DOCUMENT RESUME

ED 163 971 IR 006 644

AUTHOR Laddaga, Robert; And Others
TITLE Research on Uses of Audio and Natural Language Processing in Computer-Assisted Instruction--Third Year Report. Technical Report No. 289.
INSTITUTION Stanford Univ., Calif. Inst. for Mathematical Studies in Social Science.
SPONS AGENCY National Science Foundation, Washington, D.C.
PUB DATE 23 Sep 77
GRANT SED-74-15016 A02
NOTE 89p.; For related documents, see IR 006 644-647

EDRS PRICE MF-$0.83 HC-$4.67 Plus Postage.
DESCRIPTORS Annual Reports; *Artificial Speech; Beginning Reading; *Computer Assisted Instruction; *Computer Programs; Curriculum Development; *Educational Research; Elementary Education; Higher Education; Logic; Mathematics Instruction; Programing Languages; Set Theory
IDENTIFIERS *Computer Generated Speech; *Natural Language Processing

ABSTRACT

Research carried out during the year focused on meeting project objectives in three main areas: computer-generated speech, complex teaching programs with audio, and teaching reading with audio. Work on computer-oriented speech was concerned with improving the facilities and procedures for utilizing the speech system software and the Micro Intoned Speech Synthesizer ("MISS machine"), as well as the continued development and improvement of sentential synthesis through intonation contouring with word concatenation. In the three complex teaching programs studied, work included the completion of the writing of audio and display only versions of lessons in a portion of the logic course, improving the interface with curriculum and lessons for the proof theory course. In the area of teaching beginning reading, a study in which three systems of computer-generated speech were compared to each other and a human-voice control on the task of producing individual letter sounds was designed and conducted with a group of first graders as subjects. A comparison of the three systems on a more complete list of sounds was carried out with fifth grade students. Experimental objectives, procedures, and results are detailed for each area, and a bibliography is provided. (BBM)

RESEARCH ON USES OF AUDIO AND NATURAL LANGUAGE PROCESSING IN COMPUTER-ASSISTED INSTRUCTION--THIRD YEAR REPORT

by

Robert Laddaga, Arvin Levine, and Patrick Suppes

NSF Grant No. SED-74-15016 A02

TECHNICAL REPORT NO. 289

September 23, 1977

PSYCHOLOGY AND EDUCATION SERIES

INSTITUTE FOR MATHEMATICAL STUDIES IN THE SOCIAL SCIENCES

STANFORD UNIVERSITY

STANFORD, CALIFORNIA 94305

Table of Contents

List of Figures

List of Tables

Acknowledgments

The authors wish to thank the following people for their contribution to this report: Lee Blaine, Inge Larsen, William Leben, Lawrence Zaven Markosian, James McDonald, Teri Pettit, William Sanders, and Wilfred Sieg.

For their contributions to the research conducted at the Institute and described in this report we also thank: Barbara Anderson, George Black, Edward Bolton, Scott Daniels, Mark Davis, Thomas Dienstbier, David Ferris, Michael Hinckley, Vladimir Lifschitz, Ingrid Lindstrom, Sten Lindstrom, Ronald Roberts, Marguerite Shaw, Richard Thompson, and Mario Zanotti.

In addition to the above we thank: Jon Allen, Avron Barr, Clarence Francois, Marty Hargrove, Sherry Hunnicutt, Dorothy Huntington, Dennis Klatt, William Rybensky, Elizabeth Raugh, Earl Schubert, Carol Simpson, and the staff at the Willow School in Menlo Park, for their help with the experiment on the use of audio in CAI in initial reading.

The work described in this report was supported by the National Science Foundation under NSF Grant No. SED-74-15016 A02.

1 Project Objectives

In the renewal proposal submitted to the National Science Foundation in 1976, project objectives were summarized under three main headings. We quote from the abstract of the proposal.

> 1. Computer-generated speech. The previous research has resulted in the development of the MISS system, which generates speech efficiently from digitally stored parameters. Proposed research will improve the quality and efficiency somewhat, and will concentrate on the development of methods of prosodic manipulation.
>
> 2. Complex teaching programs with audio. Previous research has produced college-level mathematically-based courses such as logic and set theory. New research will improve the language of mathematical proofs and apply the computer-generated audio to various tasks of describing and explaining the material to the students.
>
> 3. Teaching reading with audio. Previous research has resulted in the use of computers in elementary reading using audio. The proposed research will compare the MISS-produced audio with that of four other synthesis techniques.

In this section we comment briefly on these objectives and the work done to reach them. In the following we report fully on the research conducted in these three areas.

1.1 Computer-generated Speech

The work carried out in the past year of the grant in the area of computer generated speech has had two principal focuses: improving the facilities and procedures for utilizing the speech system software and the Micro Intoned Speech Synthesizer ("MISS machine"), and, continued development and improvement of sentential synthesis through intonation contouring with word concatenation.

For the goal of improving the speech software, we modularized the user procedures for accessing speech. There is now a bi-level structure where the user need only be concerned with the library of programs available in the "upper" level. The lower level is shared by all users and contains the lexicon of stored sounds (words and phrases) as well as the low level routines for accessing the lexicon. The main word lexicon (named "English") was enlarged by the recording and analysis of 2,00 new words. We investigated techniques relating to the compression of sound data and possible interactions between compression and prosodic manipulation.

The focus of the work in intonation synthesis has been fairly linguistic. We have developed practical methods for utilizing prosodic features so that the utterances will have a natural feeling to the listener.

In particular, there are several specific questions which we have pursued. Within the context of the autosegmental hypothesis[1] which we have been using for declaratives, we have conducted preliminary examination of pitch contours in question sentences. A preliminary modification of our pitch assignment algorithm now accommodates both declaratives and questions, although the experiments we have done suggest that further work to make the assignment procedure more general needs to be done.

We also studied duration assignment and have refined and tested our procedures for this component of intonation. In particular, we conducted an experiment to compare our duration assignments to observed utterance lengths which yielded mixed results due to a washout effect.

[1] Cf. Goldsmith (1975), Leben (1975), Levine (1976).

Another experiment building on the first was more useful, establishing a link between text structure and relative utterance speed.

Finally, we have been increasing our understanding of syntactic bracketing of surface strings which are vital to our intonation assignment procedures. We have compared several linguistically justified systems to see which hold the greatest promise in relation to prosodic manipulation.

1.2 Complex Teaching Programs with Audio

1.2.1 Logic Course

During the past year the work on writing audio and nonaudio (display only) versions of the lessons in the PASS portion of the logic course was completed. Also generated were synthetic prosody versions of each lesson having an audio version. A number of experiments, including several examinations of student preference for audio or nonaudio modes, were performed during the winter and spring quarters of the 1976-77 academic year. These experiments are currently being analyzed, with results forthcoming in proposed articles and technical reports.

1.2.2 Set Theory Course

Work in the set theory course this year has concentrated on improving the interface with the student at the terminal by: 1) introducing the capacity for producing audio messages; 2) writing an online introduction to the course and the proof checker using audio messages; 3) incorporating a help system, also using audio, to provide online assistance with administative or course-content difficulties and questions; 4) improving the theorem prover, and adding and improving

inference rules to facilitate the production of proofs with a mathematically more natural style.

1.2.3 Proof Theory Course

The work on the proof theory course was begun in the autumn of 1975. During the last year we improved and expanded the curriculum, wrote corresponding audio lessons in VOCAL, and supplemented the logical machinery in the proof checker (see Section 3.2.3.6).

1.3 Teaching Initial Reading with Audio

One of the most critical components of teaching initial reading by audio is the generation of individual letter sounds. Recognition of such sounds is difficult because of the absence of context. However, the recognition of individual letter sounds to be matched to the appropriate grapheme by the young student, is an esential component of beginning reading. We therefore designed a test in which three systems of computer-generated speech were compared to each other and a human-voice control, on the task of producing individual letter sounds. The subjects were all first graders, little older than the anticipated target population for courses in initial reading. Besides a straightfoward statistical comparison of the results obtained, a learning study was conducted, since there is undoubtedly a learning component to the task of recognizing spoken sounds produced by any unfamiliar source.

Many important sounds to be produced by an audio system for CAI in initial reading are not contained in the sounds for individual letters. Clearly, a comparison of the three systems on a more complete list of

sounds would be an important and useful additional comparison. A second experiment was therefore performed to compare the systems on the production of words in order to test a fairly complete list of consonant and consonant cluster sounds, both as initial and final portions of monosyllabic words. The experimental subjects chosen in this case were fifth graders, as the list of words with the desired sounds as components required a wider reading vocabulary than that possesed by most first graders.

2 Computer-generated Speech

2.1 The Audio Procedures

Modularization of user procedures for accessing speech was completed during the grant period. The present structure is bilevel, where the top level is a library of well documented Sail procedures for performing all the typical actions a speech user is likely to need. In all, there are two hundred and thirty procedures available in this library. These library procedures when used become subroutines in user programs. The lower level is the "lexicon fork" which contains the lexicon and those procedures which heavily access the lexicon separating them from user programs and from the upper library routines.

2.1.1 Audio Languages

The audio system allows creation of independent languages for stored sounds. The main language for word concatenation and intonation synthesis is "English", the language containing phrases recorded for the Logic course is "Logic". Other languages are available and new ones can be created as needed.

Each language contains a data base ("lexicon") where sounds or words may be looked up by name and which contain information such as the word's part of speech, its initial fundamental frequency, its maximum fundamental frequency and a pointer to the storage location for the sound itself. Since these lexicons are very large, they cannot fit in the same virtual memory address space with any reasonably complex user program, e.g., a sophisticated curriculum driver. Our operating system TENEX has facilities for jobs to contain more than one virtual address

space ("fork") and for efficient communications between them. We have used a separate fork for the lexicon and those procedures which heavily access the lexicon. Doing this has the additional beneficial side effect of allowing both the lexicon and its accessing procedures to be shared between diverse curriculum drivers resulting in a most efficient implementation. Thus, top level procedures from the audio library, such as SPEAK("Can you hear me?), simply pass string arguments to the lexicon fork where all the computation required to have these words spoken is done. We have not implemented the intonation synthesis procedures in the lexicon fork since they have been under intensive development, as described below in Section 2.2.

An additional two thousand English words were recorded, analyzed and added to our main lexicon ("English") which now contains a total of fourteen thousand words representing over three hours of spoken individual words. Some ten thousand sentences for the Logic course were also recorded. These sentences represent seven and a half hours of continuous speech. This data base of three hours of words and seven and a half hours of sentences is, to our knowledge, the largest digital speech data base ever assembled from a single speaker.

2.1.2 Compression Techniques

We investigated the application of new compression techniques which could reduce the amount of data required for representing Linear Prediction of a speech signal. The Linear Prediction techniques produce twelve reflection coefficients which represent the short term spectrum of the voice over the time of application. The total representation of a sound in our system includes these twelve coefficients, as well as

coefficients for gain, pitch and duration. The MISS machine, which will synthesize speech from this representation, is a finite word length machine and thus each number in the representation must be truncated to a small number of bits. Also the gain coefficient must be divided into three separate but interacting coefficients to minimize the noise introduced in the calculations of the MISS machine due to the finite word length.

The literature contains a number of techniques for further compressing this type of representation, (Markel and Gray, 1974), (Makhoul and Viswanathan, 1974). One typical technique is to allocate a different number of bits to each individual parameter in the representation so as to minimize the total number of bits needed for a particular spectral distortion figure. Another typical technique is to only update a parameter when it has changed sufficiently to cause a certain amount of spectral distortion.

We have simulated many of these promising techniques to study what effects they would have on the prosodic manipulations we normally perform on our words and we were able to determine, in an informal way, that techniques which compress the linear prediction coefficients or which increase the length of time one set of coefficients is used do not significantly interfere with previously performed intonation manipulations. However, compressing the fundamental frequency and gain parameters must be done in a more conservative way when prosodic manipulations will be later performed since inaccuracies in the parameters may be compounded by some of the intonation techniques. We intend to implement these techniques in the MISS system in the near future.

2.2 Intonation Generation

2.2.1 Summary of our Prosodic Analysis Method

Our previous work has led us to adopt a "tone group" analysis. On this analysis a simple pattern of abstract (phonological) tones becomes elaborated through the syntactic and semantic structure until the specificity corresponding to phonetic observation is achieved. For a more complete review of the tone group analysis see (Levine, 1976) (Levine, 1977). We associate peak fundamental frequencies in herz (represented by the numbers corresponding to individual words) with the "tone" for each word in a sentence and then elaborate the intra-word contouring from there. In the earlier years of this grant we developed this analysis specifically for declarative sentences.

2.2.2 Some Simple Questions

It is generally accepted that wh-questions (or "information" questions) have pitch contours similar to declaratives while yes/no questions have a different contour. Below, we give short sample analyses of questions in terms of our basic tone group method. We will analyze wh-questions with the tone group (M)HL[2] as for declaratives, and, the yes/no question with (M)LH.

2.2.2.1 Some Wh- Questions

There are some small differences among the pitch contours of the three recordings of sentence 1, but the general impression is that a mid tone on 'what' leads up to the elaboration of the rest of the tone group on the rest of the sentence. From this pervasive mid tone on 'what' we

[2] Parenthesized M is an optional "mid" tone, H is a "high" tone and L is a "low" tone.

conclude that (at least in this type of question) the 'wh-' word is a minor lexical item and therefore able to receive the mid tone of the tone group.

1.

	[what [is [the answer]]]
(a)	[196 [200 [169 196]]]
(b)	[179 [192 [182 200]]]
(c)	[192 [204 [182 172]]]

In recordings 1a and 1c, there is a downstep from 'is' (the verb) to the noun phrase 'the answer'. The high to low tone contour agrees with the pattern we saw in the short declaratives above. We must say in these cases that the 'is' is sufficiently important in these clauses to avoid receiving a mid tone. In 1b 'is' does receive a mid tone.

'The' upsteps to the head noun in 1a and 1b but not in 1c which may be related to the relative importances of the two words in the noun phrase. It is also possible that the word 'answer' is slightly stressed in 1b. If we lowered the pitch on 'answer' in 1b, it would show the same pitch pattern as 1a.

2.

[[what [rule [of inference]]] [was used]]

[[167 [222 [200 182]]] [114 108]]

The next example (sentence 2) again shows the upstep (mid to high) from 'what' to the rest of the phrase even though it is not adjoined to the sentence but only to the noun phrase. Notice the large step between the noun phrase 'what rule of inference' and the verb phrase 'was used'; It is roughly an octave (222 herz to 114 herz). This indicates an isolation of the two phrases. Eurthermore, the word 'of', which would normally upstep, does not in this case, indicating that the weights of

different major and minor lexical items differ in isolated (stressed) phrases. Further evidence that this is a stressed phrase is the fact that 222 herz is above the normal pitch range observed for this speaker.

We have provided one level of syntactic simplification for example 3 because of its length and complexity. This simplification portrays the declarative fall pattern more clearly. Again, in 3b and 3c (and arguably in 3a), 'what' has a mid tone relative to the rest of the phrase. The interval of upstep is smaller here than in sentence 2, since the entire noun phrase is less prominent in the sentence; also it is not particularly isolated in relation to the rest of the pitch contour.

