clustering in order to diminish the number of semantically indistinct theories. This starts to make SPEECHLIS a system for understanding speech rather than recognizing it. We have introduced the notion of Crisp Clusters - "cluster", to indicate a set of semantically indistinguishable words, "crisp", to

indicate that the word matches for these words share the same right and left boundaries. Thus, for example, "give", "list" and "print", all matching from segment Ø to segment 4, would be grouped into a single theory as a crisp cluster, which in form resembles a fuzzy word match. The members of a crisp cluster may or may not be syntactically distinguishable. In the above example, they are, since "list" has the option of being considered as a noun, whereas the others do not.

The current strategy for building crisp clusters is insensitive to context, but any really satisfying semantic clustering of words or phrases (if "clustering" is still a useful notion at phrase level) will have to take context into account. This is all part of our continuing development of valid methods of maintaining legitimate vagueness in order to avoid the combinatorial explosion while displaying understanding.

As the control programs are developed and refined in conjunction with our analyses of real segment lattices, both the need for previously untapped sources of knowledge and ideas for how to represent and apply such knowledge appear. For example, knowledge of word boundary co-articulation phenomena is needed when a new word match is considered for inclusion in an existing theory. Such knowledge can be brought to bear if proposals, word lattice monitors, and word lattice queries carry along their context (i.e. the adjacent word matches, if any, being considered). Such context is particularly useful for dealing

with syntactic proposals of small words to fill gaps in a sequence of word matches. A method of representing such contextual information has been designed and is being implemented.

I. Publications and Presentations

During the past quarter, a paper entitled "Control Concepts in a Speech Understanding System" was submitted for presentation at IFIP74, and has been published as BBN Technical Report 2703. Also a paper entitled "Selective Linear Predictive Spectral Matching" was presented at the 86th meeting of the Acoustical Society of America, in Los Angeles, and a representative of the speech project participated in a session on Speech Understanding at the annual meeting of the American Society for Information Science in Los Angeles in October.

During the previous quarter, a number of papers were published and talks were given that were not mentioned in our previous progress report. These included a paper entitled "Mechanical Inference Problems in Continuous Speech Understanding" presented at the IJCAI73 and accepted for publication in the journal of Artificial Intelligence, and a presentation entitled "Syntactic and Semantic Support for a Speech Understanding System" given at the annual meeting of the Association for Computational Linguistics at Ann Arbor in August and in the free session at IJCAI73.

III. DISTRIBUTED COMPUTATION AND TENEX IMPROVEMENTS

A. Introduction

During this quarter we continued our efforts toward making the TENEX system a convenient, efficient, and reliable means for accessing ARPANET resources.

The RSEXEC user interface has been considerably improved and its logging functions have been extended. With our assistance both the AMES-360/67 and Multics groups have implemented prototype RSEXEC user and server programs which cooperate with the TENEX RSEXEC's.

A considerable effort has gone into improving the reliability perceived by TIP/TENEX users: a number of deficiencies were cured in TENEX's NCP, and the error reporting to the user has been greatly improved. This effort is expected to continue at a high level for several months to bring about a number of additional changes which are needed, including changes to the host-IMP and host-host protocols.

We replaced the TENEX scheduler by a more efficient one, reducing typical system overhead from 35% to 15%. Additional work has gone into installing new facilities in the system: JSYS traps, raw network messages, the new TELNET protocol, an autodialler, and KI-10 support.

B. Meetings

We attended several development coordination meetings this quarter: the ILLIAC-IV Project hosted a west coast TENEX meeting on October 27, 1974, which was attended by representatives of eight TENEX sites. Discussion centered on TENEX monitor improvements made at the various sites, (including BBN) coordinating the inclusion of generally useful facilities into the next distributed version of TENEX, and future monitor improvements needed to meet the research goals of the various sites. One important conclusion reached here was that only about 60% of the monitor development work is being done at BBN, and there is no effective site-wide management structure to coordinate this distributed programming effort. We expect to discuss this problem in some detail at the next TENEX meeting.

We also attended a packet radio meeting December 11, 12 at Collins Radio in Dallas. This was our first introduction to the effort so the experience was mainly getting up-to-date on the previous developments. We hope to assist in the design of a packet radio station in the last half of 1974, once the radio propagation equipment and microcomputer controller are well along in development.

C. Special Events

In December, IMP 5 was moved from the Network Control Center at BBN to the BBN-TENEX computer room, and a new TIP was

installed in the NCC. This change was necessary to make enough host ports available to the TENEX research effort: BBN-TENEX, BBN-TENEXB, BBN-11X (PDP-11 peripheral processor), and BBN-11XB. Ports on the NCC-TIP support the PDP-1D and the NCC host. The scarcity of host ports on IMPs has turned out to be quite a problem: planned expansion to additional network service sites BBN-TENEXC and BBN-TENEXD will require still more host ports. Fortunately, the high-speed IMP effort will provide expansion capability to many host ports per IMP.

Another unusual event was the changeover to daylight savings time legislated for January 6, 1974. Since the TENEX operating system keeps its time internally in GMT and converts to local time upon request, the conversion algorithm needed to be modified for this law. This coding was completed and distributed to the network sites well in advance of the actual date of the change and all sites, except one, had installed the modification in time for the change.

D. Distributed Computation

1. Improvements to the RSEXEC System

We have made several improvements to the RSEXEC system which will be distributed to TENEX sites shortly. The following summarizes these improvements.

a. The user interface to the distributed file system has been changed to correct two problems which we feel have prevented more wide spread use of RSEXEC. First, users have pointed

out that the ENTER command (which is used to gain access to the distributed file system environment) was difficult to understand and to use. The source of confusion was that there were two totally different forms of the command depending upon whether the user had a permanent RSEXEC profile (additional information from the user on his first use of ENTER is required to create the profile for him). The ENTER command has been rewritten to prompt first-time users for the additional information necessary for profile creation. The second problem was that the means available to a user for managing his Composite Directory were fairly crude in the sense that all files in all component directories were automatically part of it. (Recall that the Composite Directory is the file directory seen by the RSEXEC user; it consists of the files in the component directories specified in his profile.) As a result, when a user's profile contained a large number of component directories or when the directories themselves were large, construction of the Composite Directory (done as part of the ENTER command) was a very time consuming operation. Furthermore, there were many situations in which it was unnecessary or even inconvenient (i.e., in terms of file naming conflicts) to have all files from all component directories in the Composite Directory. We have improved the situation by giving the user more control over his Composite Directory by allowing him to specify (in his profile, at the time he uses ENTER, or both) that only certain component directories are to be included in his Composite Directory at ENTER time and, in addition, to dynamically add and remove component directories from the Composite Directory during the course of an RSEXEC session.

