THE ARGONNE BRAILLE TRANSLATOR

by

Lois C. Leffler

Introduction

The work I'm describing was done at Argonne National Laboratory with Arnold Grunwald as Project Leader. It was supported under a grant from the Department of Health, Education and Welfare.

The project grew out of work reported in Science in 1966 by Mr. Grunwald.[1] In this article, he showed that by using a continuous stream of Braille running under the reader's fingers reading rates of up to 320 words per minute could be achieved. This compares favorably with reading rates for sighted readers and is in marked contrast to the rates of 50 to 60 words per minute usually reported.

Arnold has a blind son, Peter. First-hand observation of the problems Peter had getting Braille materials led Mr. Grunwald to think of ways modern technology could facilitate the production, distribution, and ultimate consumption of reading matter for the blind.

The development of our Braille production and distribution system is an attempt to answer this need for compact, timely, readily available, and low real-cost Braille translations for use by the blind community. Our computerized Braille translator is to be a part of this complete system. The heart of the system is the Argonne Braille Machine. This device causes information stored on an ordinary magnetic tape to be transformed into a continuous stream of Braille characters which are then formed on a moving plastic belt. The reader places his fingertips on the belt, senses the Braille characters and thereby reads. The magnetic tape holds the equivalent of one inkprint book on a 3" reel with room left over for indexing and the reader's notes. The computer translation algorithm we developed was implemented to produce magnetic tape for the Braille machine, partially eliminating the need for trained Braillists.

In the papers which you are hearing at this meeting, different computerized Braille translators are described or implied. You might well ask: "Why did the Argonne staff develop their own translator?" There are two main reasons. The existing programs were most often written in machine or assembly language for machines other than the computer complex used at Argonne National Laboratory. DOTSYS III in COBOL and another program in PL/I have been written since our work began. Further, most existing programs are highly specialized. For instance, some are meant for use by computer programmers, not general readers, and may use a modified Braille. Others require text to be input in a particular way. Finally, some produce output not suitable for the use by the Argonne Braille Machine. The machine itself imposes unique requirements on the material placed on the magnetic tape it reads. Not the least of these is the need for a continuous stream of Braille with no hyphenation. When the difficulties of modifying existing programs and the unique needs of the Argonne Machine were considered, it seemed appropriate for us to write our own translator.

Most people have heard of Grade I Braille, a one-to-one translation of inkprint characters into Braille. Its translation is completely straightforward and of no interest as a programming problem.

English Grade 2 Braille presents a difficult and interesting computer translation problem. The same sequence of letters may be translated differently depending upon position and function in the word or sentence and also upon pronunciation. To our knowledge the theoretical question of whether Braille can be translated perfectly by machine has not been addressed. We therefore decided to make the user the ultimate judge of the quality of the translation as determined by a field test. This field test is planned for 1974.

Design Philosophy

We had three objectives in mind as we designed the translation system. It was to be modular. It was to be easily modified, and it was to give as perfect Grade 2 Braille as possible. If it also turned out to be efficient and require minimal core storage, that was fine.

The modular approach makes it possible for one translator program to serve as a building block for any one of several different complete translation systems. For example, one might choose to have text read by an optical reader one day and have a secretary prepare copy on an MT/ST the next. The output generated by each different input module could go to the same translator module. Likewise one has the option of obtaining supplementary information along with the Braille translation depending upon the output modules chosen.

The modular approach was also helpful in meeting our second design goal - that the system be easily modified. This means that only the module needing changes has to be rewritten and recompiled. We decided not to work in machine language as a consequence of this design goal. Use of a higher level language makes the initial programming easier, and means that subsequent programmers can modify the program more easily. We chose PL/I as our computer language because of its character handling and list processing facilities.

The third criteria, that of having perfect Grade 2 Braille, was the most difficult. In a sense we will not know until the field test how well we have succeeded in eliminating all serious errors from the translation. We have consulted with blind Braille readers and a sighted Braillist. However, the final decision about whether a given rule should be fully implemented or not was up to me. I expect to be told that I have made some wrong decisions. My guiding maxim was: "if at all possible a perfect translation is to be produced". However, I had to balance this desire with the need to get the translator written.

Program Architecture

At present we have modules for three different types of input: the keypunch, the teletypesetter system used by Time-Life, Inc., and the compositor tape system used by Rocappi Computerized Corp. When using the keypunch module the person preparing the text for the keypuncher inserts supplementary characters as flags for the translator. These flags signal the need for special Braille characters to the translator program. The material is typed as written (except for the inserted supplementary characters) using 72 columns of the card. The last 8 columns may be used for identification and sequencing information. Excess blanks on the input cards are squeezed out by the translator. The two compositor tape systems take input from magnetic tape, insert the same supplementary characters and furthermore strip out, as far as possible, unwanted typesetting information.