3.
[[what [rule [of inference]]] [[was used] [to infer [line fifteen]]]]

(a):
[[196 [196 [147 169]]] [[118 172] [164 152 [145 152]]]]
[[196 [196 169]] [172 [164 152 152]]]

(b):
[[189 [196 [161 164]]] [[120 169] [156 149 [139 147]]]]
[[189 [196 164]] [169 [156 149 147]]]

(c):
[[200 [217 [182 161]]] [[120 185] [164 159 [152 152]]]]
[[200 [217 182]] [[185 [164 159 152]]]

The exact structure of the predicate ('was used to infer line fifteen') is not critical to this analysis, since an alternative structure:

[was used to infer [line fifteen]]

is also adequate for a description of the pitch contour in our tone group terms. We would like to point out that if the word 'to' had received a mid tone, as we might expect with the first syntactic structure, the second structure would not be adequate. As the facts

lie, we need to explain away the non-mid tone on 'to'. In sentence 3, as opposed to sentence 2, we do see examples of an upstep from 'of' to 'inference' (3a, 3c). We can relate this upstep to the relative neutral contour which the entire phrase takes on.

2.2.2.2 A Yes/no Question

Suppose that we were forced to choose a tone group for yes/no questions based on the sentence presented below. Let us suppose further that the choice was between (M)HL, declarative fall, and (M)LH[3].

4.

[doesn't [[that observation] [seem [exceedingly appropriate]]]

[182 [[189 189] [156 [161 192]]]
[182 [189 [156 192]]

The choice seems fairly easy to make: the (M)HL tone group doesn't fit even approximately to this sentence. We could say that 'doesn't', 'seem', and 'exceedingly' all upstep (mid to high) to the heads of their respective phrases. While 'doesn't' is believable as a minor lexical item, and 'seem' could be argued to be some sort of copula, it is hard to see what argument can be made for 'exceedingly'. If anything, we would expect it to have an exaggerated high tone.

On the other hand, the (M)LH tone group is easily applied to this sentence. The predicate, [seem exceedingly appropriate], shows a consistent rising contour which corresponds to the low to high tone elaboration. There is no particular evidence that 'doesn't' receives a mid tone in this sentence since it may be part of the elaboration of the

[3] In reality, there are a great many more potential choices available, especially if some theoretical devices that have not been utilized in this research, e.g., boundary tones (Liberman, Goldsmith), are incorporated.

low to high contour for the structure AUX1 + NP + PRED and not (as shown here) part of a structure of AUX1 + CLAUSE.

We are not yet ready to make definitive statements about the structure of question intonation, but on the basis of this example we have begun to incorporate questions into our synthesis system.

2.2.3 Duration Studies

In what follows, we have not attempted to account for the intricacies of English rhythm, but we do think that we are fairly accurate in the majority of cases.

2.2.3.1 Experimental Errors

The quantitative-duration information presented below is subject to errors stemming from the difficulty of segmenting an utterance accurately. It is often very difficult to tell (aurally and by using quantized pitch and loudness contours) where one word stops and another begins, especially when there are phonetic processes obscuring the boundary as in the assimilation of nasal sounds (e.g., "one of which"). Another source of segmenting error comes in the difficulty of separating pause time (the length of silences between words) from stop closures (both voiced and unvoiced) word initially, and unvoiced final frications. Thus a word could be given a duration twice as long in one utterance as in another simply because the judgements of where the words actually began and ended were different in the different utterances (or even in the same utterance). An example of the uncertainty associated with difficulties of measurement is the word "the", which shows a variation in length of up to three times the shortest measured length. A further difficulty comes from the fact that a given word may be

pronounced differently in different utterances, even when the utterances are repetitions of the same words (with the same meaning).

2.2.3.2 Some Elementary Facts

WORD	LENGTH (ms)		
close	642	/z/-/k/	
cloak	549	107	/kl/-/n/
			73,65
nose	569	/z/-/t/	
note	484	65	
phonograph	808	*PHONO- (length)	
graph	608	200	
phonological	955	*PHONO- (length)	
logical	680	275	
phonograph	808	/t/-/n/	
photograph	868	60	
whoever	510	observed (e= 45)	
who	293	*-ever <= 135	
ever	319	*who- <= 330	

Figure 1. Segmental durations for selected words.

Some of the linguistic factors which are involved in duration include word-level phonetic effects, syntactic and semantic phrasal effects and discourse effects. We can view these effects as hierarchically arranged, that is, the phonetic effects establish an isolation duration for a given word, the syntactic effects act on this isolation duration to yield a phrasal duration, and the discourse effects act on this phrasal duration to yield the final, actual duration.

As an example of the word-level phonetic effects, we can examine some word groups, composed of minimally different phoneme sequences. The four words, `close´(/kloz/), `nose´, `note´, and `cloak´ are a good

case. The difference between 'close' and 'cloak' is the substitution of a voiced final fricative for an unvoiced final stop. 'Nose' and 'note' also differ from each other in these distinctive features. The correspondence is not exact, since /k/ and /t/ have different points of articulation, but the similarity is substantial. We can also pair 'close' with 'nose' and 'cloak' with 'note'. Here the distinctive difference is between oral and nasal stops, word initially[4]. The duration[5] of 'close' is 624 ms, that of 'cloak' is 549 ms, 'nose' is 569 ms, and 'note' is 484 ms. The "fricative-stop" difference is 75 ms for the /k/-initial words, and, 85 ms for the /n/-initial words. Focusing on the initial phoneme difference we see an "oral-nasal" stop difference of 73 ms for the final-fricative words, and a difference of 65 ms for the stop-final words. While further examinations of such pairs would be required to reach a firm conclusion[6], these differences agree with the general facts that fricatives are longer than stops (in general) and that oral stops are longer than nasals.

We can also see the effect of morpheme concatenation on word duration by a similar examination. Besides the cloak/nose words, Figure 1 also shows the decomposition of 'phonograph' as /phono/ + /graph/, and 'phonological' as /phono/ + /logical/. There we can see that the

[4]There is also the difference between a stop and liquid cluster, /kl/, and a single stop consonant: in addition, the point of articulation differs for /n/ is a dental nasal, while /k/ is a velar oral, but the principal distinction is oral vs. nasal.

[5]Durations in this section refer to recorded isolation durations, drawn from our lexicon.

[6]Other researchers have studied and continue to study this type of contrast. These contrasts have not been a focal point of this research and we mention them only in passing.

morpheme /phono/[7] contributes differently to the durations in each case although the difference between the two `phono's may be due to error in the recording. Notice that `phonograph' and `photograph', which differ only in the oral-nasal stop phoneme, differ in duration by 60 ms. While still not conclusive, this agrees well with the differences in cloak/note and close/nose, seen above.

As a final example of this type of comparison we look at the combination of /who/ + /ever/ to yield `whoever'. The rightmost column of that part of the figure gives durations abstracted from a pitch and volume analysis of `whoever'. From these numbers[8], it is clear that some duration reduction is going on in combining the morphemes. A possible hypothesis would be that durational shortening similar to the syntactically induced shortening is involved in morpheme concatenation. We have not pursued this enticing possibility.

Several experiments (Lehiste, et. al., 1976., among others) have shown that duration information can be used to disambiguate different possible syntactic structures for an utterance. The key fact is that the final syllable in a phrase is usually lengthened from its phrase medial length. In the hierarchical view of durational effects, the phrase boundaries are seen as modifying the phonetically predicted duration for the syllable adjacent to the boundary. Thus Klatt (1976) gives a formula for vowel length that involves a minimal length for each vowel (in the language's inventory) and a proportionality constant that varies with the syntactic environment, number of syllables in the word

---

[7] We label the constructed morpheme duration with an asterisk.

[8] The error of 45 ms is the uncertainty as to the end of /who/ and the beginning of /ever/ which is continuously voiced, but has a 45 ms region between volume peaks corresponding to /u/ and /E/.

and the stress/unstress quality of the vowel. He gives a similar formula for consonant length. Semantic importance, novelty or focus can result in lengthening from the "neutral" duration (or smaller shortening from a lexical duration).

Gaitenby (1965) found that speech style results in a difference in "tempo" but not in the relative durations of segments. "In general, slow speakers tend to be slow all along the line in their acoustic segments ...." (Gaitenby 1965, p. 3.6).

2.2.3.3 Simple Experiments

There are several ways of testing a hypothesis about duration modifications. The most straightforward involves segmenting a large number of different utterances and statistically comparing the observed durations on a word-by-word basis. Another test would involve generating sample sentences embodying the duration contrasts desired and having subjects judge the contrasts. A third test is to generate durations for an utterance (phrase or sentence) and compare that statistically with an observed (spoken) duration.

The first two tests face error from the fact that duration contrasts do not exist in isolation. Pitch (and volume) contours interact with the duration contrasts, creating seeming length differences where none exist in the acoustic signal, and negating the perceptual effect of others. The durations from the first test suffers from the possibility of errors in segmenting the utterances. The third test is liable to "washout"; whatever contrasts may actually exist in the signal can be washed out as a result of the accumulation of these differences canceling each other in the average.

We have used the first sort of test to arrive at an estimate of the necessary duration modifications and we have discussed it elsewhere[9]. Other researchers (Huggins, 1972) have conducted serious tests on the second model. We have used informal listening tests on this model as well to ascertain the reliability of our predictions. Our testing of the third type was the most disappointing of all. We generated durations for about 250 utterances, ranging from one to fourteen words in length (maximum of 4.5 seconds long). The recordings to which we compared our generated durations were made by the same speaker who recorded our vocabulary. The recordings were made independently of this experiment, for use in the computer-instruction course in logic at Stanford University (1976). The experiment measured the correlation of the recorded utterance's duration to (a) our duration predictions and (b) to the sum of the lexical durations of the words from that utterance. The disappointment in the experiment was that a standard statistical regression for a linear relationship in both cases yielded correlations that were statistically significant ($p<.001$). The correlation for the generated durations indicated that correct predictions were made in most cases (the regression line had a slope of almost 1), while the summed lexical durations predicted a too high value of about 1.7 times the observed durations. The strong correlations of both predictions shows the "washout" effect -- negating (in both cases) any useful information that might be present in the results.

2.2.3.4 Utterance Length and Text Structure

It is fairly intuitive that key sentences (for example, topic

[9]See Levine (1976, 1977).

sentences) and phrases are said more slowly than the rest of a text. Since determining which sentences are "key" in a paragraph is a tricky task, we will discuss a different regularity that we have seen which links duration of utterance with position in the structural hierarchy. Figure 2 gives the bracketing for lessons selected from the logic curriculum. The numbers displayed inside the bracketings are the difference between the length of the sentence (or sentence group) as predicted by our current duration theory and the observed length of the recorded utterance used in the logic course, expressed as a percentage of the theoretical prediction. A positive value indicates that the theoretical prediction was larger than the observed, while a negative means that the theory predicted too short a duration for the utterance. If we were trying to model the observed lengths accurately, we would need to shorten sentences which had a positive value and lengthen sentences with a negative one. The method used for these comparisons is similar to that described above in Section 2.2.3.3, in describing the experiment where we tested the overall goodness of our utterance length predictions.

Lists: Clear instances of the regularity we will discuss are in paragraphs 7 and 9. Looking at the corresponding values of these paragraphs we see the similarities easily. The texts of the two paragraphs are also quite similar, both give a two element introduction to a list of four possibilities. We can summarize the observations:

1) The introduction to the list shows a length contrast between the two elements in which the second must be shortened while the first is either lengthened slightly or shortened less. We hypothesize that the first should be predicted close to normal speed while the second is predicted to be slower than normal.

2) In the list itself, the second element stands

1.
[-11 -16 21] [[-29 6] [7 8 0 -6]]

2.
[-6 -3 16 -5 8]

[11 [11 10] [16 [31 32 32] 11] -10 12]

[4 6 5] 11 6]

3.
[4 [3 17] 13] [-5 -17]

[17 [-27 5] [7 7]] 0

4.
[[10 5] [5 [3 4]] -7] 7

5.
[[12 13] [3 3 -4 1]]

6.
3 [12 9 -1] [4 7 19] [11 0 [10 1 3]]

7.
[-4 17] [6 12 1 9]

8.
[16 12] [12 40 [15 14] 11]

[[-2 8] [9 6]] 4

9.
[9 18] [7 12 1 4]

10.
3 [-1 6 -5 -4] 15

11.
[18 0 9] [18 8 0] [6 [0 -20] [2 -14] 9]

[15 [30 20]] [-1 [-8 1]] 22 20

Figure 2. Text-view structure: percentage difference of observed length of utterance from length prediction using only syntax and lexicon.

out as being the most prominently divergent from its prediction. It is predicted to be much longer than observed. We hypothesize that in general (sentential) list elements are said at close normal speed except for the second element which should be faster than normal.

3) The introduction seems to be on the whole slightly faster than the list, and the list speed is predicted approximately correctly.

Paragraph 5, does not particularly confirm the observations of 7 and 9, though the introduction is predicted to be longer than the observed by more than the list. This may indicate that the whole lesson was read at a slightly faster pace than predicted.

The last major constituent of paragraph 1 shows the contrast within the introduction which we are looking for, and the second element on the list is spoken faster than the rest of the list except the first element. This difference from our hypothesis about list sentence-lengths might be due to causes unrelated to the text-view structure or to aspects of that structure which we have not isolated as yet or may be counter evidence.

Paragraph 10 has only a single element introduction and the third prediction relating the introduction and the list seems to hold. The list structure also seems to follow our hypothesis by showing the second element needing to be shortened while the other elements need to be lengthened from the neutral prediction.

Footnotes: The last value for paragraph 10 is a footnote and is the sentence needing most to be shortened. There are other examples of footnotes which also share this characteristic of requiring shortening: the last two sentences of paragraph 11; paragraph 4; perhaps paragraph 8; not in paragraph 3. Notice that other final sentences do not show the same need for shortening, so we would not attribute this phenomenon simply to being paragraph final.

2.2.4 Discussion of Parsers

We will discuss here some fairly complicated parsing systems which seem promising as components of a speech synthesis program for a computer instruction system. We start with some brief remarks describing our parser.

2.2.4.1 Our Parser -- Overview

Figure 3 gives an outline of our parser which uses linguistic patterns to decide how constituents are to be constructed and combined. This algorithm uses certain basic constructs (here: noun, verb and prepositional phrase; above: conjunctions, or, articles, auxiliaries and prepositions) to achieve a preliminary structure for the sentence. It then fills in the structure so established by creating more complex constituents. The final step in this parse procedure is to assign all as-yet unanalyzed words to some phrase. The basic motivation in this step is that English is a right-branching language, i.e., most of the complex constructions in English occur on some right branch of the syntactic tree. An example of this complex right-branching structure is a noun with a relative clause suspended from it as in "John, who came home late last night." The structure for this phrase is

[John [who came home last night]],

one word adjoined to a clause on the right. While not every structure in English is right branching, this is a useful guess for the parser to make when it finds no other analysis.

Still more complicated parsing procedures are conceivable. What is particularly missing from surface parsers is the capability to deal with

1. Find the simplex noun phrases, verb phrases and prepositional phrases in the sentence.
2. Use these phrases, along with unphrased words, to form more complex phrases by looking for specified elements and then associating other constituents into the phrases.
   a. Some specified elements are searched for from the front of the sentence; some are searched for from the end.
   b. Associated constituents may be before or after the specified element.
3. Complete the structure by including any unphrased words as either (a) or (b).
   a. their own phrase, if there are enough words together.
   b. a left sister to some constituent, the created sisters to be dominated by some single node.