b. RSEXEC includes features that enable a user to automatically initiate and conveniently control jobs on other TENEXs. In order to give the user more flexibility in dealing with the network, those features have been augmented with a general user TELNET facility.

c. The RSEXEC server program (RSSER) has been modified to maintain a more comprehensive log of its activity. It now maintains a histogram of the various commands it performs, keeps a log of all files transferred into or out of its site including who initiated the transfer, and keeps long term statistics on file transfer throughput.

d. Both the AMES-67 and MIT-MULTICS sites have recently implemented prototype user and server programs that provide the inter-site user-user interaction features of RSEXEC (WHO, WHERE(is), LINK, etc.). As a result of experimentation with these three very different time sharing systems (TENEX, TSO and MULTICS) we, together with Hathaway of AMES and Pogran of MULTICS, have refined the RSEXEC protocol.

2. Expanded TIP-RSEXEC System

One of the known problem areas within the network is that of controlling access both to the network itself and within the various Hosts (see, for example, RFC #6Ø2). One problem in this area is that TIPs themselves provide no controlled access to the network. In order to help solve this problem we are expanding the services provided by the TIP-RSEXEC system (see BBN Reports 2544, 26Ø7). With the TIP group we have agreed upon a course of action designed to allow graceful evolution from the current state (in which anyone who knows the telephone number for a TIP port can gain access to the network through that port) to one in which a TIP user will be required to identify himself by name and password to gain access to the network. Ultimately, when a user first tries to use a TIP he will be connected automatically to an RSEXEC to which he must supply his network ID and password to gain further access to the TIP and network.

As a first step in this evolution we are expanding the TIP-RSEXEC news service to provide news specific to each TIP site in addition to the network-wide news it currently provides. With this addition TIP "administrators" will be able to keep users of "their" TIPs informed of news items peculiar to their installations. While this first step is quite modest in terms of the improvement in service TIP users will see, we feel that it is an important one because its implementation requires us to address most of the technical issues presented by the TIP network

access control procedure suggested above. In particular, the following issues must be addressed:

a. Flexible access control:
There must be a mechanism to control who may make additions or changes to the various site-specific news data bases. For example, to add a news item a user must first identify himself and establish his authority to make an addition. Some users may be able to make additions only to the news for a single site while others may be able to add news for several or all sites. Some, but not all users will be able to grant others the authority to make additions at one, some or all sites.

b. Network identifications and passwords:
The access control procedure will be based on network user IDs and passwords. It would be desirable, for reasons of compatibility, to use an existing network-wide ID scheme. However the only such ID system we are aware of is the NIC ident scheme which we consider to be totally inadequate for this application because of its 3 or 4 character size limitation. (A user's NIC ident could however be included as part of his entry in the ID/password data base.) Our current design is an ID consisting of the user's name (last name and optionally first name and middle initial) and affiliation. To identify himself the user would be required to supply only enough of his ID to disambiguate from that of other users.

c. Reliability:
The system must be available "all the time". This will be particularly important when all TIP users will be required to identify themselves before gaining access to the network. The current approach of replicating the TIP-RSEXEC service on several Hosts seems satisfactory (at present ISI and BBN provide the service).

d. Management of a distributed data base:
Reliablity considerations (see above) require that images of the site news data bases and the network ID/password data base be maintained at all TIP-RSEXEC sites. The technique currently used for maintaining the network news file at TIP-RSEXEC sites (see BBN Report 2607) is adequate for maintaining the site-specific news files to which only additions are made. However it is inadequate for maintaining the ID/password data base on which addition, modification and deletion operations will be performed from each of the several TIP-RSEXEC sites. Clearly, it is impossible to keep all images of such a data base identical at all times. Our goal rather is a technique which guarantees all images to be identical when the "steady state" is reached; i.e., when all modifications have "percolated out to" and have been incorporated in all images. We are making progress in the

design of such a technique.

3. Interprocess Communication and Process Synchronization

We have initiated a study to design and then implement an interprocess communication (IPC) and synchronization facility for processes in a distributed computer environment. The IPC mechanisms currently available on TENEX rely heavily on sharing address spaces among processes and are, therefore, not well suited in situations in which the communicating processes are running on different Hosts. The new facility would initially be available through RSEXEC, and would support both local and remote interprocess communication in a uniform way (i.e., processes running on the same Host would communicate using the same mechanisms as those running on different Hosts).

As part of this work we have co-authored a paper ["Network Considerations in the Design of Operating Systems", E. Akkoyunlu, A. Berstein, R. Schantz] which will appear in a special issue of the IEEE on operating systems. The paper describes and compares previous attempts at designing IPC facilities for use a network environment.

E. TENEX Improvements

1. Improved Reliability and Accessibility

As part of our continued effort to improve the observed reliability of TIP/TENEX usage, we have made a number of

improvements in TENEX's NCP, (Network Control Program). First, we spent three days in a coordinated effort with the BBN-TIP group to capture and diagnose all instances of misbehavior or inconsistency in TIP/TENEX connections.

These experiments revealed several problems: first, there were some timing-and-load dependent inconsistencies appearing in TENEX's connection table because several possible event sequences managed to bypass the connection table interlock.

After this problem was corrected, we found that TENEX was still sufficiently unresponsive to its IMP that the IMP declared TENEX to be dead several times a day, destroying all active connections. The reason for this was that the process responsible for handling input messages was being locked out due to an interlock being set by another process having sufficiently low priority to not run to completion and clear the interlock. To cure this, any process setting the interlock is prevented from dropping to low priority until after the interlock is cleared. This measure has improved, but not cured, the responsiveness problem: further investigations are now in progress to diagnose the problem in detail.

Error reporting has also been improved: when TENEX receives an "IMP going down" message, it types out the notice on all terminals, specifying how soon the IMP is expected to go down, for what reason, and when it is expected back up again.

