SOME REFLECTIONS ON THE CURRENT STATE OF AUTOMATIC BRAILLE TRANSLATION

by

Philip W. F. Coleman

Abstract

This paper outlines current work in Britain on Braille translation by computer. It goes on to explore possible programming methods, highlighting problems and suggesting solutions. It finishes by examining the feasibility of an international index of Braille software, suggesting a possible implementation.

Before I begin, may I say first, how pleased and privileged I feel to be here today - my pleasure will be all the greater if this conference achieves something really positive; and second, how grateful I am to Professor Werner for inviting me - I hope he will feel that my presence is worthwhile.

So much has been written already about Braille translation by computer that I feel doubtful about how much that is original there is for me to add. However, in this paper, I shall begin by outlining the work proceeding in the United Kingdom; I shall go on to suggest some programming techniques worth investigating when implementing Braille translation systems; and finally I shall talk about the possibility of the international pooling of Braille translation programs and of information about them.

The United Kingdom Scene

First, then, let me quickly sum up the current position of this work in the United Kingdom. Some of the most well-known work is that at the Royal National Institute for the Blind[1], which closely parallels that at the American Printing House for the Blind. Their suite of programs - for the IBM 1130 - takes as input punched cards containing text to be translated with embedded editing characters; proof copies are produced; there is provision for editing and correcting these as necessary; and the final output is of cards for their stereotyper. Currently, the design of the stereotyper is being modified by the Electrical Research Association of the UK, to include solid state control circuitry, and the ability to read tape input. The RNIB have looked at the possibility of using compositors' tape as input, but rejected it because of the inaccuracies inherent in the tapes available, and their small usage as yet in America and the UK.

Mr. John Gill, of the Department of Engineering Science, University of Warwick, has developed a program, written in machine-independent Fortran, which produces Braille somewhere between English Grades 1 and 2. He makes this available to anyone on request, but stresses that it has neither been properly debugged, nor is Braille tested for accuracy.

Mr. John White, Chairman of the British Computer Association of the Blind, has produced a procedure for use with the HASP option of the IBM System 360/370 Operating System: it enables the brailling not only of program output, but also of Job Scheduler messages[2].

To my knowledge, the only other programming effort in this field in Britain is my own. This work, outlined in Cleveland in 1969[3], was suspended in 1970 through pressure of work. However, I have now resumed work on it, and I hope to have a Standard English Braille (SEB, see footnote) program working by the end of this year, with the options of Grade 1, and British Computer Braille - I have had quite a demand for this program. Then will come the consolidation of my work on word frequency in technical literature, the formulation of grades of Braille oriented towards technical requirements and the PL/1 programming language, and the production of an enhanced Braille translation program including these grades. This latter program will be completely restructured to take advantage of modern programming techniques, and hardware refinements such as virtual storage.

System Considerations

With the increasing use of terminals, we should be more concerned with the problem of providing contracted Braille as an option for terminal users - particularly important when computer programmers may soon cease to be the main users of terminals. We have to capture, translate, and divert to the Braille terminal printer both outgoing and incoming signals, and the problem is complicated by the fact that we can do the translation in the main central processor to which the terminal is linked, or with a front end processor, or a processor integral with the terminal itself. The last is a special case of the second, and both approaches have been documented already [4, 5] - the former one for Grade 2 Braille.

At this point, I should like to touch upon very briefly a couple of considerations which arise in writing Braille translation programs. The first is whether to write a program around a definite Braille system (as I have done), with its grammar an integral part of the program; or whether the program should be table-driven (as in DOTSYS III), the Braille grammar being supplied as input. The second approach may be harder to implement, but gives the possibility of a common program for different Braille systems.

A technique that is attracting much attention by systems designers at present is Structured Programming[7], in which, speaking very roughly, a modular approach is taken to program design, each function giving rise to a module (not necessarily synonymous with "procedure" or "block"), each module having only one entry and one exit point. This greatly simplifies implementation, debugging, and program maintanance; it would also make easier the alteration of existing programs to fit different Braille systems, or new Braille output devices.

Another possible aid to the simplification of programming is worthy of study, I feel: the use of language preprocessors ("macro processors") in the definition of Braille grammars; the body of the program would be more or less invariable, only the macro definitions being changed for a new Braille system.

If the 64 characters of Braille are regarded as a machine language, it becomes possible to think of the input stream for translation into Braille as a high-level language, with the combinations which form contractions and abbreviations as keywords in the high-level language; the remainder of the text would have to be regarded as a peculiar kind of "noise", which is not ignored, but translated into equivalent Braille (machine language) as it stands. (Another of its peculiarities is its second role of word or contraction delimiter.)

This apparently odd way of regarding Braille might nonetheless give us another way of handling the generation of Braille translation programs - by the use of a compiler compiler technique. In a compiler compiler, we give the program as input the syntax of our high-level language, and a compiler for that input generating a particular machine language results. In the case of Braille, the language syntax input would be that of a particular Braille system, and the output a Braille translator for the Braille machine.

We have one additional complication: generally the compiler generates runs on the same machine as that of its object machine (the machine for which it generates output from the high-level language). In our case, the Braille translators may have to run on any number of different machines. Either we build a Braille translator generator for each machine required, or, at a more advanced level, details of the machine on which the translator is to run are given as additional input. Actually, a more rational solution is for the translator to be produced in a machine-independent high-level language, e.g., standard Cobol, Fortran, or PL/1.

Talking of standard languages prompts me to suggest that many problems might be solved if we were to produce formal definitions of Braille systems, as with any programming (or natural) language. This exercise would highlight the anomalous nature of the grammars of most of the Braille systems, and this itself might lead to a general tidying up process.