Figure 3. Overview of our surface parser.

missing and moved constituents[10]. Let us consider a simple-phrase structure parser, which is non-recursive; there are no embedded constituents in its parsed structures. Such a parser is not subject to this difficulty to the same degree as our full surface parser since the limited structures available as simple phrases are unlikely to contain either moved or missing words that make important structural differences. We can compare the results of the surface and transformational parsers using the example from our previous discussion of stress reduction, which we repeat below, together with different possible parses. Parse (a) would come from the simple-phrase parser, (b) would come from a full surface parser, and, (c) could result from a transformational derivation.

[10] Missing constituents and discontinuous constituents were a major motivation (historically) for incorporating transformations and/or semantics into accounts of syntactic structure.

	Eager though	I	am	to win,	I would never cheat.
(a)	[Eager though]	[I]	[am]	[to win]	[I] [would never cheat].
(b)	[Eager [though	[I	[am	[to win]]]]]	[I [would never cheat]].
(c)	[Eager [though	[I	[am ___	[to win]]]]]	[I [would never cheat]].

The focus of attention in this figure is on the different parsings for "am to win." The simple-phrase parse for those words shows two separate phrases. In terms of the prediction of contractions[11], this parse could agree with either of the other two, either incorrectly allowing contraction as in (b) because of a tonal assignment upstepping the word 'am' to the phrase "to win", or not, as in (c), instead, correctly assigning separate prominences to the word and the phrase. Parses (b) and (c) make conflicting predictions, with the empirical consequences supporting parse (c).

From a purely practical point of view, we note that the bulk of English sentence structures do not exhibit these deletions and movements. The additional complexity due to incorporating derivational schemes or semantics into a parser must be balanced against the need for correctness in these limited cases. While the surface parser does not always yield the best parse, it comes very close in a large number of cases.

2.2.4.2 More Complicated Parsers

Two parsers which perform more complicated syntactic analysis than our surface parser are those by Kaplan (1976) and Marcus (1974, 1976). Both of these attempt to produce structures which would make the correct

[11] Cf. Selkirk (1972) and Levine (1977).

predictions in the contraction cases cited above[12], that is, express generalizations about moved and deleted constituents. Neither parser tries to reproduce the derivational history of a sentence as part of a parse.

Kaplan's ATN parser

Kaplan's parser is an augmented transition network. A simple transition network is a graph of connections (arcs) between possible states. In a parser each state represents some constituent which is recognized when the parser arrives at that state. The syntactic patterns which were the basis of our parser are represented in sequences of arcs and states of the ATN. Some sequences of states which would otherwise appear many times in the parser can be factored into subroutines which are separate transition networks. This transition network can be referenced by an arc between two states, as if it were a simple word recognition. In this case, the sub-network must give a successful recognition in order for the parse to proceed. These sub-networks normally represent phrase constructs like noun phrase, verb phrase, etc.

The augmentation of the network comes from allowing the arcs to represent (unlimited) computations. These computations can be used to bracket or re-bracket constituents, to assign labels to various elements, and to perform tests besides the simple constituent recognition which the states represent. An important augmentation in Kaplan's system is the 'hold cell'. Some of the arcs have the possibility of placing a recognized constituent (either simple or complex and usually a noun phrase) into a designated cell, while other

[12]Needless to say, this expression of their aims is totally our own.

arcs are enabled to use this cell instead of performing a sequential sentence constituent recognition. Use of the hold cell allows this parser to analyze sentences with unbounded movement or deletion, which our purely surface parser could not, while avoiding reference to derivational histories. Unbounded leftward movement, which is the main example in English, is recognized by placing the (supposed) moved constituent into the hold cell[13] and then whenever a pattern may require a constituent of that type the hold cell is emptied of its contents.

An important difference between the organization of the ATN and that of our parser is that the ATN completes its parse of one (the current) constituent and then processes the next word or constituent in sequence, It is always building up a unified structural description. Our parser will create many independent constituents and then try to combine them into more complex constituents. The difference between these two approaches shows up with "garden path" sentences, such as "The horse raced past the barn fell." The ATN will initially follow a "garden path" in mis-analyzing a sentence until it can proceed no further. At that point it has to back up and recreate at least some of the parse, making some different choices from the structures assigned the first time. In the sample sentence it will parse "the horse raced past the barn" as a full sentence and then have no analysis for "fell". Then it will have to go back and analyze "raced past the barn" as a reduced relative clause in order to fit "fell" into the structure of a declarative. Our parser will always maintain the integrity of the simplex structures created in the earliest part of the parse, even if

[13] The supposition can be based on the absence of a syntactic pattern utilizing the constituent at that point.

higher level structures are difficult to form from these constituents. In the above sentence, assuming 'past' is marked (or recognized) as a preposition, the simple constituents "the horse", "past the barn", "raced" and "fell" will be parsed in the first run-through. The parser is then free to create any grouping of complex constituents it can. In fact our parser can only create a single structure for any sentence and thus cannot back up in case of parsing difficulties.

Another difference between the parsers is that an ungrammatical 'sentence' will require significantly more parsing than a grammatical one does from an ATN but not much more from our parser. This is because the ATN will attempt to recover from possible "garden paths" until all possibilities for parsing the sentence have been exhausted. Our parser will simply leave whatever constituents it can find sitting around and then give up. The ability to give up quickly is valuable in a setting where ungrammatical utterances may be encountered. The concomittant drawback is that our parser will tend to compound its errors and then leave them sitting whereas the ATN will try harder to get the right answer.

Marcus' 'Wait-and-See' parser

The 'wait and see' parser (WASP) has similarities both to our surface parser[14] and to Kaplan's ATN. Like our parser, WASP depends on creating small, simple structures and eventually combining them to form the more complex ones, and, like the ATN, it proceeds from the beginning of the sentence to the end without going backwards and forwards in looking for complex constituents. WASP works roughly as follows:

1) The parser proceeds from the first word of the sentence and looks for matching patterns (sometimes predicting what should be the next element in the sentence).

[14] Our parser has borrowed somewhat from Marcus'.

2) If the current word does not seem to form part of some constituent (does not match a pattern) it is stacked on a list of constituents.

3) If it finds a matching syntactic pattern, the words/constituents comprising that pattern are phrased together and treated as a single constituent. After a constituent is parsed, it is put on the list of constituents along with unparsed words.

4) Forming a new constituent can, by matching another syntactic pattern, cause already parsed constituents to be popped off the list (last-on first-off) in order to be incorporated into some new pattern.

5) A constituent can have various kinds of information associated with it and its sub-constituents. In a noun phrase, the entire phrase might be labeled [15], and the head noun in that same phrase might be so labeled. In a sentence, the subject and object(s) might be labeled for their function as well as for their syntactic structure.

In a sense the WASP employs a different organization of practically the same basic notions of patterns and parsing style as are used in our surface parser. We can imagine recasting our several passes across constructed constituents into interlocking patterns, where the formation of one constituent allows another pattern to activate itself and try to find all the necessary components. Also there is an appealing sort of psychological plausability to this parser (as well as Ron Kaplan's ATN) in the left-right direction of attention focus movement, and the single pass through the sentence. There is no good evidence to support the opposite view that attention scans back and forth through the words and constituents of the sentence in the way that our parser would have it done.

[15] It could receive additional labels as part of further phrasing.

2.2.4.3 Semantically Guided Parsers

The parsers we have discussed here are all syntactically based. The patterns upon which recognition depend are independent of considerations of the meaning of the words or sentences involved, or of the structure of the discourse up to that point[16]. Interesting work on parsing using world knowledge or reference to meaning has been done, especially by Winograd and Schank[17] who, despite the differences between their approaches, share the underlying assumption that non-structural information is important in structural analysis, that syntax is not self-contained but requires semantics (and possibly additional sources of information) in order to be analyzable.

We have not closely looked into the use of semantically oriented parsing for two reasons. One reason is simply the practical one that this more complicated parsing requires more resources and has not been implemented in a general enough way that it could be a candidate for incorporation into this project. The other reason for not pursuing semantic parsing is our belief that the phonology (intonation) can utilize only syntactic structure along with some very specific stress/destress information, but without access to semantic information to guide the elaboration of tone groups. If the parse itself depended on semantic information any claim in this area would be a vacuous one.

---

[16] They are also independent of the weather at the time of the parse, but we take it as evident that the reasonableness of using meaning in parsing does not require the justification that reference to the weather does.

[17] Samples of different approaches along these lines are contained in Schank and Colby (1973).

2.3 Outside Contacts

During the past year, we have had numerous contacts with Professor Jon Allen of MIT. He has provided us with updated versions of his text-to-phoneme programs. His programs are written in the language "BCPL". The TENEX system in use at our site also has a BCPL compiler but the two compilers have vast differences both syntactic and semantic. We were able to convert most of the text-to-phoneme programs to TENEX BCPL and now have them operational. We currently use his text-to-phoneme program as one part of our word analysis routine for dictionary storage of a sound. When a word is recorded, there is always some silence before and after it in the recording. For good concatenation this silence must be removed. By using the MIT program to provide a phonetic transcription of the word, our analysis program is able to more accurately determine the boundary between silence and word in our recordings.

We have also exchanged information, through personal contacts and regular correspondence, about how our different intonation algorithms work, including plots of parameter waveforms both before and after application of our algorithms and representative samples of the sentences our curriculums use. We have also provided him with some programs of a general nature and our lent our expertise on the suitability of certain peripherals for the recently acquired computer system, which is fairly similar to and largely compatible with our system.

3 Complex Teaching Programs with Audio

The Institute has developed three large scale, college level, mathematically oriented CAI courses: Elementary Logic, Axiomatic Set Theory, and Proof Theory. All three courses are used by Stanford University as regular parts of the undergraduate curriculum. Students receive three to five units of college credit for these courses, and use them as prerequisites for other, non-CAI courses at Stanford. The courses also serve as an environment for the study of learning and teaching methods. This section describes the work done in the past year to further develop and extend these courses, and the results of the experiments performed during the past year on various aspects of the courses.

3.1 Introductory Logic Course

3.1.1 Audio/Display Interaction in the Logic Course

The first portion of the logic course to be rewritten in VOCAL was that which dealt with translation from predicate logic to English and vice versa, since it was initially thought that the addition of audio to the logic course would have the most impact on such paraphrasing exercises. While the prosodic emphasis and the informal explanations available in audio mode did aid in the understanding of these important semantic concepts, it turned out that audio made a greater impact on a different area of the course.

When the remainder of the logic course was rewritten, we found that the primary advantage of an audio/display presentation over simple display was in the demonstration of processes, such as an example of the

use of a new inference rule. With an audio-supplemented presentation, only the material which would actually be typed during a derivation need appear on the display, thus preventing a confusion of explanation with object. For example, rather than using artificial devices such as bracketing to indicate text which is presumed to be typed by a student, the author may simply have the program speak "you type the line number" as a number is being typed, and speak "the computer will then print the resulting formula" as the formula is being printed. Thus, the display invariably looks clearer and less cluttered when comments are reserved for the audio.

The spoken text also allows one to add a dimension of timing to the process being demonstrated, so that its steps can be done one at a time as they are explained orally. The author can coordinate a display action with the time when a student hears a spoken comment (even though the student controls the speech rate), but there is no way to know when a student finishes reading a written comment, except by use of the HOLD opcode, and overly frequent HOLD's become annoying. Thus, in the display-only versions, we had to type out a large section of the example all at once, parallel with a large section of explanation, and leave the student to jump back and forth from one area of the screen to another as he reads. The loss of clarity is very dramatic, but it is impossible to demonstrate this adequately in a report which itself must be committed entirely to the written page.

We will attempt to convey the effect somewhat by illustrating the successive display contents, with spoken comments below. In the following example, adapted from an actual lesson, double divider lines (====) mark places where the student can have the spoken text repeated

or "hold" the presentation until he is done examining the display, and single divider lines (----) indicate places where something changes on the display. Boldface type in the display content represents brightening (display in double intensity), and underlined type in the spoken text marks words which have been tagged for extra prosodic emphasis. The actual VOCAL code which produces this lesson follows the illustration. The VOCAL author manual, which was written this summer, contains further discussion of audio/display interaction and several more samples of lesson code.

SIMULATION OF THE AUDIO VERSION:

```
P          (1)  2 + Y = 8 + X  &  X = 3 + 1 - X
P          (2)  X = 2
*
```

SPOKEN: "Our two new rules work a lot like Rules R Q and R Q R, except that instead of replacing equivalent sentences, they replace equal terms. For all four of these rules,"

```
P          (1)  2 + Y = 8 + X  &  X = 3 + 1 - X
P          (2)  X = 2
*1
```

SPOKEN: "the first number indicates the line in which the replacement is to take place,"

```
P          (1)  2 + Y = 8 + X  &  X = 3 + 1 - X
P          (2)  X = 2
*1,2
```

SPOKEN: "and the second number indicates the line which justifies the replacement. For RE and RER, this line must be an equation."

```
P          (1)  2 + Y = 8 + X  &  X = 3 + 1 - X
P          (2)  X = 2
*1,2
```

SPOKEN: "To replace occurrences of the left term with the one on the right,"

```
P          (1)  2 + Y = 8 + X  &  X = 3 + 1 - X
P          (2)  X = 2
*1,2RE
```

SPOKEN: "you use Rule RE. If you don't put any occurrence numbers after the rule,"

```
P          (1)  2 + Y = 8 + X  &  X = 3 + 1 - X
P          (2)  X = 2
*1,2RE     (3)  2 + Y = 8 + 2  &  2 = 3 + 1 - 2
```

SPOKEN: "all of the occurrences will be replaced."

==========

```
P          (1)  2 + Y = 8 + X  &  X = 3 + 1 - X
P          (2)  X = 2
*
```

SPOKEN: "If you don't want to replace all the occurrences, then list the numbers of the ones you do want to replace after the name of the rule. For example, if we only wanted to replace"

```
P          (1)  2 + Y = 8 + X  &  X = 3 + 1 - X
P          (2)  X = 2
*
```

SPOKEN: "the third occurrence of X here,"

```
P          (1)  2 + Y = 8 + X  &  X = 3 + 1 - X
P          (2)  X = 2
*1,2RE
```

SPOKEN: "then we would type 1, comma, 2, R E,"

```
P          (1)  2 + Y = 8 + X  &  X = 3 + 1 - X
P          (2)  X = 2
*1,2RE3
```

SPOKEN: "and finally a three,"

```
P           (1)  2 + Y = 8 + X  &  X = 3 + 1 - X
P           (2)  X = 2
*1,2RE3     (3)  2 + Y = 8 + X  &  X = 3 + 1 - 2
*
```

SPOKEN: "and hit escape, of course."

```
P           (1)  2 + Y = 8 + X  &  X = 3 + 1 - X
P           (2)  X = 2
*1,2RE3     (3)  2 + Y = 8 + X  &  X = 3 + 1 - 2
*
```

SPOKEN: "If we wanted to replace the occurrence of 2 in line 1 with an X,"

```
P           (1)  2 + Y = 8 + X  &  X = 3 + 1 - X
P           (2)  X = 2
*1,2RE3     (3)  2 + Y = 8 + X  &  X = 3 + 1 - 2
*1,2RER
```

SPOKEN: "we would use the Replace Equals (Right) Rule, which is abbreviated R E R, instead of Rule R E. Since there is only one occurrence of 2 in this line,"

```
P           (1)  2 + Y = 8 + X  &  X = 3 + 1 - X
P           (2)  X = 2
*1,2RE3     (3)  2 + Y = 8 + X  &  X = 3 + 1 - 2
*1,2RER     (4)  X + Y = 8 + X  &  X = 3 + 1 - X
```

SPOKEN: "no occurrence number is necessary."