Two problems were solved which had been causing increased frustration to users who had been detached from their jobs by unreliability of network service. First, under heavy load conditions, users found that they could not gain service on reconnecting to TENEX due to a "drum full" or other overload condition. This meant that even though their jobs were already using system resources, they could not regain control of those jobs. This problem was solved by defining an intermediate load level beyond which ATTACH commands are allowed, but LOGIN commands are not allowed. Thus, detached users can almost always regain connrol of their jobs.

Second, sometimes upon re-attaching to a job the user's terminal would not respond when interacting with an already running program. This was found to be due to the new connection being established via a different Network Virtual Terminal number from that which had been in use before the DETACH occurred. (This problem also existed for local users.) This problem was solved by taking specific note of the fact that I/O had been requested from the controlling terminal (as opposed to another terminal used as an I/O device), and steering this I/O to the new terminal after an attach. Previously, it had not been possible to change this in the midst of a system call.

2. Improvements in Resource Management

Measurement performed on TENEX version 1.31 running under heavy load at BBN with 192k of memory showed the system to be

severely processor limited. It was determined that 30 to 40% of the processor was being devoted to scheduling - clearly an unacceptably high level of overhead. It was with the goal of reducing scheduling overhead that a substantial revision of the TENEX scheduler was undertaken.

The scheduling function in TENEX is divided into two major segments:

a) Wait-list polling--when a process reaches a point in its work where it can no longer make use of the processor it is placed on a list called the waitlist. A pointer to an "activation" routine and, in some cases, an argument to the activation routine, is recorded in a process-indexed table. In addition, the time-of-entry to the waitlist is recorded in another process-indexed table. The activation routine, when called, answers the question "Can this process now make use of the processor?" There are many such activation routines for each of the various "wait-conditions" e.g. awaiting a character from the TTY, waiting for a real-time interval to elapse, etc. The list of waiting processes must be polled periodically (the activation test invoked) in order to discover potentially runnable processes.

   It was found that version 1.31 was polling the waitlist approximately 20 times per second. The BBN system was exhibiting an average waitlist occupancy of 100-120 processes. The cost of this too-frequent polling of so many processes amounted to 10-12% of the processor. TENEX 1.32 has been modified to eliminate some injudicious waitlist scan requests which were a side-effect of a need to reschedule upon queuing a pseudo-interrupt request.

   The primary improvement was a result of noticing that most of the wake-ups were confined to a relatively small number of processes. The ordering of the waitlist was reversed to newest-arrival through oldest-arrival, and the waitlist scan modified to terminate when a process is found which has resided more than 3 seconds on the waitlist. Every 3/4 second the entire list is polled. The average scan depth was reduced from 100-120 to 12-15 and the time consumed to less than 1% of the processor.

b) Balance set management--The responsibility of the balance set manager is two-fold.

1) Determining which processes shall occupy the balance set (the subset of runnable processes which are candidates for control of the processor and whose pages are protected from overlay).

2) Deciding which process next receives control of the processor.

Here again, it was found that the 1.31 scheduler was spending large amounts of time performing these functions (10-15%). The cause of this was, unfortunately, imbedded in the organization of the scheduler data base and was less amenable to repair than was the waitlist manager. Basically, the difficulty lay in the requirement that the scheduler apply an algorithm to per-process data in order to determine the relative priority of a process. This has the advantage of permitting a very flexible priority function at the expense of requiring examination of every process in the balance set, for example, when trying to find a process to run, or examination of every non-balance-set but runnable process when trying to increase the occupancy of the balance set.

The solution adopted was to organize all (balance set and non-balance set) runnable processes into a set of n lists, where n = the number of desired queue levels (presently 5). These lists contain both forward and backward pointers to eliminate the need for searching in list-maintenance operations. The lists are kept in strict priority order, thus a process' relative priority is now determined solely by its position in its RUNLIST. As a result of this change, time spent in the balance set manager has been reduced to under 5% generally. It is recognized that a beneficial aspect of the 1.31 scheduler, namely, the tendency of a process to accrue priority when not running, has been eliminated by the new scheme. Restoration of this desirable feature is possible without incurring the unacceptable overhead of the 1.31 scheduler. This is scheduled for implementation in the near future.

In addition to the above described changes, TENEX 1.32 contains various detail changes to the memory management algorithm, which was seen to be deficient in TENEX 1.31. This whole area of memory management is scheduled for substantial revision in TENEX 1.33. One of the changes currently implemented

has the undesirable side-effect of reducing system responsiveness to interactive requests, such as the EXEC control-C and control-T functions. This change has remained a part of the system because it is highly beneficial in terms of total system throughput. An effort is now being made to eliminate this side-effect prior to system release.

There has been a reorganization of the system measurement tables, and a new table has been added to provide a breakdown of how the system spends its time. Included in this table are measurements of sold-time, idle-time, swap-wait time, etc.

The RSEXEC will be able to collect this CPU utilization cost accounting information from all TENEX systems, making possible the remote diagnosis of system bottlenecks. The data base collected in this fashion should be helpful in ARPA's resource management decision-making.

3. Improved Security and Protection

During this quarter, a hole in TENEX system integrity was discovered by the RISOS (Research in Secured Operating Systems, Lawrence Livermore Labs) effort. In checking the validity of their discovery, the BBN system was accidentally crashed. Fortunately, system programmers were on hand at the time this occurred, and the system was patched to solve the problem within a half hour of the crash. This patch was distributed to the other network sites that same evening.

As a result of the system crash which occurred during this test, new procedures are being set up to allow both RISOS and TENEX service to proceed cooperatively and productively.

4. Peripheral Processor

Work on our ARPANET mini-host PDP-11 peripheral processor has continued this quarter, and it is now connected to the network as host 5, BBN-11X. A second identical PDP-11 system is now in house: it will be used for checkout and development work while the first system is doing production work. We expect to adapt an existing PDP-11 operating system to serve our peripheral processor function, but the decision has not yet been made between the ANTS (ARPAnet Terminal System) system developed by the University of Illinois and the ELF system developed by Speech Communications Research Lab. The first nonlocal deliveries of both systems are scheduled for first quarter 1974, and we expect to learn a good deal about both systems by observing that process.

In the meantime, we continued development of our cross-network debugger in order to support our modifications and adaptation of the selected system. The debugging protocol is specified and low-level routines for driving that protocol from a TENEX user program have been coded. The mini-monitor required to support that protocol and perform the debugging functions within the PDP-11 is complete and operational: it uses about 256 words

of core.