Since there is a measure of human intervention in the use of the keypunch system, we do get the best translation here. The compositor tape systems cause many problems, some of which have been well documented by the American Printing House, and also the people at MIT. We have also done a study of the method. To summarize the difficulty briefly, the tapes used in the United States are far from clean. They contain a myriad of typographical errors. The publishers find it cheaper for workers to proofread the output from "dirty" compositor tape and correct the resulting plates by hand rather than to produce a perfect compositor tape. Further the tapes are often fragmented, poorly indexed, and carelessly handled by the publisher.

The output from each input module is given to the translator as a group of characters terminated by a blank. This is assumed to be a word. The output of the translator program consists of groups of 8 bits referred to as Braille equivalents. The first 6 bits of each group indicates the presence or absence of a raised dot in a specific matrix position. The seventh position is a parity bit which the Argonne Braille Machine uses for error checking. The eighth bit may be used to signal the presence of indexing information to the Braille Machine.

Dictionary Construction

When one is writing a translator, some means must be provided to translate groups of characters into Braille. One usually resorts to the use of a structure in the program called a "dictionary". This dictionary may be extremely large and contain all words which one plans to translate with the correct Braille translation for each word. On the other hand it may be very minimal, leaving much of the translation decision-making process to the program. In our system we have attempted to strike a compromise. Our dictionary consists of entire words, word fragments and single characters. If a word is difficult to translate by means of a simple decision process within the program, the entire word, with its appropriate Braille translation, is placed in the dictionary. The contractions of Braille are in the dictionary and their Braille equivalents. Also, parts of words containing Braille contractions may be placed in the dictionary. In this latter way one can often force a correct translation. Finally, we have single characters, such as the letter "r".

A dictionary entry contains the Braille translation of the letter or letters and also contains decision information in the form of a bit string made up of "ones" and "zeros". The word "knowledge" would have a "l" in position #5 meaning that the Braille contraction for "knowledge" can be used next to a hyphen. It would also have a "l" in position #6 indicating that the contraction can be used next to punctuation, and position #7 indicating that it can be used as a stand-alone word, with "0's" elsewhere in the decision string. However, the "rea" word fragment would have a "1" in position #2 indicating it can be used in the beginning of the word, in position #5 and #6 indicating it can be used next to a hyphen and punctuation and would have "0's" elsewhere. The single character "r" would have "1's" in position #1, #5, #6, and #7, the "1" in position #1 would indicate that the letter can be used anywhere in the word. The program sets appropriate flags to indicate that, for instance, the group of characters being translated occurs next to a space or is followed by punctuation. An interrogation of the bits in the bit string and also the flags set by the program must be done in order to determine whether or not a given contraction can be used in a particular place. Decision table techniques can be used here but have not been fully implemented yet.

Consider the following list of entries for a sample dictionary: "blank", "and", "with", and "wh". In principle, there will be a cell for each letter in each entry. Each cell will contain a pointer to the preceding cell, two pointers to the following cells, and a single alphanumeric character entry. If the cell also terminates a dictionary entry, it will also contain decision information and a Braille representation.

Using a binary tree structure to represent the above dictionary list, the circles representing each cell and arrows indicating pointers, one has a structure represented in Figure 1. Braille equivalents and decision information would be stored in the double circled cells. In this representation I use the convention that at any given decision point (or node) we branch to the right if the answer to the question is "yes", and we branch to the left if the answer is "no".

Let's see how this would work in a word. Let's assume that we want to find the Braille equivalent for the word "with". One would then start down the tree looking for the first character "w". At each character which we encounter which is not a "w" we branch to the "left". When we finally reach the "w" and we branch to the right. In this way we proceed past the blank "a", "b", etc, down to the "w". We then look at the next character in our word and find that it is an "i". When it is found the "t" and the "h" are sought, again going to the right, and located. At the final position, "h", a check is made for a Braille equivalent and decision information. If these aren't found or if the use of the Braille equivalent is not permitted, one would retrace his steps upward to the "t", "i" and "w", going upwards only until the Braille equivalent and appropriate decision information are found in the cell.

One further module has been incorporated into the basic translation package. This takes the digitized Braille output and retranslates it into inkprint characters and a representation of Braille output using the period to represent a raised dot. This module also enables us to obtain statistical information about the frequency of usage of the various contractions and is also very handy for proofreading purposes. Contractions are indicated by vertical placement of the letters of the contraction under the Braille output. This output is for the benefit of the sighted proofreader. In another version of the module, Braille is produced on the line printer in the form of raised dots for the benefit of the blind readers.