=========END=OF=EXERCISE=========

THE VOCAL CODE FOR THE ABOVE EXERCISE:

```
[EXERCISE 3 "RE & RER:  `Replace Equals' and `Replace Equals (Right)'"
[AUDIO
(TEM2 "
P           (1)  2 + Y = 8 + X  &  X = 3 + 1 - X                    %1_
            xxx  A          B     C          D
P           (2)  X = 2                                              %2_
            yyy  z   w
*1,2RE      (3)  2 + Y = 8 + 2  &  2 = 3 + 1 - 2                    %3_
aXcYRR      ppppppppppppppp E  q  F rrrrrrrr G
*1,2RER     (4)  X + Y = 8 + X  &  X = 3 + 1 - X                    %4_
bUUUVVV     ttt  I uuuuuuuuuuuuuuuuuuuuuuuuuuuuuu

"
"
*1,2RE3     (3)  2 + Y = 8 + X  &  X = 3 + 1 - 2                    %13_
eSSSSST     sssssssssssssssssssssssssssssss H
"
```

```
(HOLD  (S (T 1 2 a)
          "Our two new rules work a lot like Rules R Q and R Q R,"
          "except that instead of replacing $3 equivalent $2 sentences,"
          "they replace $2 equal $1 terms."
          "For all four of these rules,"
          (B X x)
          "the first number indicates the line in which the"
          "replacement is to take place,"
          ((U X x) (T c) (B Y y))
          "and the second number indicates the line which"
          "justifies the replacement."
          "For R E and R E R, this line must be an equation.")
)
(HOLD  (S ((U Y y) (B z B C D))
          "To replace occurrences of the left term with"
          (B R)
          "the one on the right, you use Rule R E."
          (W 250)
          "If you don't put any occurrence numbers after the rule,"
          ((U R z) (T p) (B E) (T q) (B F) (T r) (B G))
          "$2 all of the occurrences will be replaced.")
)
(HOLD  (S ((U B C D) (OE 3 13) (T e))
          "If you don't want to replace all the occurrences,"
          "then list the numbers of the ones you $2 do want to replace"
          "after the name of the rule."
          (W 250)
          "For example, if we only wanted to replace"
          (B D)
          "the third occurrence of X here,"
          (T S)
          "then we would type 1, comma, 2, R E."
          (B T)
          "and finally a three,"
          (W 250)
          "and hit escape, of course.")
          (W 1000) (U T) (T s) (B H) (T b)
) (COMMENT "end of hold")
(HOLD  (S ((U D H) (B A))
          "If we wanted to replace the occurrence of 2"
          (B w)
          "in line 1 with an X,"
          ((T U) (B V))
          "we would use the Replace Equals (Right) Rule,"
          "which is abbreviated 'R E R', instead of Rule R E."
          "Since there is only $2 one occurrence of $3 2 in this line,"
          (U V)
          "no occurrence number is necessary.")
          (W 1000) (U w) (T t) (B I) (T u)
)  (COMMENT "end of hold")
      (S (U A I)
          "The next exercise will give you some practice"
          "with these two powerful rules of inference.")
)]] (COMMENT "end of Exercise")
```

3.1.2 Student Preference in Audio-nonaudio Choice Situation

During the winter and spring quarters of the 1976-77 academic year, 166 students were enrolled in the logic course. Data was collected on connect time in exercises, audio choice at login, and calls to Browse-mode. Students were divided into two groups, each of which was exposed to audio and nonaudio versions of an initial segment of the course. In the winter quarter half of the students were exposed to audio in the first three lessons and nonaudio in the next three; the other half had nonaudio followed by audio. Beginning with lesson 7 and continuing through lesson 18, the students were free to choose, at each login, either audio or nonaudio versions of the course. During the spring quarter, the forced switching initial segment was reduced to two lessons, and data was collected through lesson 20. In addition to this experiment, the spring groups were further divided into two groups for each original group. Half of each of the original groups were flagged for precompiled synthetic prosody as opposed to long sounds (see (Hinckley, et. al., 1977)), so that if they chose audio at login during or after lesson 21, they would hear the synthetic prosody. Questionaires were used during the spring quarter to provide some background on the students' view of the course, the audio component, and the reasons for their choices.

The data collected from these experiments is still in the process of being analyzed, mainly with view toward generating a stochastic model of the students' preference in terms of the choice paths. Forthcoming articles and technical reports will provide detailed accounts of these experiments and their analyses.

In addition to the above experiments, a study was conducted on

students' behavior in interpretation exercises. These exercises require the student to generate a counterexample in the domain of integer arithmetic for an invalid argument, and then prove that their interpretation of the argument is indeed a counterexample. More complex exercises of this type require the student to first decide whether a given argument is valid or invalid, then prove the argument or generate a counterexample. Another use of this type of exercise is in showing the consistency of a set of statements.

Data from the interpretation exercises, including use of the 'hint' feature, were collected in the winter and spring quarters of the 1976-77 academic year. The data were collected for two purposes: first, to predict the difficulty a student would have on a particular exercise from the structure of the exercise; second, to find a stochastic model that would describe the student behavior on the interpretation exercises where the student has to decide whether an argument is valid or invalid. Data analysis is now being performed and the results of the investigation are expected to be published summer 1978 in the form of a dissertation by Inge B. Larsen.

3.2 Set Theory Course

3.2.1 Audio Introduction to EXCHECK Based Courses

The goals of the OVERVIEW program mentioned in the proposal have been expanded to include a historical model of each student's style of proof building, as well as the model of the current object dialog. As a consequent, this program is still in the development stage. Rather than wait for its completion before writing an audio introduction to EXCHECK,

though, we have utilized a HELP system, which was originally written for the logic course, to perform this function.

The HELP system is completely implemented. A set of help modules, similar to the explanatory exercises of the logic course, are written in VOCAL. They can include derivations and other types of questions, but such exercises are for assistance only. They are not "scored", and the student may skip them if he desires. The system is meant to contain a help module for each topic which students may need tutorial-style assistance on. It can be expanded by the teaching assistants as they encounter student problems. The course authors also maintain a "graph" of how the various modules in the course HELP system relate to each other.

Unlike the intended OVERVIEW program, HELP does not itself keep track of how the student is performing in the course, nor does it interrupt and volunteer information to a student who is having trouble. Rather, the system is called by the course driver when a student types 'HELP', and then the student is given a series of topic choices intended to narrow down his particular area of difficulty or interest. He may then ask for one of these topics, or any other topic which he knows the name of, and the requested audio helpmodule will be presented. A further list of related choices (specified by the aforementioned graph) is then presented, and the procedure repeats until the student asks to return to the outer course.

The HELP program is passed two arguments by the main driver: the lesson number which the student is currently at, and a list of topics which may need special emphasis. Each lesson has associated with it a subset of the helpmodules which are particularly relevant to the

material presented in the lesson. The special emphasis list is usually null, but if the student has recently been given an error message, it will be set to include any relevant topics (e.g. to the modules on syntax when a student has entered a formula which will not parse, or to the modules on quantifier restrictions when a student has attempted an invalid use of a quantifier rule). The lesson emphasis and error mode emphasis modules are then added to the initial list of topic choices. Thus, for the HELP system to become more responsive to the student's particular needs, no modification of the HELP program itself is needed; rather, the main driver, with the help of OVERVIEW, need only become more sophisticated in its choice of arguments to pass to HELP.

Since EXCHECK is used for several courses (currently set theory, proof theory, and the A-grade sequence in probability theory of the logic course), some students who begin one of these courses will have already encountered EXCHECK in another course. Others may be familiar with the general operation of our computer assisted instruction system through courses, such as "Introduction to Logic", which do not use EXCHECK. The HELP system is thus an especially appropriate vehicle for the introductory sequence in the use of EXCHECK and the rest of the instruction system. Students can ask to view just as much of the material as is new or useful to them.

Therefore, all the introductory material not specific to Set Theory was put into audio help modules. The following list, taken from a recording of the topic lists output in a test of the HELP system, is representative of the more than 150 help topics currently included in the EXCHECK sequence. Most have associated tutorial output, but some topics (like ADMIN, SYSTEM, and QUANTIFIERS) are used only to guide the

interrogation which leads to selection of an appropriate module. (The bracketed letters indicate the minimum string which a student must type for the system to recognize which topic name is intended.)

ADMIN	[AD]	Administrative matters
SYSTEM	[SY]	Communicating with the computer system
AUDIO	[AU]	Problems with the audio system
REPEAT	[REPE]	Repeating the most recent material (↑A)
BROWSE	[BR]	Using Browse Mode (↑B)
HINT	[HI]	Getting hints in derivations or questions (↑H)
SPEED	[SP]	Controlling the speech rate (↑S)
BACKSPACING	[BA]	Erasing mistypes with ↑W, ↑X, & the DEL key
ZAP	[Z]	Logging out or leaving a subsystem (↑Z)
EXERCISES	[EXE]	The various types of exercises in the course
GRIPE	[GRI]	How to send a complaint or suggestion
NEWS	[NEW]	How to ask for news on the course
EXCHECK	[EXC]	The use of the proof checker
WORKING	[WO]	The use of working premises
SORTS	[SORTS]	The sorts of variables and terms
META-PROOFS	[META-P]	How to prove theorem schemata
REPLACE	[REPL]	The Replace rule
REP-SUMMARY	[REP-S]	Summary of the operation of REPLACE
REP-EG1	[REP-EG1]	A basic example of the use of REPLACE
REP-EG2	[REP-EG2]	An example using more features of REPLACE
RER	[RER]	The Replace Equals (Right) rule
VERIFY	[VERI]	The Verify rule
VER-LIMITS	[VER-L]	Limitations on the operation of VERIFY
UNAVAILABLE	[UNA]	Rules from Logic 57 which don't work here
REMOVE	[REM]	Removing proof lines from the display region
ABBREVIATE	[ABB]	Defining and using your own abbreviations
OWN-FORMULA	[OW]	Proving your own formulas with SETDERIVE
QED	[QE]	What the QED command does for you
SCOPE	[SCO]	The scope of a quantifier
BOUND	[BO]	The definition of bound variables
QUANT-RULES	[QUANT-RU]	The Rules which manipulate quantifiers
QUANT-RESTR	[QUANT-RE]	Restrictions on the quantifier rules
AMB-NAME	[AM]	Using variables as ambiguous names

In addition, a short sequence of audio exercises on the language of set theory accepted by EXCHECK was written, to be presented as the first lesson of the Set Theory course. Since the languages differ in some ways from course to course, this material will not cause repetition problems when presented to all set theory students, as would the material on inference rules and general system use. One of these first

exercises explains how to enter the HELP system, and suggests using it to view all the introductory material with which the student is not already familiar.

Formerly, the only place this introductory material existed was in a course manual. The on-line course contained frequent injunctions to read a given section of the manual before proceeding to the next proof. Now that all this material is included in a much more instructive, interactive form in the HELP system, the manual can be purged of its tutorial-style sections, and made into a smaller and cleaner reference manual to be kept at ones side during a session at the terminal, rather than read in preparation for such a session.

3.2.2 Introduction of Audio Capability in EXCHECK

When computer-synthesized speech first became available for use in CAI, a primary concern of the IMSSS research staff was its effective utilization by course designers, curriculum authors and students. The Stanford logic course provided the appropriate environment for investigating these problems since it had the most well-developed curriculum (written and extended over a period of many years by a diverse group of authors) and a consistently large student enrollment (120 students enrolled in the logic course during the 1977 Spring quarter.) Therefore the initial implementation of the programs to support audio and coordinated visual displays were specifically designed for the logic course.

During the past year the capability for computer-synthesized speech has been added to programs associated with the more mathematically-sophisticated EXCHECK system. This addition represents a major step

away from the experimental usage of audio in CAI courses, and toward the use of audio as a standard system component in CAI.

The first step in extending audio capability was to to allow lessons for all the Stanford CAI courses to be written with an audio component. The VOCAL lesson compiler/interpreter, developed for the Stanford CAI logic course, was extended to allow audio lessons to be compiled and tested for any EXCHECK-based course. Essentially this extension involved merely allowing the use of arbitrary parsers. However, to use this program efficiently under the TENEX operating system, the program had to be restructured at two levels: the source files had a new compilation structure imposed on them, and the runtime program had a new "fork" structure added. The resulting runtime program now has a main fork (process) which contains entirely shared code, and an audio fork which is also shared among all users. The only non-shared code consists in the parser fork, which may differ per course. The fork structure was not critical when computer-synthesized speech was used only by writers of logic lessons, since they all used the same (sharable) program; however, with essentially the same code in use for different courses, proper utilization of the TENEX operating system requires complete sharability of programs.

Additional problems in moving from a purely experimental design to a design admitting widespread application were attacked. Most important after proper system utilization was the problem of producing code which ran more efficiently than its experimental forebearers. Efficiency in string-handling and list processing made strong demands on the extended TENEX-SAIL compiler, especially on the SAILISP extension. Design problems in SAIL and SAILISP were found which necessitated a rewrite of

the latter. In restructuring the compilation of source files, many steps were taken to make the code more efficient (efficiency was increased in both string and s-expression handling).

In addition, the new compilation structure provides a greater degree of modularity, which makes it relatively simple to implement local program changes. The initial logic program with audio capability required about 90 minutes of runtime to compile and load. (In addition it required extensive and inefficient use of disk space.) Minor program changes often required the full 90-minute recompilation sequence. Such an expensive compilation sequence would not allow us to extend audio capability to provide students with analyses of their individual proofs. With many individuals now working on program modules for different courses it is essential to allow "local" program modifications to be incorporated without lengthy recompilations of the entire system. The restructuring of the course drivers (which handles the audio and display components) will allow such local modifications to be made with only a small expenditure of computer resources.

An analogous but even more extensive restructuring of the EXCHECK system to incorporate audio capability in its driver program is now in process and is expected to be completed in September. The development of a dynamic system for proof explication using audio has been temporarily discontinued, pending completion of the introduction of audio capabilities in EXCHECK, and improved real time generation of audio messages with synthetic prosody.

3.2.3 The EXCHECK Proof Checker

The proof checker used in the EXCHECK system is a general-purpose

proof checker for many-sorted axiomatic theories. A description of the proof checker and previous refinements to it may be found in (Smith, Graves, Blaine and Marinov, 1975), (Smith and Suppes, 1976), and (Smith and Blaine, 1976). The basic design philosophy of the checker is to accept proofs presented in the style of standard mathematical practice. That is, just as the goal of a natural language system is to understand language as it is actually used, the goal of the EXCHECK system is to understand and check proofs as they are actually presented. We are as yet a considerable distance from that goal but in the last year progress has been made in making more natural the basic commands of the proof language used by the students to express their proofs.