We are offering this program and protocol to any group which might be interested in cross-network loading and debugging of network access machines. Keeping the debugging language, action routines, tables, and symbol tables in a large network TENEX server permits great performance advantages, while requiring very little core occupancy on the network access machine.

5. JSYS Traps

During this quarter we implemented the JSYS trap mechanism (see BBN Proposal P73-CSC-14A) for TENEX. This is a mechanism which enables one process to define and control the virtual machine seen by other processes. We expect to distribute the code for JSYS traps shortly after the TENEX 1.32 release.

The controlling process does this by specifying that it wishes to "monitor" (trap) selected system calls (JSYS's) when other (inferior) processes attempt to execute them. When a monitored process attempts execution of one of the JSYS's specified it is suspended (frozen) and the monitoring process is notified (via a pseudo-interrupt signal). After gaining control, the monitoring process is free to take any action it sees fit; it may choose to perform the JSYS "for" the other process, to allow the process to perform the JSYS itself, to first modify the parameters for the JSYS call and then allow the process to resume

execution of it, etc. If the monitoring process chooses to handle the JSYS trap by allowing the trapped process to resume execution of the JSYS, the trap will "percolate" up the process hierarchy to the next process (if any) monitoring execution of that JSYS. If (when) there are no further processes monitoring that JSYS in the hierarchy, the trapped process is dispatched to the standard system code for that JSYS. Should the monitoring process choose to handle the JSYS trap by executing the JSYS for the trapped process, it itself is subject to any traps for the JSYS which superior processes may have set.

To implement JSYS traps the dispatch vector, formerly a resident page shared by all processes in the system, was made process private. Trapped processes have a modified dispatch vector in which entries corresponding to trapped JSYS's point to a "trap and interrupt" routine and those for untrapped JSYS's point to the standard monitor routines for those calls. We saw two ways to make the dispatch vector private:

1. Make a minor modification to the JSYS instruction so that it uses a page in the process private area rather than one in the public area as its dispatch vector;
2. Leave the JSYS instruction unmodified but when a trapped process is running set the monitor map word for the JSYS dispatch vector to point to a page in the per process area and cause the pager to map the resident monitor (see above).

The primary advantage of the second approach is that it allows the trapping software to run on the existing hardware at all TENEX installations. Its disadvantages are two: slightly slower execution for trapped processes resulting from the mapping

operation on each reference to the resident monitor and severe constraints on the software for (planned) dual processor configurations resulting from limitations of both the paging device and PDP-1Ø processor. The current implementation provides for both alternatives, allowing each installation to choose one at "system generation time".

The JSYS trap mechanism will be used by RSEXEC to provide the distributed file system environment, currently available only at the command language level, to executing programs. RSEXEC will do this by trapping execution of file system JSYS's by processes running "under" it. Operations involving the manipulation of remote files will be "slaved" across the network to a cooperating RSSER process while those involving local files will be handled locally either by RSEXEC on behalf of the trapped process or directly by the trapped process itself.

Although motivated by RSEXEC requirements, we feel the JSYS trap mechanism to be an important and powerful addition to TENEX that will be useful in applications which require a controlled execution environment.

6. Raw Network Messages

Two new system calls have been added to TENEX on an experimental basis to provide a privileged user with access to "raw network messages", i.e. messages which conform to 1822-type host-IMP protocol. This permits a user to create his own

experimental host-to-host protocol and write his NCP in user code. These JSYS's are privileged, and require the "NETWIZ" (network wizard) user capability bit to be enabled. They are "Send-to-IMP" (SNDIM) and "Receive-from-IMP" (RCVIM).

In order to permit non-interfering network access by the current NCP and the SNDIM/RCVIM facility, these two facilities are assigned mutually exclusive ranges of message identifiers. In order for a message to satisfy SNDIM/RCVIM restrictions, it must either:

a. Have the "to IMP/from IMP" bit set

or

b. Have a 12-bit message identifier number greater or equal to 2200 octal. This corresponds to links above 71 decimal in the original 1822 specification.

We are now using this facility for cross-network PDP-11 loading and debugging, and the BBN-TIP group is using it for TIP monitoring and statistics.

7. New TELNET Protocol

In accordance with the January 1, 1974, deadline for implementing te new TELNET protocol, we have installed a TELNET server into TENEX which understands option negotiation and is willing to negotiate modes such as echo, binary, supress goahead, and character-at-a-time. We were able to make the implementation backward compatible with the old TELNET protocol, so both old and new TELNET user programs may utilize this server.

In addition, we have implemented the earlier specification change that output buffer clearing is accomplished by using the INS-data mark protocol facility. This change makes it possible for a TIP user to stop unwanted output instantly by typing two control-C's to TENEX.

8. Automatic Dialler

A considerable amount of effort has been spent during this quarter on hardware and software additions to the BBN-TENEX-A system to drive an automatic dialler attached to a dataset. This device is being used by the staff responsible for TIP development, to test modem ports on the TIPs throughout the ARPANET. Due to the hardware involved, this particular facility will not be useful to other sites at present, but it would be straightforward to standardize and distribute these changes should standard hardware become available at this and other TENEX sites.

9. KI-10X System

In preparation for TENEX system operation on the two KI-10 processors on the network, code has been merged with the standard TENEX sources to conditionally assemble for KI-10 or KA-10 systems. This merging has not been completed, due to the parallel development of KA-10 TENEX and the lack of a KI-10 system at BBN for testing purposes. It is expected that this will be tested in early 1974 at the USC-ISI installation, and a

working KI-1Ø system will be distributed shortly after the TENEX system 1.32 release.

1Ø. PAL11X Cross-assembler

The TENEX PAL11X cross assembler was adapted from the PALX assembler at Massachusetts Institute of Technology, Artificial Intelligence Labs. PALX was converted to run under TENEX, and to produce relocatable binary output which is directly loadable by either PDP-1Ø linking loader. Three pseudo-ops were added for the definition of external symbols and entry points, and the listing format was altered to indicate such definitions. A temporary TENEX subsystem, SAVBIN, was written to convert the PDP-1Ø core image into PDP-11 absolute loader format.

Our peripheral processor work is now done by assembling, loading and formatting PDP-11 core images on TENEX. These programs are then sent across the network for execution on the PDP-11.

11. New BSYS

A new revision of the "Backup System" (BSYS) program has beeen developed and is in the final stages of checkout. The main improvements have been in the area of reliability. The program can now make use of redundant information stored on the tape to reconstruct tapes which suffered errors during writing.