In this program we have used decision table techniques to determine the translation of each Braille character. The input to the retranslation step is the output from the Braille translator so therefore it consists of strings of eight binary digits. The first six of these represent the presence or absence of Braille dots. Each group of six digits is considered in the program to be a binary number and is used as a subscript in an array of 64 statement labels. In our program the labels are in an array called "labels", and the binary number is called "sub", so therefore the PL/I statement "Go To Labels (sub)" sends the execution process to the proper sections of the program in order to translate the Braille character and assemble the Braille cell. When an entire line of Braille representations and inkprint equivalents is filled up it is printed out.

At the time the retranslation is accomplished, an appropriate counter is incremented to indicate that a particular contraction or character has been used. At the end of the retranslation process the values of each counter can be printed out. This provides us with statistical data.

Results

We do not usually get 100 percent perfect Braille although we did on the second try with the book The Little Prince. We can approach a perfect translation asymptotically as the size of the dictionary (i.e., the number of words contained in it) approaches "infinity". Our approach has been pragmatic. If I come to a problem situation, I find as many words as I can that contain this combination of characters and then program it in such a way that more than half the time I get the correct translation. If the field test shows that there are serious errors produced in this way, we will put the problem words in the dictionary and force the correct translation if changing the program is not practical.

On the IBM system 360/50-75 complex which we were using, our Braille translation rate was approximately 22,000 Braille characters per minute. Since there are about 4.5 Braille characters per word, this is somewhat in excess of 4,000 words per minute - actually almost 5,000.

The field test will give us crucial information. We hope to gain enough information so that the various discussions about the need for proofreading of computer Braille can be settled. We also hope that we will have some hard data that can be used to influence future changes in the rules of English Braille. I do not believe in bending people to suit the computer. If, however, a rule cannot be handled by the computer so that a correct translation results and a form the computer _can_ handle is equally acceptable to the reader, I feel that the rule should be changed.

We feel the Argonne National Laboratory machine has a place in the world of the blind reader. We feel further that the entire system, including the use of the computer, envisioned by the Argonne National Laboratory team will make it possible for the blind individual to make a giant step toward the realization of his potential as a productive human being.

References

1) A. P. Grunwald, _Science, 154,_ 144 - 146, 1966

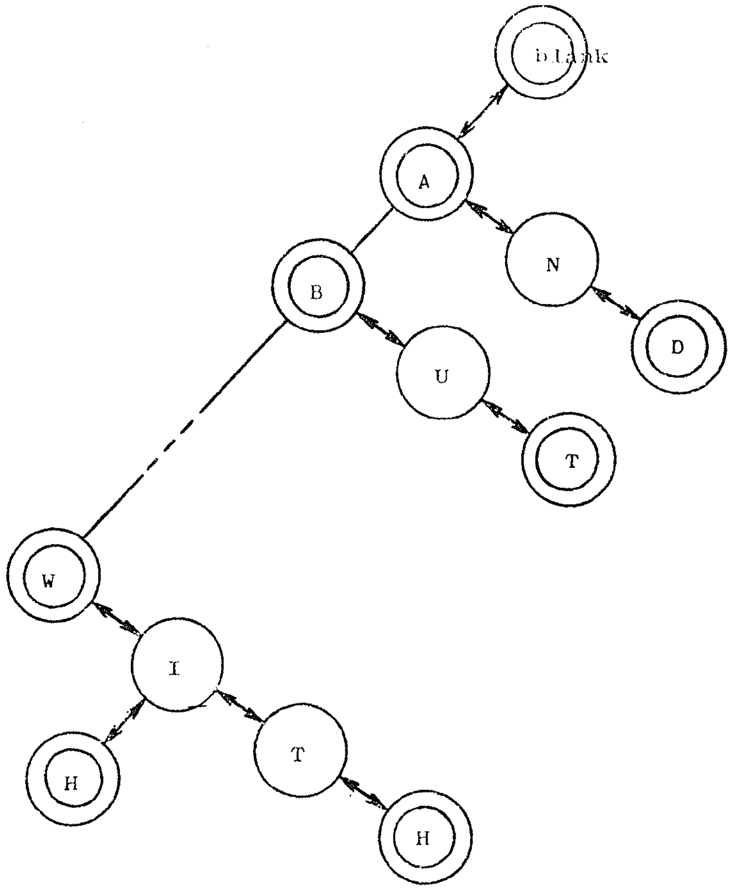

Figure 1

A SAMPLE DICTIONARY STRUCTURE