Where several alternative contractions could be used in translating a given letter configuration, parsing input strings in reverse ensures that that giving the fewest characters is used - e.g., in Standard English Braille, "THESE" would give "(THESE)", not "(THE)SE" or "(TH)ESE" - where parentheses surround a Braille contraction; it would also tend to give the correct translation of words like "WHEREVER" and "ADHERENT".

However, it is almost impossible to eliminate altogether the need for "special casing"; there is a very good case for simplifying the rules of all Braille systems, much as the Swedes and Dutch have done - if no-one else does so, I hope shortly to produce a draft Braille system using most of the present Grade 2 contractions, but with considerably simplified rules. One example would be the allowance of the symbol for "BE" unconditionally at the beginning of words (I am so used now to this simplification after using it in personal work for over two years, that I have to make a very positive effort to write correct SEB).

The sighted defenders of the old rules should be prepared to give us blind users credit for being able to use the intelligence we actually have: after all, we can distinguish printing errors of often a quite major kind, and British blind people read Standard English American Braille (SEAB), and doubtless Americans read SEB, without trouble.

Program Sharing

Now I should like to talk about sharing of programs and information about them on an international basis. There are two human factors which, unless recognized and counteracted individually and collectively, will continue effectively to bar all sharing, and will cause continued duplication of effort. The "not invented here" syndrome arises, I suspect, from pride: sometimes it is disguised as, "yes, that program is almost what I want, but not quite; so I really must write one to fulfill my requirements" - instead of building on work already done.

Personal reticence about disclosing work is the other factor, and I believe it stems from fear of public failure or late delivery of the goods. Since, as fallible humans, we are all liable to misjudge the time and resources required to do a job, we should not flinch from giving information about projects, including provisional completion dates. Let me try and establish the spirit of openness among us by admitting my own failure to deliver on time.

On a more positive plane, the British Computer Association of the Blind is currently drawing up an index of Braille translation programs throughout the world, containing all those of significance; but it is getting little cooperation from those approached who are outside of the UK, and would like to hear from anyone with details of such programs[6]. Information being collected includes: program name; machines and operating systems for which designed; type and grade of Braille; a brief description of the program; and the person or organization from whom it may be obtained, with their address. If this work succeeds, it could prove the basis for an international index of Braille software, which would include such additional information as: which country's Braille system was involved, and whether specialized Brailles like science or music could be produced. The BCAB itself is not in a position to undertake this, as two important criteria must be established before commencement of work on the index: the first is generous funding on a sound, continuing basis; and the second is good technical support by committed personnel with wide experience in both computing and Braille, able to be generous with the time they can give to this work, and to assess the programs they index.

Summary

In this paper, I have tried to give a picture of current and future developments in our field, as it appears to me at present. I think the image I should like to leave with you is something like this: a situation of great potential for cooperation and technical excellence, where little has yet been achieved; where more personal openness is required, and where ways may have to be found around the legal requirements which bind the evident good will of commercial concerns; a situation where little will be achieved without significant international funding, pooling of expertise, and good will.

I am doubtful to what extent we have the last of these requirements, let alone the others. May we go from here determined to achieve them all in some measure.

Acknowledgements

I should like to acknowledge the help of Mr. Norman Verrill, of the BCAB, who helped me clarify the idea of the international index of Braille software; and my company, IBM United Kingdom Laboratories Ltd., who provided me with the means of producing this paper and delivering it at Münster. I should like to absolve them from any controversy which may arise from the opinions in this paper, which are very much my own.

References

1) "Computer Produced Braille": Clive L. Windebank. Data Systems, November 1968, p. 18.

2) Available from: John N. White, CAV ltd., Computer Department, Hoath Lane, Gillingham, Kent, UK

3) "Man-Machine Communications for the Blind Programmer": Philip W. F. Coleman. Proceedings of "The Blind in Computer Programming"; ACM Newsletter for Blind Computer Programmers, July 1970, p. 75.

4) "A Real Time Braille Translation System": Fred Louis Lucont. MIT, 1965.

5) "An Inexpensive Braille Terminal Device": Anderson Rogers. Communications of the ACM, June 1971.

6) Details to: Richard West, Civil Service Department, Central Computer Agency, Norvic House, 29-33 Chapelfield Road, Norwich, UK

7) see e.g., "Chief Programmer Team Management of Production Programming": F. T. Baker. IBM Systems Journal,

Footnote - Explanatory Note on English Braille Systems

This note is intended for those computer personnel reading this paper who are not familiar with Braille, let alone the various Braille systems throughout the world. Without going into any detail, Braille is a tactile system consisting of characters based on a three-by-two dot matrix, giving 64 characters including the blank character (no dots). Broadly speaking the international Braille system recognizes letters, numerals and punctuation only, and then each country works out its own system of Braille, including the use of abbreviations and contractions (a contraction being where several characters are replaced by fewer).

In this paper, I mention explicitly three English Braille systems, all officially recognized: Standard English Braille (SEB), the official British system for literary Braille; Standard English American Braille (SEAB), the American equivalent, with identical symbology, but different rules for handling it; and British Computer Braille, which gives character-for-character representation of 56 standard computer characters, plus 8 non-specified characters which differ from system to system. This last system was developed for computer printout purposes in December 1970 at a meeting at the Royal National Institute for the Blind, at which were represented the British Uniform Type Committee, the British Computer Association of the Blind, and the three main groups of users in this country - IBM System/360, ICL 1900 and ICL System/4. The equivalent American system is Nemeth Computer Code, which is quite different in symbology, and does not always represent a character by one character in Braille.