Previous improvements made by BBN and SRI-ARC have been

integrated into the new version. Several internal changes have been made such as recognizing industry-compatible end-of tape marks as well as BSYS style EOT's.

Steps are being taken to improve the human engineering of the program. One such change is simply clearing the terminal input buffer a half second after an operator typing error is discovered. This helps prevent BSYS from interpreting type-ahead as erroneous commands.

12. EXEC

The EXEC has been modified so that it never outputs a code Ø37 meaning TENEX END-of-line. The ASCII standard CR-LF sequence is used instead. This means that files written by the EXEC are now compatible with non-TENEX sites.

The deferred control-C interrupt mechanism has been implemented in the EXEC. This makes it possible to include control-C's in the input stream. The interrupt does not effect the process until the control-C is read from the input buffer. Thus, type-ahead after control-C's is preserved.

13. Documentation

BBN-TENEX was the featured site in the December issue (#1Ø) of the ARPANET News. The article described our various research interests and efforts, and the network service facility provided by BBN-TENEX.

The new JSYS manual, which included descriptions of all new monitor calls and a greatly expanded section on network use, was distributed to all sites and made available to all users.

Printing was completed on our new TECO and PAL11X manuals and they will be distributed shortly. The new TECO manual is a great improvement over previously available documentation: it contains a one-hour self teaching course on basic use of TECO, in addition to a complete and detailed reference manual for the language.

We also published RFC#594, Speedup of Host-Imp Interfaces, which describes our experimental results obtained in speeding up the BBN-TENEX and BBN-TENEXB distant host interfaces to around 300K bits/sec. The experiment was successful in every way, resulted in valuable performance improvement, and is recommended to all ARPANET hosts.

IV. LANGUAGES

A. LISP

1. Sysout File Changes

In the area of swapping or overlaid LISP, we have written a new MAKESYS/SYSOUT which can recover the fork structures of the new system. It quickly became obvious that we should include the "runnable sysout" feature in the new code. Runnable sysouts allow the user to run a sysout from the EXEC level, or more generally by GET'ing the file into a fork and starting at the entry vector as with any ordinary TENEX image file. Previously it was necessary to remember the name of the right makesys file, run it, and give the name of the sysout file to the function SYSIN. The sysout file now contains the name of the right makesys, and normally it will find the makesys, merge the images, and run. In the event that the makesys has been renamed, deleted or overwritten since the sysout was made, the program will print the old name and ask the user for a new name. After a positive check that the name supplied is indeed the compatible makesys, it will carry on as before, but the program will write the new name on the sysout file so that the question need not be asked again the next time the sysout is run. In the odd case where the user is running someone else's sysout and does not have write access, nothing can be done, and the question will have to be answered every time until a new sysout is made or a write-enabled user

runs the old one and answers the question correctly. The LISP function SYSIN still provides the capability to "chain to" a sysout from within LISP, under program control.

Runnable sysouts make utility subsystems feasible in LISP, since they can be put in the SUBSYS directory without making each one a separate makesys to achieve the standard runnable form. The latter practice caused loss of potential page sharing and required either that such programs be loaded in stripped-down LISP's, depriving their users of the convenience of escaping to regular LISP work and perhaps reprogramming the utility a little to suit themselves, or else, if a full LISP was provided, each one required 300 file pages or more. As a consequence, LISP has never been used for this purpose and LISP programs remained the private preserve of LISPers.

The new sysout/makesys file format is completely sequential, with no "holes" or non-existent file pages to cause problems with sequential devices such as net connections. (FTP fills out such holes with zeroes, inflating the file). Finally, the new sysouts will start up much faster. Page protection is more accurately specified, avoiding unnecessary trapping on the initial writes. It is no longer necessary to map in all the pages of the makesys, and initialize the system, just to call SYSIN, which then overlays many of the pages with its own; by running the sysout first, only those makesys pages which are needed are brought in.

2. User Defined Data Types

We have completed the initial implementation of user defined data types. This allows the user to specify data structures containing only what he needs in a more compact form, and with faster access time, than if list structures were used. The storage allocation and garbage collection of user defined data types is the same as for any regular LISP data type, and is therefore just as efficient. This data definition facility is not completely general; for example, not all the data types that currently exist in LISP are user defineable. However, it provides a significant increase in the data definition capabilities of LISP.

The scheme used provides the ability to define fixed length data types composed of fixed length components. Each component may be a pointer, an integer, a floating point number, or some number of bits. In addition to the basic SUBRs described below, there also exist facilities to easily define functions to create a datum (analogous to CONS) and to set and extract components (analogous to RPLACA and CAR). We have also provided a way to print structure definitions and data of a user defined type. For debugging purposes the user can, at any time, retrieve the format of an item or the name of the structure to which it belongs.

SUBRS For User-Defined Data Types

DEFTYPE[nwrds; nptrs]

Defines a new data type such that each object of that type has nwrds words and nptrs pointers within those nwrds. If there are not available types, an attempt is made to find a data type which has; (1) the proper number of pointers, (2) at least the needed number of words, and (3) has been marked as "not in use" via TYPESTATUS. The data type number selected is returned as the value of DEFTYPE.

GETNPTRS[typen]

Returns the number of pointers in an object whose type is typen.

GETNWRDS[typen]

Returns the number of words in an object whose type is typen.

NALLOC[typen]

Allocates an object of type typen, clears its contents to zero and returns a pointer to it.

USERCONS[typen; p1; p2; ... ; pn]

Similar to NALLOC but fills in the pointers within the object with the arguments p1, p2, ... , pn. If too many arguments are given the extra ones are ignored. If too few arguments are given, NILS are supplied.

TYPESTAT[typen; status]

Sets the status of data type typen to status and returns the previous status. If status is NIL, the status of typen is left unchanged. The meaning of the status is as follows:

Status = Ø - the data type has never been used.

= 1 - the data type is currently in use, the status is set to 1 by DEFTYPE when it selects a type.

= 2 - the type was previously in use, but is now available for selection by DEFTYPE if no other types are available.

Actually, a status value of 1 is the only value DEFTYPE checks for. It is advisable to set the status to 2, instead of Ø, when freeing a data type, simply to distinguish between "not in use" and "has never been in use".

3. West Coast Meetings

During November we visited SRI and Xerox PARC. At SRI we described, demonstrated, and answered questions about the multiple environment system. We also learned about their particular needs and requests for additional features in LISP which we will try to include in future systems.