3.2.3.1 Decision Procedures

It is common in standard mathematical practice simply to state as obvious elementary mathematical results rather than to construct explicit derivations of those results from axioms and theorems. The same freedom can be provided in proof checkers for those parts of elementary mathematics or logic for which there is a feasible decision method. One such area is quantifier-free boolean algebra or, equivalently, quantifier-free set algebra. The EXCHECK system contains an inference rule BOOLE based on the decision procedure for quantifier-free set algebra, an inference rule TAUTOLOGY based on a truth table decision procedure, inference rules VERIFY and IMPLIES based on a resolution theorem prover. The TAUTOLOGY and BOOLE rules were described and illustrated in prior reports. In this last year a new inference rule TEQ was added for use in inferences involving only tautology and identity.

The TEQ rule will accept most inferences that can be obtained by repeated use of the sentential rule and identity rules. In particular, it handles the congruence properties of identity as can be seen from the example below. Also, see the example for the REPLACE rule for another use of TEQ.

```
*WP     (1)  *A = B
*WP     (2)  *Pow(A) = pow(C)
*1,2teq$  (3)  *Card(pow(C)) = card(pow(B))$
Will you wish to specify? (No) *$
Using *g$0
*
```

Decision procedures, such as those used in the BOOLE and TAUTOLOGY rules, if they are to be used in programs for informal mathematics, have to be feasible and should decide a perspicuous class of statements. The procedures used in BOOLE, TAUTOLOGY, and TEQ satisfy these requirements but the resolution procedure used in VERIFY and IMPLIES does not satisfy the requirement of perspicuity. In the last year the VERIFY and IMPLIES rules have been augmented with natural deduction heuristics to make them correspond more closely to what users find obvious.

3.2.3.2 Sorts

Elementary mathematical and logical facts are another related kind of detail that must often be handled explicitly in proof checking programs while they are almost never handled explicitly in informal proofs. Sorts of complex terms are a good example of this kind of detail. If a program is to accept natural proofs it will have to implicitly handle such details as the sorts of complex terms. Sorts might be unusual, however, in that they can rather neatly be implicitly handled—taken to be a part of the implicit context as in standard

practice. Most of the procedures for the implicit handling of sorts were rewritten this past year, with a considerable gain in efficiency.

In the set theory course there are currently five basic sorts: general, set, function, ordinal, and cardinal. Associated with each sort is a group of variables that range over that sort. In the current set theory course, `A' and `B' range over sets, while `a' and `b' range over ordinals. Hence, the statement that for every set there is an ordinal equipollent to it could be expressed: for every A there is a b such that b is equipollent to A. The sorts are closed under union, intersection, and relative difference and form a set algebra. Hence, the relation of inclusion between sorts is decidable.

Complex terms also have sorts in that they denote objects that are sets, or ordinals, or functions, or the like. In our version of set theory, ordinals are sets. In fact, an ordinal is the set of all smaller ordinals. It follows that the intersection of two ordinals is an ordinal—the smaller ordinal. However, not all sets are ordinals and the intersection of two sets might be a set that is not an ordinal. Hence, the object denoted by a compound term formed using `∩' might, for example, be ordinal or it might be a set that is not an ordinal. The EXCHECK program must determine a sort for complex terms before it can substitute them for sorted variables. Rather than have the user explicitly establish the sort of a complex term, the program tries to compute the sort on the basis of information it has available to it. The curriculum authors supply EXCHECK with basic information about the sorts of variables and, for each function symbol, information about how the sort of a compound term formed using that operator is related to the sorts of its subterms. For example, part of the information for `∩' is

that if both subterms are ordinal, then the compound term is an ordinal; otherwise, if both are sets then the compound is a set.

Using the information available to it, the EXCHECK program will compute a sort for any compound term. However, occasionally the sort computed is insufficient to permit the desired inference. In such cases the user is required to supply information justifying assigning a more restrictive sort to the compound term. Once this is done, the information about the new sort is saved so that the student need not repeat the process each time the term is used. Information about the new sort is stored on one of two lists depending upon whether or not extratheoretical assumptions are required to establish the new sort. If no extratheoretical assumptions are required, the result about the new sort is a theorem and it is made available as a standard part of the implicit sort machinery. If extratheoretical assumptions are required, the result is only made available in the context of those assumptions.

3.2.3.3 Schemata

The instantiation of axiom and theorem schemata is an area where some effort must be made to provide routines that do not involve the user in logical details. The procedures involved in the instantiation of schemata were completely redesigned and rewritten in the last year. They were extended to automatically handle almost all of the logical detail involved when used explicitly by the student and to automatically do all of the work when used in conjunction with IMPLIES.

In standard practice one simply says or writes down the appropriate instance. Using proof checking programs it involves less work to specify the instance and let the program generate it. Further, the same basic

routines are used by the program to compute what instance to use in an application of the IMPLIES rule involving a schema.

The program generates an instance of a schema by:

i) Replacing the variable being used to mark the parameter places by the parameter that occurs in the schema. In (2) above, this means that 'z' is replaced by 'x' everywhere in the formula 'z in A U B or z = y'. In a case where we have 'FM(x,y)', 'x' would replace the variable which marks the first parameter place and 'y' would replace the variable which marks the second parameter place.

ii) Substituting the formula which results from i) for 'FM(x)' in the schema.

In both steps variables will be rebound to avoid capture or clashes of bound variables as needed.

So far the handling of schemata is quite straight forward; however the case where the sort of the parameter in the schema differs from the sort of the parameter in the desired instance requires more care.

The approach we have taken is to modify the algorithm to note the sort of the variable being used to indicate the parameter places. If this differs from the sort of the corresponding parameter in the schema then it is regarded as an instruction to generate the instance where the parameter is of the new sort. To do this the program substitutes a formula that is made up by first replacing the variable by the parameter in the schema and then forming the conjunction with the assertion that the parameter is of the new sort. The program then rewrites the result in the new sort where possible. (In fact the code is more sophisticated, accomplishing everything in one pass.) The following two examples should make this clear. In the second example the bound parameter cannot be rewritten in the new sort.

*th$EOREM (Number or Name) *0.1$
Schema:
If (E! x)FM(x) then (E x)FM(x)

Replace for FM * A = pow(B)$

Which variable indicates the parameter places? *A$
Th. 0.1 Instance: A = pow(B) for FM
(i) If (E! A)A = pow(B) then (E A)A = pow(B)

What has happened here is that the program has implicitly substituted `set(x) & x = pow(B)' and rewritten the result replacing `(E! x)(set(x) & x = pow(B))' by `(E! A)(A = pow(B))' etc.

*ax$IOM (Number or Name) *sep$ARATION
Schema:
(E C)(A x)(x in C <-> x in B & FM(x))

Replace for FM * A is a subset of D$

Which variable indicates the parameter places? *A$
(E C)(A x)(x in C <-> x in B & set(x) & x sub D)

Do you want to specify for B *n$
A: SEPARATION Instance: A sub D for FM
(1) 'A B)(E C)(A x)(x in C <-> x in B & set(x) & x sub D)

In this example the program implicitly substituted `set(x) & x sub D' but could not rewrite the bound parameter `x' as a set variable because the rewritten formula is not a consequence of the result of the implicit substitution and the sort axioms.

In summary then the modified algorithm accomplishes what is desired: it allows the user to specify the instance he wishes without requiring him to confront distracting logical detail. The procedure is best possible in the sense that it is complete with respect to the sort axioms: it will allow every instance that can be obtained by instantiation and rewriting sorts--every instance that is, so to speak, a sort consequence of the schema.

3.2.3.4 Let Rule

During the last year a new inference rule LET was added to permit the introduction of a object with certain properties provided that it has been or can easily be established that such an object exists. Before students had to first prove (E x)FM(x) and then use ES to get (say) FM(y). LET combines these two steps into a single step. To use LET the student in effect types a sentence of the form: Let v be such that FM(v). The program will try to VERIFY (E v)FM(v) from the axioms, definitions, theorems, and lines cited. If it is successful it will generate a line of the form FM(v) (where v is now an ambiguous name). An example of the use of LET follows.

```
Wp      (2)  pow(A) <=: A

*2Let$  (variable) *f$  be such that
    (formula) *inj(f) and dom(f)=pow(A) and rng(f) sub A$
Using *def$INITION (Number or Name) *leq$uipollent
Using *def$INITION (Number or Name) *map$
Using *def$INITION (Number or Name) *injection$
Using *g$O
```

3.2.3.5 The REPLACE rule

The rules for replacing formulas by equivalent formulas and terms by equivalent terms were combined into a single rule REPLACE that replaces expression by equivalent expressions. Such generalization and coalescence while far from dramatic makes the system easier to understand and use. ALso the system becomes more natural in that logical niceties such as separate rules for replacing terms and formulas do not occur in standard mathematical practice; there one simply replaces equivalent expressions.

See the examples below for the details of how REPLACE is used. The

'Intro' and 'Elim' in the listing of options are intended as mnemonics for the case in which the equivalence is a definition. In such a case replacing the left hand side is eliminating the defined symbol and replacing the right hand side is introducing the defined symbol.

```
Derive:
If A sub B & B sub C then  A sub C


HYP      (1)  A sub B and B sub C
*lrep$LACE
Finish, Left(Elim), Right(Intro), or Print (F,L,R,P)? (F)*l$EFT
Will you wish to specify? (No) *$
Using *def$INITION . (Number or Name)  *sub$SET
Occurrences (ALT MODE for all) *$


Finish, Left(Elim), Right(Intro), or Print (F,L,R,P)? (F)*$
1 REPLACE Using:  Df. SUBSET
        (2)  (A x)(x in A -> x in B) and (A x)(x in B -> x in C)

*2vERIFY  (3)  *(A x)(x in A -> x in C)
Will you wish to specify? (No) *$
Using *g$O

*3rep$LACE
Finish, Left(Elim), Right(Intro), or Print (F,L,R,P)? (F)*r$IGHT
Will you wish to specify? (No) *$
Using *def$INITION  (Number or Name)  *sub$SET
Occurrences (ALT MODE for all) *$


Finish, Left(Elim), Right(Intro), or Print (F,L,R,P)? (F)*$
3 REPLACE Using:  Df. SUBSET
        (4)  A sub C
*
```

Use of REPLACE to prove the identity condition for ordered pairs

Derive:
$\langle x,y\rangle = \langle u,v\rangle$ iff $x = u \;\&\; y = v$

HYP (1) $\langle x,y\rangle = \langle u,v\rangle$
1 REPLACE Using: Df. ORDERED, Th. PAIR-IDENTITY, Df. SINGLETON, Th. PAIR-IDENTITY
(2) $[((x = u \;\&\; x = u) \text{ or } x = u \;\&\; x = u)$
$\&$
$(x = u \;\&\; y = v) \;V\; x = v \;\&\; y = u]$
V
$((x = u \;\&\; x = v) \;V\; x = v \;\&\; x = u)$
$\&$
$(x = u \;\&\; y = u) \;V\; x = u \;\&\; y = u$
2 TEQ (3) $x = u$ and $y = v$
TEQ (4) If $x = u \;\&\; y = v$ then $\langle x,y\rangle = \langle u,v\rangle$
(1,3 CP),4 LB
(5) $\langle x,y\rangle = \langle u,v\rangle$ iff $x = u \;\&\; y = v$

3.2.3.6 ZFSTART and ZFFINISH

When working in a metatheory one often has to establish that, under given conditions, certain results are provable in the object theory. One way to do this is to axiomatize the provability relation of the object theory in the metatheory and to establish the result from these axioms. However, it is often far easier to simply derive the result directly in the object theory and then use this fact in the metatheory.

Such procedures were recently added to the EXCHECK system for the proof theory course. Two inference procedures are involved: ZFSTART--for starting a derivation in ZF from the metatheory; and ZFFINISH for finishing the derivation in ZF and returning to the metatheory. After starting a ZF derivation from the metatheory you may reference prior results from the metatheory or the metatheoretic part of the derivation. In the second example below, two lines in the metatheoretic part of the derivation are referenced from the ZF part of the derivation. There is a

restriction on the form of metatheoretic results that may be referenced from inside ZF: they must be atomic formulas of the form ZF |- F or ZF* |- F. Conjunctions of such formulas are also allowed. Two simple examples of uses of these rules follow.

Example 1:

```
Derive:
IF Y IS Z THEN  ZF* |- |Y| = |Z|

*hyp$
HYP     (1)  Y IS Z
*zfs$TART
                  ****************  ZF  ****************
    #teq$  (2)  *|Y| = |Y|
Will you wish to specify? (No) *$
Using *g$O
    #zff$INISH
                  ****************************************
2 ZFFINISH
        (3)   ZF* |- |Y| = |Y|
*1,3teq$  (4)  *ZF* |- |Y| = |Z|$
Will you wish to specify? (No) *$
Using *g$O
*
```

Example 2:

```
Derive:
IF  ZF |- x=y AND  ZF |- y=z THEN  ZF |- x=z


WP      (1)  *ZF |- x=y
WP      (2)  *ZF |- y=z
*zfs$TART
                  ****************  ZF  ****************
    #1,2teq$  (3) *x=z$
Will you wish to specify? (No) *$
Using *g$O
    #3zff$INISH
                  ****************************************
3,1,2 ZFFINISH
        (4)   ZF |- x=z
```

3.2.3.7 VERIFY

The VERIFY command is designed to give the student a reasonably powerful method of verifying the correctness of a formula given prior results. For example, given A in B and B in pow(C) it is convenient for the student to be able to verify A in C simply using the definitions of subset and powerset. The guiding principle is that the student should be doing set theory (or probability or proof theory) and not first order logic. Alternatively, VERIFY should be able to prove anything that is obvious to the student (and correct) within a few seconds to a few minutes of real time.

A simple use of the prover by the VERIFY command would be:

```
        1) x in A
        2) A in pow(B)
1,2v$   3) x in B
USING *D$EFINITION (number or name) *SUB$SET
USING *D$EFINITION (number or name) *POW$ERSET
USING *G$O
*       [Failure to find a proof would cause a message to be printed.]
```

The student is able to cite prior lines by number, and prior axioms, definitions, and theorems by number or name. VERIFY attempts to use everything cited, and incorporates rather few theorems implicitly, so the student's judicious choice of these prior results is essential to successful application of this command. Since VERIFY is intended to be used as part of an integrated system, see the section on sample proofs for actual examples of it in use.

VERIFY uses what is basically a resolution theorem prover. The predecessor to this prover was written by Vesco Marinov; see (Marinov, 1973). It is a level saturation prover that uses a merge strategy to limit the growth in length of clauses. After the first

round, as a resolvant is generated at least two of its literals must merge for it to be accepted. Furthermore, the depth of terms in that clause must be at most one greater than the deepest term input. Equality is dealt with via demodulation. Whenever an equality becomes asserted, the simpler term is uniformly substituted for the more complicated term everywhere in the clauses being used, simplicity being primarily a measure of the depth of a term. While restrictions such as these seem at first glance a bit severe, they are empirically based, and have been chosen to maximize the range of "obvious" proofs obtainable in less than about 20 cpu seconds. Thus far, relaxing any of them has dramatically reduced the number of proofs obtainable in our domain. Paramodulation, for example, is hopelessly slow for us. Surprisingly, the prover has been able to get more proofs in our domain without its set of support strategy than with it. The proofs we do are small enough that it is apparently quite efficient to simply let forward and backward chaining meet in the middle.