At Xerox PARC we worked with Warren Teitelman on the modifications to the entire system required by the new stack discipline for multiple environments. We created a second experimental version of the multiple environment system with the full LISP system with the exception of te compiler.

We have begun work on modifying the compiler for multiple environments. This portion of the task has been somewhat more difficult than was anticipated because the scheme for handling PROG and LAMBDA bindings described in Progress Report #1Ø was incomplete and has required more work than we originally estimated.

B. Automatic Programming

From the outset, our efforts in Automatic Programming

research have been directed toward problems of automatic synthesis on the grounds that eventual automation of increased fractions of the programming process is essential to the achievement of much needed software productivity gains.

To review briefly, last quarter we developed:

1. Techniques for automatic synthesis of data representations (based on an axiomatic treatment of data structures),

2. Techniques for automatic synthesis of "loop predicates" (inductive assertions required for program verification proofs), and

3. Techniques for automatic derivation of "invariance" assertions in programs (based on a theory of well-founded, partially ordered sets).

These techniques are each logic-based --- they depend upon logical formalisms for their expressivity and upon mechanical inference systems for their effectiveness.

Accumulating experience with the performance properties of logic based mechanical inference systems seems to reveal limitations. In particular, the complete, uniform search procedures (such as resolution) seem to exhibit more or less irrepressibly wasteful search behavior. They seem to work well only on examples of modest size in micro-worlds of limited complexity, and we have come to doubt whether they can be improved to give the muscle required to deal with many problems that arise in practice.

As a result, attention (both in our own efforts and

elsewhere) has recently been directed at discovering mechanical inference techniques that avoid blind combinatorial search as much as possible, substituting instead sophisticated guidance mechanisms that channel the available problem solving energy into profitable and relevant avenues.

For example, the "programmed trial-and-error" techniques, embodied by programs written in languages such as PLANNER, CONNIVER and QA4, exhibit one approach to ths goal. Here, the objective is to program significant knowledge about what problem solving aproaches to try in particular classes of situations, and in what order to try them.

Our own efforts this quarter have been directed at the discovery of new methods for program synthesis that avoid wasteful search characteristic of the logic-based techniques. To find new ideas, we collected and analyzed protocols of human program synthesis behavior. The results are reported in "Observations and Hypotheses About Program Synthesis Mechanisms" [5].

Some of the results are rather interesting. Among the program synthesis mechanisms we hypothesize people use are:

1. Refinement of abstract programs into concrete programs (by filling in relevant details).
2. Extensive use of model domain knowledge to provide strategies, representations, constraints and examples.
3. Direct translation of problem domain knowledge into programs.
4. Program synthesis by transformation of known programs.
5. Induction and generalization from examples.

The report [5] gives further information about other mechanisms (such as the use of plans, techniques for synthesizing data representations, debugging and patching, synthesis by case analysis and fusion, and others). Limited space precludes any deep discussion of these subjects.

As a result of having studied and analyzed protocols, we are armed with a stock of fresh new ideas for building program synthesis systems.

Our work for the next quarter is aimed at learning how to build program synthesis systems based on two of the above new mechanisms: (a) abstract program refinement, and (b) the encoding and use of model domain knowledge.

V. SPEECH COMPRESSION

A. Alternate Transmission Parameter Sets

The all-pole model used in a linear predictive system has a transfer function

$$H(z) = \frac{1}{A(z)} \triangleq 1 + \sum_{n=1}^{\infty} h_n z^{-n},$$

where the inverse filter A(z) is given by

$$A(z) = 1 + \sum_{n=1}^{p} a_n z^{-n}.$$

Given below is a list of possible sets of parameters for characterizing uniquely the linear predictive filter H(z):

(1) Impulse response of the inverse filter A(z) i.e. predictor coefficients a_n, $1 \le n \le p$.

(2) Impulse response of the all-pole model h_n, $1 \le n \le p$, which are easily obtained by long division. Note that the first p coefficients uniquely specify the filter.

(3) Autocorrelation coefficients of $\{a_n\}$,

$$b_i = \sum_{j=0}^{p-|i|} a_j a_{j+|i|}, \quad a_0 = 1, \ 0 \le i \le p. \tag{1}$$

(4) Autocorrelation coefficients of $\{h_n\}$, R_i, $\emptyset \le i \le p$

(5) Spectral coefficients of A(z), P_i, $\emptyset \le i \le p$, (or equivalently spectral coefficients of H(z), $1/P_i$)

$$P_i = b_o + 2 \sum_{j=1}^{p} b_j \cos \frac{2\pi ij}{2p+1}, \quad \emptyset \le i \le p, \tag{2}$$

where b_i are as defined in eq. (1). In words, P_i is obtained from b_i through a discrete Fourier transform.

(6) Cepstral coefficients of A(z), c_n, $1 \le n \le p$, (or equivalently cepstral coefficients of H(z), $-c_n$)

$$c_n = \frac{1}{2\pi} \int_{-\pi}^{\pi} \log[|A(e^{j\theta})|^2] \, e^{jn\theta} \, d\theta. \tag{3}$$

Using the results in (Gold and Rader, 1969), eq. (3) can be manipulated to obtain the following recursive relation between c_n and a_n:

$$c_n = a_n - \sum_{m=\emptyset}^{n-1} \frac{m}{n} \, c_m \, a_{n-m}, \quad 1 \le n \le p. \tag{4}$$

(7) Poles of H(z) (or equivalently zeros of A(z)).

(8) Reflection coefficients k_i, $1 \le i \le p$, (or partial correlation coefficients and simple transformations thereof, e.g. area coefficients).

Some of the above sets have p+1 coefficients while others have only p coefficients. However, for the latter sets the signal energy needs to be transmitted, thus keeping the total number of parameters as p+1 for all the cases. Although the above sets provide equivalent information about the linear predictor, their properties under quantization are different. Certain aspects of the sets (1), (4), (7) and (8) have been studied in the past (Atal and Hanauer, 1971; Itakura and Saito, 1972). Our purpose in the last quarter has been to investigate the relative quantization properties of all these parameters with a particular emphasis on the reflection coefficients.

B. Preprocessing Methods

Before we discuss the quantization properties of the different parameters we should mention that such properties can be improved by proper preprocessing which is later undone during synthesis. For each set of parameters (1-8 above), we have observed that the short-time spectral dynamic range of the speech signal is the single most important factor that affects the quantization properties. We use two methods of preprocessing speech to reduce the spectral dynamic range and thereby to improve the quantization properties (Makhoul and Viswanathan, 1973). In the first method, optimal (linear predictive) preemphasis is applied to the speech signal which reduces the spectral

dynamic range by reducing the general spectral slope. The second method, called the SIGMA method, involves multiplying the impulse response of the inverse linear prediction filter by a decaying exponential, which increases the pole bandwidths, resulting in a reduction of the spectral dynamic range. Preprocessing by either of these methods can be done after the linear prediction analysis, so that it can be viewed as part of encoding.

C. Quantization Properties

We have investigated experimentally the quantization properties of the sets of parameters discussed above, with and without preprocessing of the speech signal. The absolute error between the original and the quantized log spectra was used as a criterion in this study.* A summary of the results is provided in what follows. For the purpose of quantization, two desirable properties for a parameter to have are: (a) boundedness with known upper and lower bounds and (b) filter stability upon quantization. Only the poles and the reflection coefficients possess both of these properties. We have mainly concentrated on the quantization properties of the reflection coefficients.

The parameters (1) and (2) are highly susceptible to

---

*We have assumed here that a good spectral match is necessary for synthesizing speech with good quality.

causing instability of the filter upon quantization. Positive-definiteness of autocorrelation coefficients [(3) or (4) above] is not ensured under quantization, which would lead to instabilities in the linear prediction filter. An attempt to synthesize speech with quantized autocorrelation coefficients (4) resulted in distinctly perceivable "clicks" in the synthesized speech. Thus, the parameters (1) through (4) can be used only under minimal quantization, in which case the transmission rate would be excessive.

In the experimental investigation of the spectral and cepstral parameters, we found that these parameters are generally inferior to the reflection coefficients. The spectral method often yields results comparable to those obtained by quantizing the reflection coefficients. However, for the cases when the spectrum consists of one or more sharp peaks (narrow bandwidths), the effects of quantizing the spectral coefficients cause some of the points in the approximate spectrum to become negative. Preprocessing the speech signal by the SIGMA method remedies this situation, but the spectral deviation in these regions turns out to be relatively large. Quantization of cepstral parameters can also lead to instabilities. As before, with a proper preprocessing by the SIGMA method, stability is restored but at the expense of increased spectral deviation.

Considering the quantization of poles, it is well-known

that pole bandwidths may be quantized less accurately than pole frequencies. Pole frequencies lie in the interval $(\emptyset, F_s/2)$ where F_s is the sampling rate. This means that the quantization level becomes relatively large leading to a coarse quantization, unless poles are identified as an ordered set of formants in which case smaller ranges can be chosen for quantizing the individual formant frequencies. However, the problem of identifying the ordered formants is computationally complex. Also, computing the poles requires finding the roots of a p-th order polynomial.

The reflection coefficients for a stable filter are all bounded above and below by 1 and -1 respectively. This, of course, makes it easy to ensure stability under quantization. It should be clear that the same is true for any parameter that is a simple transformation of the reflection coefficients (such as the area coefficients).

It has been observed by some researchers that the first few reflection coefficients are the most sensitive to the effects of quantization. While this is true, we believe that the high sensitivity is not due to the fact that these reflection coefficients are the leading ones but because on the average they assume magnitudes closer to 1 than the others. In order to test this conjecture experimentally, we defined the following spectral sensitivity measure:

$$\frac{\partial S}{\partial k} \triangleq \lim_{\epsilon \to \emptyset} \frac{1}{N+1} \sum_{j=\emptyset}^{N} \frac{1}{\epsilon} \left| S(k_1, \ldots, k_i+\epsilon, \ldots, k_p; \omega_j) - S(k_1, \ldots k_p; \omega_j) \right| \quad (5)$$

where $S(\cdot,\omega_j)$ is the spectral value in decibels at the frequency ω_j. The quantity after the summation symbol in (5) gives the absolute deviation in spectral amplitudes due to a perturbation in the ith reflection coefficient. Experimentally, $\frac{\partial S}{\partial k_i}$ is computed by using a sufficiently small value for ϵ. We have found that the sensitivity curve $\frac{\partial S}{\partial k_i}$ versus k_i has the following properties:

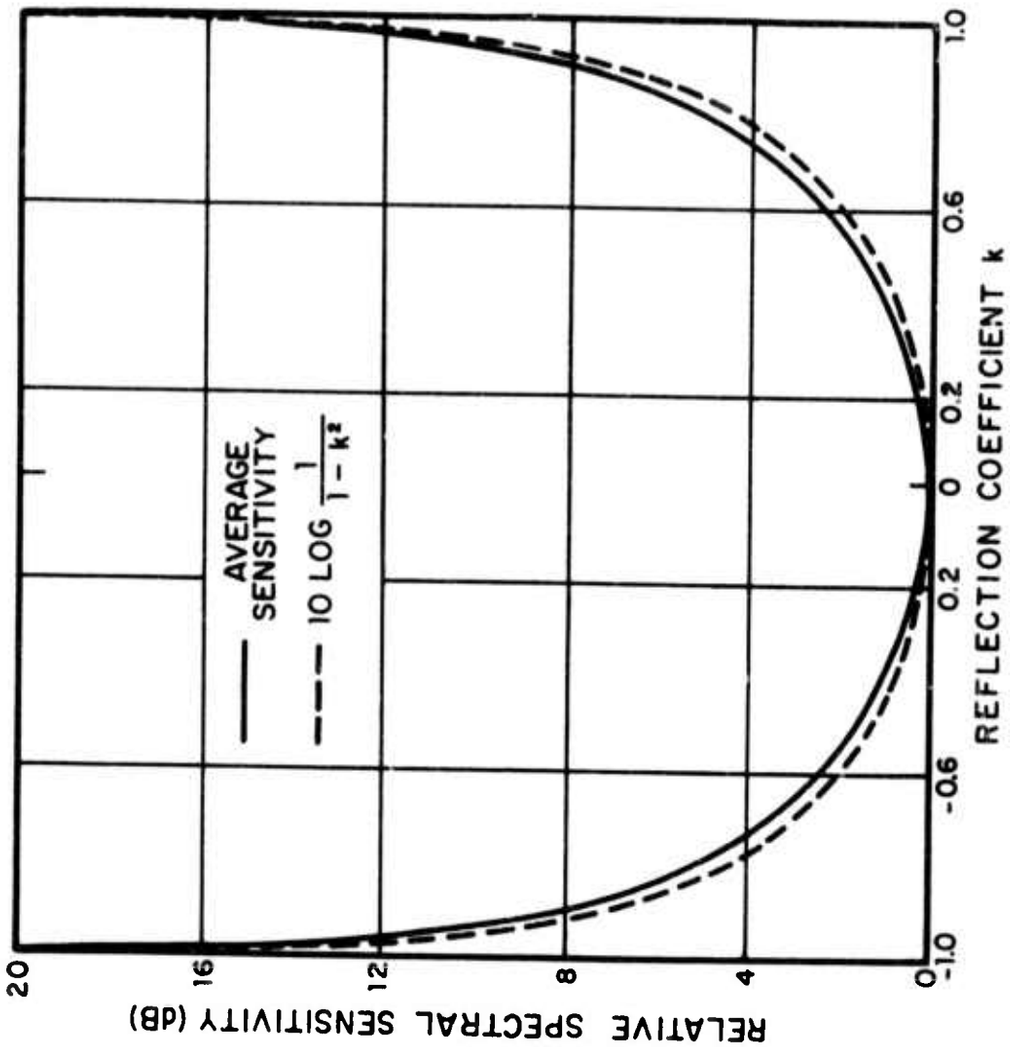
(i) It has the same shape irrespective of the index i and irrespective of the values of k_n, $n \neq i$, at which the sensitivity is computed.