While some work has been done to explore the standard logical characteristics of this prover, the emphasis to date has been upon extending it to be flexible enough to deal with features such as sorts and types, formula-binding terms, and answer extraction.

Sorts and types present special difficulties for a mechanical prover. For a resolution prover, the major effect of sorts is to restrict the unifications permitted. In our set theory, for example, ordinals are sets and sets are general objects, so a universally quantified set variable can be unified to an ordinal constant, but not to a general constant. To help effect this checking of sorts, each atomic term explicitly carries a list of its sort and all higher sorts.

Constants are given a sort during initialization, variables are implicitly sorted, and the global mechanism may note that some term is not in its usual sort. (We may say that A, normally a set, is actually an ordinal, or specifically not an ordinal.) Thus, given a universally quantified variable and an atomic term, the unification algorithm need only see if the sort of the variable appears in the list of sorts applicable to that term.

This simple scheme is complicated in three ways. First, the theories we use also have sort predicates. Thus, during the course of an attempted proof, we may generate new sort information that permits unifications that were previously blocked. The second complication arises in that the sort of a complex term depends dynamically upon the sorts of its substituent atomic terms. Thus, if an atomic term changes sort, even during unification itself, the sort of any complex term involving it may change. To dynamically sort complex terms, each operator is given a type during initialization, and the sort of a complex term is computed each time it must be referenced. Note that to unify two complex terms the sorts usually need not be computed, since if the operators are the same and the atomic terms unify, then the sorts of the complex terms must be the same. A third complication due to sorts arises when a formula is generated similar to:

(A x)(x is a set -> x is an ordinal) .

This is very powerful information and could dramatically speed up a proof if recognized, yet may yield no resolvants, since the literal set(A) may be suppressed as redundant. Currently such information is not very well used.

Another source of difficulty lies with formula binding operators such as abstraction (for example, {x: x in A and x in B}). The decision whether two abstraction terms can unify may require the full power of set theory. For example, is {i: i cubed equals j cubed plus k cubed; i,j,k integers} empty? When unifying {x: FM1(x)} with {y: FM2(y)} a reasonable approach is to attempt a proof of (A x)(FM1(x) <-> FM2(x)). Should it succeed, the two terms will unify. This strategy has the nice property of unifying abstraction terms which denote the emptyset by virtue of having inconsistent formulas, even if those formulas are quite different.

A refinement upon formula binding operators which we employ is a term and formula binding operator, for example the sequence [i: A(i) (i < n)], read the set of all A sub i such that i is less than n. To unify [x: TM1(x) FM1(x)] with [y: TM2(x) FM2(x)] the same strategy as above will work, except that the subproof must be of (A x)(sigma(FM1(x)) <-> sigma(FM2(x))), where sigma is the most general unifier of TM1(x) with TM2(x).

Biconditionals expand into conjunctive normal form in a most unfortunate way, yielding from P <-> Q the clauses (NOT P V Q) and (P V NOT Q). Thus if P is generated, Q will follow, and then P again, so that many duplicate clauses may be generated. Since the formulas may be biconditionals of biconditionals of ..., schemes to avoid this problem by looking at the clauses tend not to work. The effects are too diffuse by then. By splitting proofs of biconditionals into two proofs, one for each conditional, we obtained an order of magnitude increase in speed (from 80 to 8 seconds for one typical proof), bringing many proofs below the 20 second time limit we impose.

3.3 Proof Theory Course

3.3.1 Curriculum

Goedel's Incompleteness Theorems are presented in the course as formulated for the system ZF of set theory. The axioms of ZF and their intended models (segments of the cumulative hierarchy> are carefully described in the first part of Chapter 1. In the second part we recall how informal mathematical (in particular, number theoretic) notions can be represented in a subsystem ZF* of set theory. (ZF* is ZF without the axiom of infinity and of the same strength as arithmetic.) The logical form of these definitions is analyzed. In the third part, attention is given to the problem of representing number theoretic functions and predicates given by (informal) recursion or induction. We show that SIGMA-recursive functions can be introduced in a definitional extension of ZF*.

The informal metamathematical arguments involved in the above considerations serve as the motivation for a more rigorous description of the syntax of ZF. That description is actually given in the first part of Chapter 2. In the second part of the chapter, we analyze the syntactic objects as binary trees and formulate a theory for them (analogous to Peano-arithmetic). The theory is called TEM, and provides a framework for describing and comparing formal systems. Finally we indicate how syntactic notions can be presented in TEM.

In Chapter 3 Goedel's First and Second Theorems are proved--assuming basic representability and derivability conditions. Some examples of nonstandard representations of the theorem are given; they show that the derivability conditions are crucial to the Second Theorem.

3.3.2 Audio Lessons in Proof Theory

The material described above formed the basis of the CAI course in proof theory. During the past year, the VOCAL language was used to prepare lessons in a lecture style format with audio. The display features of VOCAL were particularly helpful in describing the tree-like structure of (well-founded) sets and syntactic objects. The text of the lessons was presented in Prosody Mode. That is, the text to speech step was achieved by concatenation of recorded words. The syntax of the each spoken expression was also automatically analyzed, and the audio parameters of the individual words adjusted to fit the syntactic analysis. See (Hinckley, et. al., 1977).

3.3.3 Augmentation of the Proof Checker

In the second year report (Smith and Suppes, 1976) we mentioned that TEM was fitted straightforwardly into the existing proof-machinery, and that the central results were proved on the computer. Yet for those proofs we used (in addition to the proof theoretic conditions mentioned above) some metamathematical rules. To dispense with the latter, ZF and ZF* were also implemented and a 'switching mechanism' was devised for the latter of these (see Section 3.2.3.6). This allows conceptually clearer derivations of the main theorems.

4 Teaching Initial Reading: Evaluation of Audio

4.1 Letter Experiment

This section describes one of the experiments on recognition of computer generated speech. In this experiment the ability of first graders to recognize individual letter sounds was tested.

4.1.1 Experimental Set-up

48 first graders were selected by the teachers of three classes at the Willow School in Menlo Park, California. 12 students from each class made up 3 treatment groups, 4 additional students from each class made up the control group of 12 students. Each treatment group received 7 sessions of taped, computer generated speech and an 8th session with taped human speech. The control group received 8 sessions of taped human speech.

The sessions consisted of listening to 26 items, each consisting of the carrier phrase, 'Circle the letter:', followed by a letter of the alphabet, and after each item heard, circling the letter name from 3 choices on an answer sheet. The two confusion choices came from two sets of letters which were used on alternate sessions. Each session covered the alphabet without repetition, in one of eight random orderings. Approximately 6 seconds after each item was presented, the subjects heard a beep and the correct answer was displayed on a flash card. The total time between items was approximately 9 seconds. The duration of the items was approximately 3 seconds, so an entire session took 5 to 6 minutes to present.

The sessions were presented in groups of two, over a two day

period; two sessions each morning and two each afternoon. The first session in each group was followed by a short (5 minute) pause before beginning the next session. At each session, the subjects sat around a table at the head of which was an experimenter with flash cards and tape recorder. Each subject was supplied with an amplifier, headset, answer sheet and pencil.

Before the first and fifth sessions the experimenters introduced themselves and the equipment to the subjects, and told them that they were going to hear a computer 'talk'. The subjects were informed that the experimenters were interested in how well the computer talked, so that it was the computer which was being tested, not the subjects. They were then told that the task consisted of listening to each item, circling the letter they heard on the appropriate row of the answer sheet, and look up at the flash card when they heard the beep. They were told that the purpose of the flash cards was to help them understand the way the computer talked.

4.1.2. The Computer Systems for Speech Generation

Four different systems for computerized speech synthesis were used for this experiment, and the one using words. One system (referred to below as MIT) is a sophisticated phonemic synthesizer developed by Professors Jon Allen and Dennis Klatt at the Massachusetts Institute of Technology under NSF support. The MIT system converts text by rule into the control parameters for the synthesizer and thus into speech. The MIT tapes for this experiment were prepared in two stages. First, the text was converted into phonetic commands on Professor Allen's PDP-9. Then, these commands were used by Dennis Klatt's program on a PDP-20

which generated a digital representation of the speech. This was coverted to an analog speech signal by a digital to analog converter on the PDP-20. Two other systems were the VS6 and the ML1, commercial systems produced by Votrax, a division of the Federal Screw Works Inc. The VS6 system was used for the letter experiment, and the more expensive ML1 was used for the word experiment. These systems are also phonemic synthesizers, similar in this respect to MIT, but the phonetic control parameters are generated by hand, rather than by rule, and, these systems allow less control over the allophones then the MIT system. The Votrax tapes for both experiments were prepared by Dr. Carol Simpson, of the Psycho-Linguistics Research Company. The fourth synthesis system involved in the experiment was the Micro-Intoned Speech Synthesis (MISS) system developed here at the Institute under NSF support (referred to below as LPC). It uses recorded words, which are digitized and then compressed for storage on disk using a linear predictive coding (LPC) algorithm. Sentences for the experiment, made up of either a letter or word together with a carrier phrase were formed by concatenation from a vocabulary of stored words, with parameters adjusted according to a syntactic prosody algorithm (developed at the Institute), then expanded using the LPC algorithm, and converted to an analog signal.

It should be emphasized that we are comparing systems which are quite different both in kind and to a certain extent in purpose. Neither the MIT system or the LPC system is commercially available in any form while the Votrax systems are currently being marketted. Also, a vastly different amount of human intervention is required in the three devices. The MIT system requires no human intervention once the text is

presented to the system; however, it does not operate in real time. The Votrax systems require a trained phonetician to manually transform text into phonetic commands to drive the synthesizers although some "text-to-Votrax" command systems are under development. Finally, the LPC system requires initial human interaction to record the individual words, but once all the words needed are in the vocabulary, it converts text to speech both automatically and in real time. Despite the fact that these systems require differing amounts of human intervention and operate at various speeds, they all purport to allow computers to talk to people and it is on that basis that we are comparing them.[18]

4.1.3 Comparison of Mean Scores for Letters

Although the test for individual leters in isolation is not an easy one, the students did rather well: the mean correct scores for each session were between 83 and 98 percent; see Figure 4.[19] The variances were relatively large, so there was insufficient separation of the mean scores for session by session comparison. The difference between mean scores exceeds the sum of the standard deviations in only 6 cases: control over MIT in sessions 3 and 4; control over Votrax in sessions 4,6, and 7; LPC over Votrax in session 6.

---

[18] We originally proposed to test also the delta modulation system (developed and formerly used at the Institute). However, the delta system is not comparable in quality to the other systems, and by not testing it, we were able to increase the number of subjects hearing the other systems.

[19] Do to the absence of some student who began the experiment, and exclusion of two students who did not adequately respond to the task, and one outlying score, some of the mean scores are based on less than the original 12 subjects. For the control, 11 scores were used to compute the means for sessions 7 and 8. For Votrax, 11 scores were used to compute the mean for session 3, and 9 scores for sessions 5 through 8. The means for MIT and LPC were computed using 11 scores for all sessions.

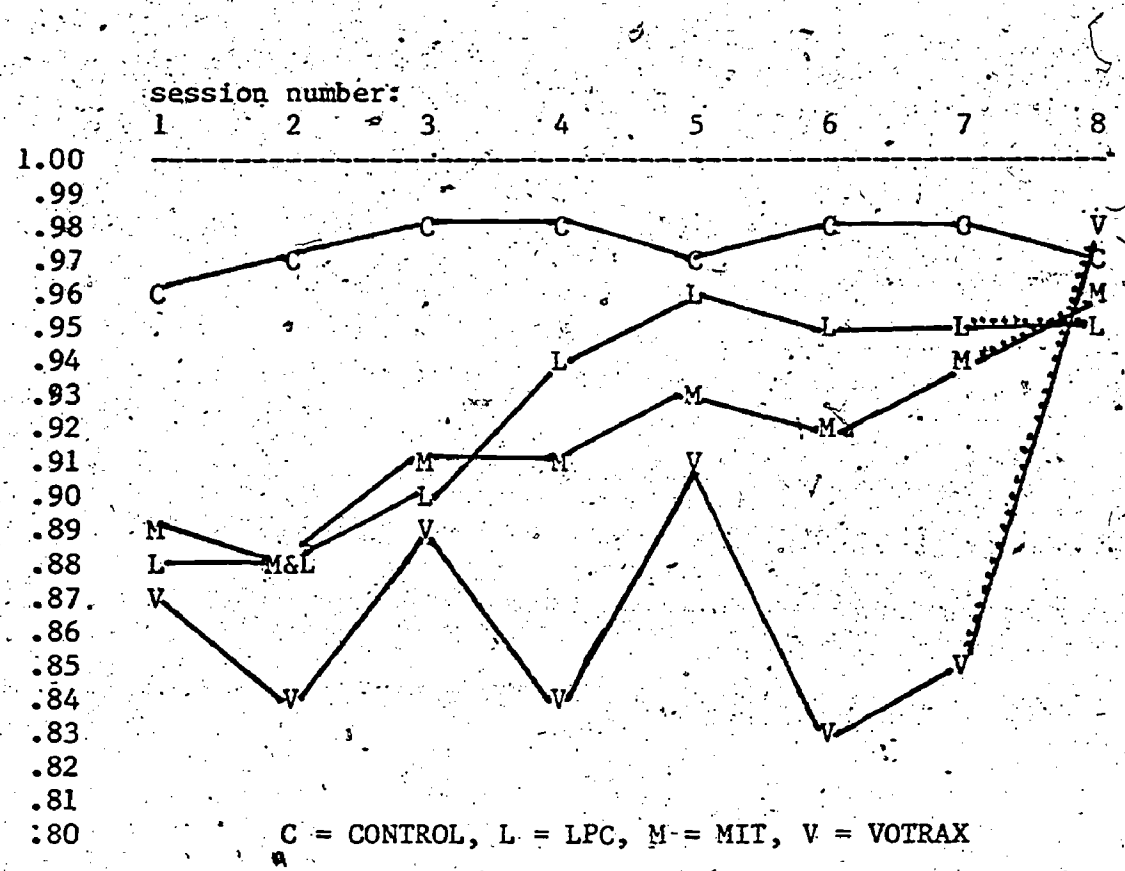


Figure 4. Mean Scores, Letter Experiment, by Session.

A more salient aspect of the data is the regularity over the first seven sessions of the rank of the systems. A sign test was used to examine this feature of the data. We began with a null hypothesis that the probability on any given session of system a having a higher mean score than system b is one half. We then used a binomial distribution to compute the probability under the null hypothesis of system a scoring higher than system b in x sessions out of seven. The scores on the eighth session in which each class heard the recording of the human voice did not indicate a significant difference between the four groups, so we assume any differences detected are the result of differences in the systems used. The control group scored highest in seven out of seven sessions. Both MIT and LPC scored higher than Votrax in seven sessions. LPC scored higher than MIT in four sessions, lower in two

sessions, and the two groups scored the same in one session.[20] The results of these sign tests are that evidence for the superiority of the human voice over any of the computer systems is significant at the .008 level, as is evidence of the superiority of both LPC and MIT over Votrax. The probability of MIT doing as well or better than the 4-2-1 outcome in the comparison to LPC, under the null hypothesis of equal quality, is .363.