(ii) The sensitivity curve is U-shaped. It is even-symmetric about $k_i=\emptyset$, and has large values when the magnitude of k_i is close to 1 and small values when the magnitude of k_i is close to zero.

Owing to the properties of the sensitivity curve of the reflection coefficients, it is meaningful to compute an averaged sensitivity function

$$\frac{\partial S}{\partial k}(x) = \frac{1}{pT} \sum_{t=1}^{T} \sum_{i=1}^{p} \frac{\partial S}{\partial k_i}\Big|_{k_i=x} \qquad (6)$$

where t refers to the number of the analysis frame (time averaging). The averaged sensitivity function for a representative speech sample is shown plotted as the solid curve in Fig. 1. In this plot the sensitivity values are given in decibels relative to the sensitivity at $k=\emptyset$.


AVERAGE SENSITIVITY
10 LOG $\frac{1}{1-k^2}$
RELATIVE SPECTRAL SENSITIVITY (dB)
20
16
12
8
4
0
-1.0
-0.6
-0.2
0
0.2
0.6
1.0
REFLECTION COEFFICIENT k

D. Optimal Quantization of Reflection Coefficients

In view of the sensitivity properties of the reflection coefficients discussed in the previous section, it is clear that linear quantization of the reflection coefficients is not satisfactory, especially when some of them take values close to 1 in magnitude. However, nonlinear quantization procedures with better characteristics can be developed. A nonlinear quantization of a reflection coefficient is equivalent to a linear quantization of a different parameter that is related to the reflection coefficient by a nonlinear transformation. Denoting the transformed coefficients as g_i, we have

$$g_i = f(k_i) \tag{7}$$

where $f(\cdot)$ is the underlying nonlinear mapping. Using the concept of spectral sensitivity, we have developed a method to determine an optimal nonlinear transformation. In our method, a mapping $f(\cdot)$ is said to be optimal if

$$\frac{\partial S}{\partial g_i} = \frac{1}{L} = \text{a constant} \tag{8}$$

Writing formally,

$$\frac{\partial S}{\partial g_i} = \frac{\partial S}{\partial k_i} \frac{dk_i}{dg_i} = \frac{\partial S}{\partial k_i} \Big/ \frac{df(k_i)}{dk_i} \tag{9}$$

Thus, $f(\cdot)$ is optimal if

$$\frac{df(k)}{dk} = L \frac{\partial S}{\partial k} \tag{10}$$

It is clear from (8) that for the optimal case, the parameters g_i have a flat sensitivity characteristic. This implies that a linear quantization of the g_i's would lead to a minimal spectral deviation.

In the rest of this section, we derive a transformation that satisfies the optimality condition (10) approximately. An experimental fitting of the averaged sensitivity curve in Fig. 1 has revealed that the function $2/(1-k^2)$ approximates the sensitivity function reasonably well (to within a multiplicative constant), as shown in Fig. 1. Thus, in view of (10), the approximately optimal transformation is given by

$$\frac{df(k)}{dk} = \frac{2}{(1-k^2)} \tag{11}$$

Integrating (11) we obtain

$$f(k) = \log \frac{1+k}{1-k} \tag{12}$$

The transformation given in (12) is the logarithm of the ratio of area functions A_i, since the latter are defined by

$$A_i = A_{i+1} \frac{1+k}{1-k}, \; A_{p+1} = 1, \; 1 \leq i \leq p. \tag{13}$$

Thus, we have shown that the logarithms of the area ratios provide approximately an optimal set of coefficients for quantization. Our experimental investigations into the quality of the synthesized speech also indicate that the logarithms of the area ratios possess good quantization properties.

Also in the last quarter, a paper entitled "Adaptive Preprocessing for Linear Predictive Speech Compression Systems," was presented at the 86th meeting of the Acoustical Society of America (Makhoul and Viswanathan, 1973).

REFERENCES

1. B. Gold and C.M. Rader, Digital Processing of Signals, McGraw-Hill, New York, 1969.

2. B.S. Atal and S.L. Hanauer, "Speech Analysis and Synthesis by Linear Prediction of the Speech Wave," J. Acoust. Soc. Am., vol. 50, pp. 637-655, 1971.

3. F. Itakura and S. Saito, "On the Optimum Quantization of Feature Parameters in the PARCOR Speech Synthesizer," Conference Record, 1972 Conf. on Speech Comm. and Processing, Newton, Mass., pp. 434-437, April 1972.

4. J. Makhoul and R. Viswanathan, "Adaptive Preprocessing for Linear Predictive Speech Compression Systems," Presented at the 86th meeting of the Acoustical Society of America, Los Angeles, Oct. 30-Nov. 2, 1973 (Also written as NSC note 5).

5. T.A. Standish, Observations and Hypotheses about Program Synthesis Mechanisms, BBN Automatic Programming Memo 9, 19 December 1973.