4.1.4 Learning Study for Letters.

The model selected for study of the mean learning curve is linear in the change in the probability of error:

$$q_{n+1} = aq_n$$

where q_n is the probability of error on trial n, and a is the factor by which the probability of error decreases in a single session. In this case, q_n is the mean error probability during session n, averaged over students and letters of the alphabet. The parameters we need to estimate for this model are q_1 and a, since for any $n > 0$, $q_n = a^n q_1$.

Averages of the error probabilities for the first two sessions were used as the estimates for the initial probability of error for each system, because the results of the initial session were affected by the students unfamiliarity with the specific task.[21] Given $\hat{q}_1$, we then found

[20] In accordance with the assumption underlying the null hypothesis, we assumed the probability of an equal score indicating superiority for either system to be one half. Thus, an average was taken over the probability of LPC scoring higher 4 times, and the probability of LPC scoring higher 5 times.

[21] Do to a scheduling confusion on the first day of the letter experiment, the groups hearing the MIT and LPC tapes were switched, so that from session 3 through session 8, the group that started with MIT heard instead the LPC tapes, and the original LPC group heard the MIT tapes. Since the average error probabilities over the first two lessons were so close (LPC: .115, MIT: .114), and the scores on the second

the maximum likelihood estimate $\hat{a}$ for a, and $\hat{\sigma}$ for σ, assuming that the observed values were normally distributed with variance: σ^2, about the mean value: $a^n q_1$. The maximum likelihood estimates were computed using the remaining five data points in the case of the computer systems, and the remaining six points for the control.

Table 1

Estimated Parameters for Linear Model

system	a	q_1	σ	additional sessions (n) to criterion ($q_n < .024$)
LPC	.791	.115	.012	2
MIT	.891	.114	.006	9
VOTRAX	.985	.148	.033	115
CONTROL	.930	.034	.007	0

An average over the relatively stable error values in the final six sessions with human speech was used to set a criterion for performance for the computer systems. The criterion thus computed was a probability of error $q = .024$. The estimates obtained for the initial probability of error and the decremental factor were then used to compute estimates of the number of additional sessions (sessions beyond the seven sessions used to estimate the initial error rate and the decremental factor) needed for the given computer systems to reach the criterion.

The linear learning model was selected for its simplicity and robustness. The data seems insufficient for comparison with a more complicated model. A rough judgement of the goodness of fit of the

session exactly the same, all the LPC scores were taken together as a 7 session run, and likewise with the MIT scores.

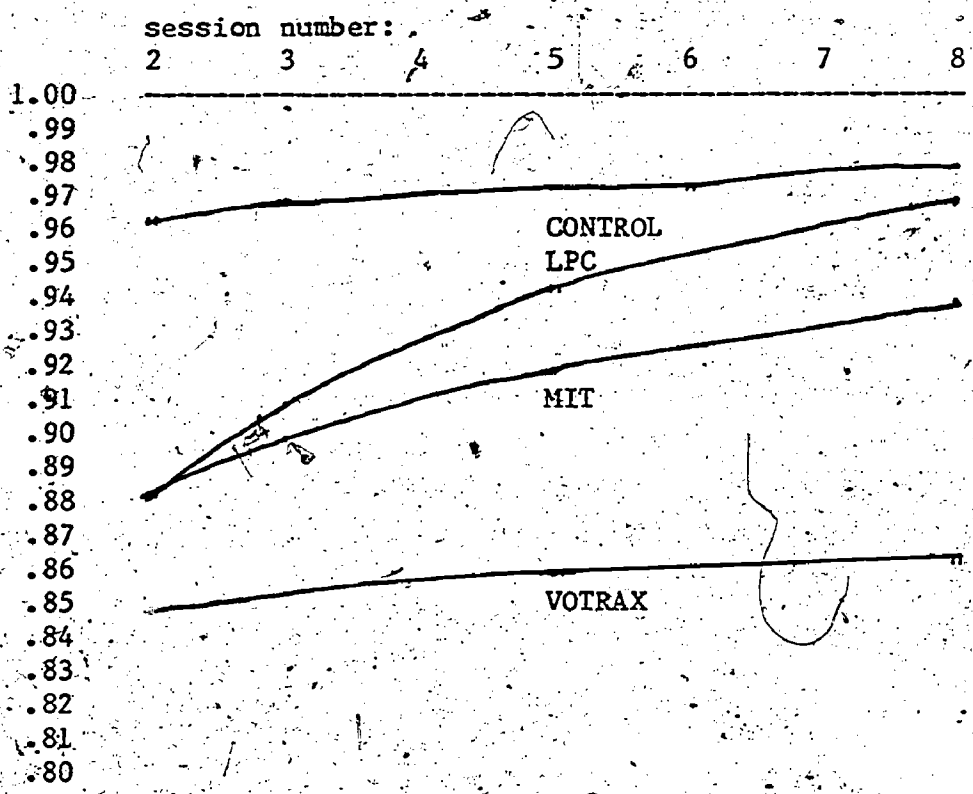


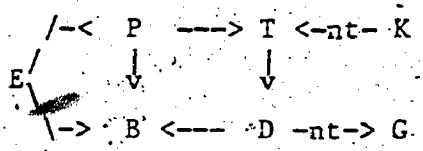
Figure 5. Predicted Scores (Learning Model) Letters

model can be obtained by comparing the observed and predicted scores for the letter experiment; see Figure 4 and Figure 5, noting in particular the poor fit to the Votrax data. Also of interest regarding the fit of the linear model is the estimated standard deviation of the normal distribution of error about the predicted means. This parameter and the other estimated parameters are contained in Table 1, which also contains estimates of the number of additional sessions to criterion. Since each session is approximately 6 minutes long, the estimates for the number of sessions to criterion mean an additional 12 and 54 minutes of exposure respectively for LPC and MIT to meet the criterion of human recorded speech. The model predicts the need of an additional 11.5 hours of exposure for Votrax to meet the criterion. However, the variance of the Votrax estimate for a is quite high; a value of a one standard deviation

below the mean would lead to a prediction of only 3.2 additional hours exposure to criterion. Also, a value of a less than half of one standard deviation greater than the mean would imply that for Votrax, no learning is occurring.

4.1.5 Evaluation of Specific Problem Sounds.

Several criteria were used in an attempt to focus on the specific problem letters for each of the systems. The first of these was to select the letters for which the sum over seven sessions of the percentage of students mistaking that letter exceeded 67 percent. We also checked, which of those letters had no mistake-free sessions out of the first seven. Under the first criterion, Votrax had 12 problem letters: D, E, F, J, K, N, P, Q, T, V, X, and Z; 5 of which (F, K, T, V, Z) were also problems under the second criterion. MIT had 5 first criterion problem letters: C, G, J, N, and Z; all but C were also second criterion problems. LPC had 4 first criterion problems: B, D, G, and Z; of which just Z was also a second criterion problem. For comparison, the control had one first criterion problem (N), and no problems under the second criterion.



VOTRAX

arrows indicate direction of shift
nt = indicated shift not tested

Figure 6. Diagram of Stop Shifts; Letter Experiment

The data was also examined to see if patterns of errors could be found in terms of human phonetic parameters. Only in the case of the data for Votrax was there sufficient error information to detect some patterns. There we found a tendency to shift from unvoiced to voiced stops, maintaining the place of articulation, and a tendency to shift the place of articulation of other stops, while maintaining the voiced or unvoiced quality. See Figure 6 for a diagram of these stop shifts, and note also in that diagram the shifts from P to E and E to B, paralleling the direct shift from P to B. Other problems for Votrax were strong reciprocal confusions between K and J, and between Z and V, and a strong tendency to hear S for the Votrax F.

All the systems had difficulty with the letter Z, which was in general heard as V, however, only Votrax had the reciprocal confusion. Votrax, MIT, and the control all had difficulty with N, which was heard as M.[22] Both MIT and LPC had a problem with G, which was heard as either B or D in both systems. There is evidence of vowel confusion in the case of the J to G shift in the MIT system.

4.2 Word Experiment

This section describes the second of the experiments on recognition of computer generated speech. In this experiment the ability of fifth graders to recognize initial and final consonant (or consonant cluster) sounds was tested.

[22] N was not a first criterion problem letter for LPC, having a summed error percentage of 45 percent

4.2.1 Experimental Set-up.

48 fifth graders were selected by the teachers of three classes at the Willow School in Menlo Park, California. 12 students from each class made up 3 treatment groups, 4 additional students from each class made up the control group of 12 students. Each treatment group received 8 sessions of taped, computer generated speech and a 9th session with taped human speech. The control group received 9 sessions of taped human speech.

Table 2

Initial Consonant Sounds, with Confusion Words

test word		confusion words		test word		confusion words	
b	bear	dare	pear	s	sink	zinc	think
bl	block	flock	clock	sc	scare	care	stair
br	breathe	wreathe	sheathe	scr	screw	threw	stew
k	cash	dash	crash	sh	shop	stop	chop
ch	chose	close	shows	sk	skin	thin	spin
cl	clean	lean	seen	sl	sleeve	leave	weave
cr	crash	sash	cash	sm	smell	swell	fell
d	dime	time	lime	sn	sniff	stiff	miff
dr	drip	grip	trip	sp	spark	stark	shark
f	fast	cast	past	spl	split	flit	spit
fl	flock	block	clock	spr	spring	string	sing
fr	free	fee	three	st	start	part	tart
g	gate	date	great	str	string	spring	thing
gl	glow	low	go	sw	switch	which	ditch
gr	grade	braid	trade	t	task	cask	mask
h	hand	and	land	tr	trip	drip	strip
j	jump	bump	lump	tw	tweed	weed	reed
l	look	hook	book	th^	those	nose	shows
m	march	arch	starch	th	thin	fin	spin
n	nice	rice	mice	thr	three	tree	spree
p	post	boast	toast	v	verb	curb	herb
pl	plank	prank	blank	w	went	bent	meant
pr	price	rice	slice	y	year	ear	hear
qu	quack	crack	pack	z	zoo	do	sue
r	rasp	grasp	clasp	(th^ = voiced alveolar fricative)			

Table 3

Final Consonant Sounds, with Confusion Words

test word	confusion words			test word	confusion words		
b	tub	tug	tough	p	shop	shot	shock
ch	switch	swish	swim	pt	rapt	rack	rat
d	grade	great	grain	r	bear	bend	bet
f	oaf	oat	own	rb	verb	verse	verge
ft	left	let	leg	rch	march	marsh	mark
g	rug	rub	rut	rd	bird	birch	birth
j	fudge	fuzz	fund	rf	turf	turn	turk
k	block	blot	blond	rj	large	lark	lard
kst	next	neck	nest	rk	spark	spar	spot
l	all	or	awe	rl	curl	curb	curve
lb	bulb	bulk	bulge	rm	warm	war	warn
lch	mulch	much	munch	rn	barn	bark	bar
lf	self	sells	said	rp	harp	hard	heart
lj	bulge	bulk	bulb	rs	horse	horn	hoard
lk	milk	mills	mink	rsh	marsh	march	mark
lm	palm	pond	park	rt	start	starch	star
lp	help	health	held	rth	earth	earn	urge
ls	false	fault	fall	rv	starve	stars	start
lt	fault	falls	false	s	price	prize	pride
lth	health	help	held	sh	cash	cast	camp
lv	delve	dell	deaf	sk	task	tack	tax
m	dime	dire	dine	sp	rasp	rat	rash
mp	camp	can't	cam	st	fast	fats	fan
n	skin	skiff	skim	t	gate	gave	gain
nch	crunch	crumbs	crutch	th	path	pass	pad
nd	hand	ham	had	th^	breathe	breeze	brief
nk	plank	plant	plaid	v	gave	gaze	gate
nth	tenth	tent	tends	z	chose	chore	choke
nt	went	when	wet	zh	beige	bathe	bays
ng	spring	sprig	sprint	(th^ = voiced alveolar fricative)			

The first 8 sessions consisted of listening to 27 items, each consisting of the carrier phrase, 'Circle the word:', followed by a monosyllabic word, and after each item circling the word heard from 3 choices on an answer sheet. The taped words came from a list of 108 items (including some repeats -words used to check both an initial and final consonant sound) to check 49 initial consonants or consonant clusters (see Table 2), and 59 final consonants or consonant clusters

(see Table 3). The list was presented twice in different random orderings: once in the first 4 sessions, again in the next four sessions. The ninth session was a control session with letters, which used the same ordering as the eighth session of the letter experiment, and was given shortly after the eighth session of words. Apart from these differences, the experimental set-up was exactly as in the letter experiment; see: Section 4.1.1. For a description of the computer systems used see: Section 4.1.2.

4.2.2 Comparison of Mean Scores for Words.

The scores on the word experiment were quite high: the mean correct scores for each session were between 78 and 100 percent; see Figure 7.[23] The variances were relatively large, as in the letter experiment, so there was insufficient separation of the mean scores for session by session comparison. The difference between mean scores exceeds the sum of the standard deviations in only 9 cases: control over MIT in session 7; control over Votrax in all but sessions 6, 8, and 9; LPC over Votrax in sessions 4 and 5. A more salient aspect of the data is the regularity over the first eight sessions of the rank of the systems. As in the letter experiment a sign test was used for a pairwise comparison of the systems; see Section 4.1.3 for a description of the test.

The control group scored higher than Votrax and MIT in eight out of eight sessions, and higher than LPC in seven sessions. Both MIT and LPC scored higher than Votrax in eight sessions. LPC scored higher than MIT

[23] Do to the absence of some student who began the experiment, some of the mean scores are based on less than the original 12 subjects. For the control, 10 scores were used to compute the means for sessions 5 through 9. For MIT, 9 scores were used to compute the mean for sessions 5 through 9.

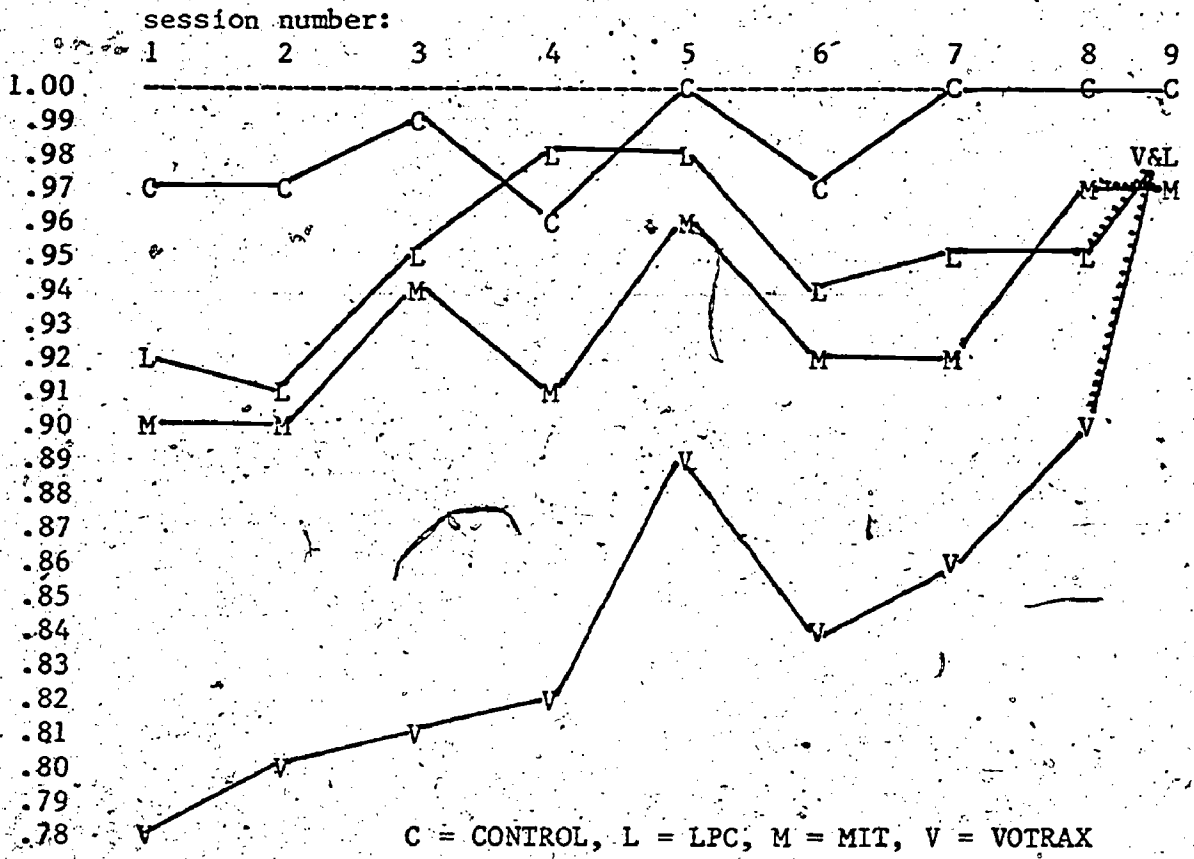


Figure 7. Mean Scores, Word Experiment, by Session.

in seven sessions and lower in one session. The results of these sign tests are that evidence for the superiority of the human voice, LPC, and MIT over Votrax is significant at the .004 level, as is evidence of the superiority of the human voice over MIT. The evidence for the superiority of the human voice over LPC, and of LPC over MIT is significant at the .035 level.

4.2.3 Evaluation of Specific Problem Sounds.

In order to focus on the specific problem consonant sounds for each system, lists of items that more than x percent of the students missed (on either of the sessions that the item was tested) were prepared for x = 15, 25, 33, 50, and 67; see Table 4. Each list was checked to see

what patterns of errors could be discerned in terms of human phonetic parameters. For the most part, the 33 percent error range was most amenable to analysis.

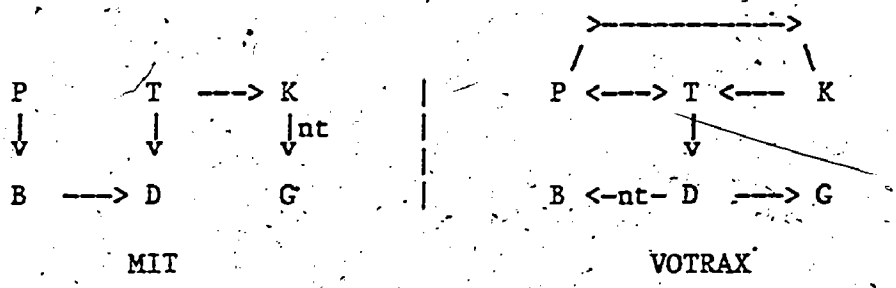
Table 4

Number of Problem Sounds

Error Level:	15%	25%	33%	50%	67%
			initial consonants		
VOTRAX	27	17	13	5	1
MIT	15	12	10	4	1
LPC	11	8	5	2	2
CONTROL	3	1	1	1	0
			final consonants		
VOTRAX	35	26	17	7	3
MIT	10	8	6	2	0
LPC	10	6	2	1	0
CONTROL	3	2	1	1	0
			totals		
VOTRAX	62	43	30	12	4
MIT	25	20	16	6	1
LPC	21	14	7	3	2
CONTROL	6	3	2	2	0

The data for LPC indicate, aside from isolated errors, problems with place of articulation. However, since most of these place errors concern the th (unvoiced theta) sound, it may be more appropriate to say that LPC has a place of articulation problem with the th sound, confusing it with both the s and f consonant sounds. Of the isolated errors, the most notable were hearing the kr cluster for kw (orthographic q) and dropping the y sound in 'year'.

Most of the problems for MIT occurred in consonant clusters, rather than in individual consonant sounds. There was a tendency for the p, t, and s sounds to be dropped from initial consonant clusters beginning



arrows indicate direction of shift
nt = indicated shift not tested

Figure 8. Diagram of Stop Shifts, Word Experiment

with these sounds (the effect did not appear notably in the final clusters). There was a strong tendency to shift from voiceless to voiced sounds, as in: p to b, t to d, and s to z. Place of articulation problems were noted in several stops and with the th sound. It would appear that MIT has problems with stops and with the s and th sounds. A surprising result was the shift from j to k in final clusters where the j followed an l or r sound.

Votrax had problems with stops being dropped from an initial position in a consonant cluster, both with initial and final clusters. There were also place of articulation problems with stops, and a tendency to shift from unvoiced to voiced stops. There were also problems with th shifting to s and f, as in the other systems, and a surprising shift from th to dz. As with MIT, there were problems with final lj and rj, but they shifted in a less surprising manner to lb and rd. There was also a tendency for both b and d to shift to _. The pronounced tendency toward reciprocal shifting is indicative of problems with the overall clarity of Votrax speech, and makes an analysis in terms of patterns of error more difficult. Of some help in this regard is an examination of the list of items with error percentages of 50

percent or more. The errors at or above the 50 percent level would seem to indicate that for Votrax, place of articulation errors are more pronounced than voicing errors.

In Figure 8 we have diagramed the shifts in stops for both Votrax and MIT. The voiceless stops have been placed over the voiced stops, maintaining the place of articulation. Sounds in both the voiced and voiceless series are placed with respect to position in the mouth: labial, alveolar, and velar. Some corroboration of the Votrax pattern can be found in Figure 6 which gives the same sort of diagram for Votrax stops in the letter experiment.

Although we were only testing consonant sounds, there were a few clear vowel problems for MIT ('block' sounded like 'black') and Votrax ('left' sounded like 'lift').

The only serious problem for the human speaker was the th sound shifting to f.

4.3. Use of Computer Generated Speech in CAI in Initial Reading.

The high probability of recognition of sounds as evidenced in the letter and word experiments indicates that some form of computer generated speech is adequate for use in computer assisted instruction in initial reading. The scores for LPC and MIT, which were generally well above 90 percent correct on both the letter and word experiments, is strong evidence of the adequacy of these systems. The scores for Votrax, generally between 80 and 90 percent are somewhat less impressive. Some attention, however, will have to be paid to the specific problem sounds for any system chosen, and effort made to provide extra practice on those items by appropriately altering the

curriculum. Of major importance in this regard, is the amount of time that must be spent in teaching a child to understand the speech system as opposed to teaching reading. Here again, as shown in Table 1, the extra time required by MIT and LPC seems sufficiently small. The extra time required by Votrax, however, might lead to the unpleasant result of postponing or even excluding the teaching of words containing certain problem phonemes. Cost analysis will also be a major issue in that the more adequate systems of computer generated speech, such as LPC and MIT are still far too expensive for widespread classroom use.

References

Atal, B. S., and Hanauer, S. L. "Speech analysis and synthesis by linear prediction of the speech wave." Journal of the Acoustical Society of America, 1970, 50, 637-655.

Benbassat, G. Time-domain two-dimensional pitch detection (Tech. Rep. 267). Stanford, Calif.: Stanford University, Institute for Mathematical Studies in the Social Sciences, 1975.

Boyer, R. S. and J. S. Moore, "The Sharing of Structure in Theorem-proving Programs," B. Meltzer and D. Michie (eds.), Machine Intelligence, vol. 7, John Wiley and sons: New York, pp. 101-116, 1972.

Boyer, R. S., Locking: A Restriction of Resolution, Ph.D. Thesis, University of Texas at Austin, Texas, 1971.

Copi, I. M. Symbolic logic (2nd ed.). New York: Macmillan, 1965.

Gaitenby, J. H. "The elastic word", Status Report on Speech Research, Haskins Laboratories, New York: SR-2, 3.1-3.12, 1965.

Goldberg, A. A generalized instructional system for elementary mathematical logic (Tech. Rep. 179). Stanford, Calif.: Stanford University, Institute for Mathematical Studies in the Social Sciences, 1971.

Goldberg, A., and Suppes, P. "A computer-assisted instruction program for exercises on finding axioms." Educational Studies in Mathematics, 1972, 4, 429-449.

Goldberg, A., and Suppes, P. "Computer-assisted instruction in elementary logic at the university level." Educational Studies in Mathematics, in press.

Goldsmith, J. English as a tone language. Unpublished manuscript, Massachusetts Institute of Technology, 1974.

Gray, A. H., and Markel, J. D. "A spectral-flatness measure for studying the autocorrelation method of linear prediction of speech analysis." IEEE Transactions on Acoustics, Speech, and Signal Processing, 1974, ASSP-22(3), 207-217.

Green, C., "Application of theorem proving to to problem solving," Proceedings of the First International Joint Conference on Artificial Intelligence, pp. 219-239, 1969.

Hinckley, M., Laddaga, R., Prebus, J., Smith, R. and Ferris, D. VOCAL: Voice Oriented Curriculum Author Language., Institute for Mathematical Studies in the Social Sciences, Technical report (forthcoming), Stanford University, Stanford, California 1977.

Huggins, A. W. F. "Just noticeable differences for segment duration in natural speech." Journal of the Acoustical Society of America, 51.4, pp.1270-1278, 1972.

Hunt, E. B., Artificial Intelligence, Academic Press, Inc.: New York, 1975.

Kane, M. T. Variability in the proof behavior of college students in a CAI course in logic as a function of problem characteristics (Tech. Rep. 192). Stanford, Calif.: Stanford University, Institute for Mathematical Studies in the Social Sciences, 1972.

Kaplan, R., et. al. Seminar in Psychology, Philosophy and Linguistics, Stanford University, Winter, 1977.

Klatt, D. H. Segmental duration in English, Journal of the Acoustical Society of America, 59.5, pp.1208-1221, May 1976.

Leben, W. R. "The tones in English intonation." Linguistic Analysis, 1976, 2.

Lehiste, I., J. P. Olive and L. A. Streeter, "Role of duration in disambiguating syntactically ambiguous sentences." Journal of the Acoustical Society of America, 60.5, pp. 1199-1202, 1976.

Levine, A. English intonation and computerized spech synthesis, Ph.d dissertation, Stnaford University 1977. Also available as Institute for Mathematical Studies in the Social Sciences, Technical report no. 287, Stanford University: July, 1977.

Levine, A. Report on prosodic generation, Institute for Mathematical Studies in the Social Sciences, Technical report no. 272, Stanford University: June, 1976.

Liberman, M. Y. The intonational system of English. unpublished doctoral dissertation, Massachusetts Institute of Technology, 1975.

Makhoul, J. "Spectral analysis of speech by linear prediction." IEEE Transactions on Audio and Electroacoustics, 1973, AU-21(3), 140-148.

Makhoul, J., and Viswanathan, R. Quantization properties of transmission parameters in linear predictive systems (Rep. 2800). Cambridge, Mass.: Bolt Beranek and Newman, 1974.

Marcus, M. " 'Wait and See' strategies for parsing natural language, Working papers 75, MIT-AI laboratory, 1974.

Marcus, M. A design for a parser for English, ACM'76, Proceedings of the Annual Conference pp. 62-68, October, 1976.

Marinov, V. G. Maximal Clause Length Resolution, Doctoral Dissertation, University of Texas, Austin Texas, 1973.

Markel, J. D., and Gray, A. H. "A linear prediction vocoder simulation based upon the autocorrelation method." IEEE Transactions on Acoustics, Speech, and Signal Processing, 1974, ASSP-22(2), 124-134.

Mates, B. Elementary logic. New York: Oxford University Press, 1965.

Moloney, J. M. An investigation of college student performance on a logic curriculum in a computer-assisted instruction setting (Tech. Rep. 183). Stanford, Calif.: Stanford University, Institute for Mathematical Studies in the Social Sciences, 1972.

Rawson, F. L. Set-theoretical semantics for elementary mathematical language (Tech. Rep. 220). Stanford, Calif.: Stanford University, Institute for Mathematical Studies in the Social Sciences, 1973.

Robinson, G. A. and L. Wos, "Paramodulation and theorem proving in first order thoeries with equality." in B. Meltzer and D. Michie (eds.), Machine Intelligence, vol. 4, American Elsevier: New York, pp. 135-150, 1969.

Sanders, W. R., Benbassat, G. V., and Smith, R. L. "Speech synthesis for computer assisted instruction: The MISS system and its applications." SIGCSE Bulletin, 1976, 8(1), 200-211.

Schank, R., and K. Colby (eds.), Computer models of thought and language, Freeman Press, 1973.

Selkirk, E. O. The phrase phonology of English and French, unpublished doctoral dissertation, Massachusetts Institute of Technology, 1972.

Smith, N. W. A question-answering system for elementary mathematics (Tech. Rep. 227). Stanford, Calif.: Stanford University, Institute for Mathematical Studies in the Social Sciences, 1974.

Smith, R. L. and Blaine, L. H. "A generalized system for university mathematics instruction." SIGCUE Bulletin, 1976, 8(1), 280-288.

Smith, R. L. and Suppes, P. Research on Uses of Audio and Natural Language Processing in Computer-assisted Instruction., Institute for Mathematical Studies in the Social Sciences, Technical report no. 275, Stanford University: August, 1976.

Smith, R. L., Graves W. H., Blaine L. H., and Marinov, V. G. "Computer-assisted axiomatic mathematics: Informal rigor." O. Lacarme and R. Lewis (Eds.), Computers in Education, IFIP (Part 2). Amsterdam: North Holland, 1975. Pp. 477-482.

Smith, R. L., Smith, N. W., and Rawson, F. L. Construct: In search of a theory of meaning (Tech. Rep. 238). Stanford, Calif.: Stanford University, Institute for Mathematical Studies in the Social Sciences, 1974.

Suppes, P. Axiomatic set theory. Princeton, N.J.: Van Nostrand, 1960.

Suppes, P. Introduction to logic. Princeton, N.J.: Van Nostrand, 1957.

Suppes, P. "Computer-assisted instruction at Stanford." Man and computer. (Proceedings of international conference, Bordeaux, 1970.) Basel: Karger, 1972.

Suppes, P. "New foundations of objective probability: Axioms for propensities." P. Suppes, L. Henkin, C. G. Moisil, and A. Joja (Eds.), Logic, methodology, and philosophy of science IV: Amsterdam: North-Holland, 1973.

Suppes, P., Jerman, M., and Brian, D. Computer-assisted instruction: Stanford's 1965-66 arithmetic program. New York: Academic Press, 1968.

Suppes, P., Smith, R., and Beard, M. University-level computer-assisted instruction at Stanford: 1975 (Tech. Rep. 265). Stanford, Calif.: Stanford University, Institute for Mathematical Studies in the Social Sciences, 1975.