A System for Converting Braille into Print

Paul Blenkhorn

Abstract— **This paper provides a detailed description of a method for converting braille, as it is stored as "characters" in a computer, into print. The system has been designed to be configurable for a wide range of languages and character sets, and uses a predominantly table driven method to achieve this. The algorithm is explained in the context of the conversion of Standard English Braille into print and the tables for this transformation are given.**

I. Introduction

An increasing number of computer-based systems are becoming available to assist people with disabilities. For blind people, such systems typically use a standard or braille keyboard for inputting into a computer, with speech or braille being used for output. These systems can vary from ones which have been specifically designed to meet a particular need, such as a transcription of blind children's examinations [1], "personal organizers" [2], or general purpose solutions which enable a user to access standard software on industry standard hardware [3]–[5]. In those systems which permit a blind person to enter information using a braille keyboard, there is a requirement to convert the braille codes into standard text. This conversion can take place directly onto paper [6], [7], as the information is spoken or printed [2], or at the point at which it is stored in the computer's memory [8]. Although such systems have been developed before, there has not been a detailed description published of how this can be achieved. The aim of this paper is to provide such a detailed description, together with the limitations of the method. Although this algorithm has been designed to deal with a large number of both languages and character sets, it is discussed here mainly in the context of the conversion of Standard English Braille into text.

This conversion utility is part of a more general system concerned with the translation of a wide range of codes used by disabled people, including the reverse translation to the one described here, i.e., text into braille. This work will be reported elsewhere.

II. The Braille System

The braille code was adapted by Louis Braille in the early part of the nineteenth century from a military system which used raised dots to send messages at night. After competition with other raised systems earlier this century [9], it has become the main system for the majority of those blind people who read and write using tactile means, and can be found in many countries around the world. Braille uses the raised dots in groups of six which are arranged in three rows of two, and which are numbered from 1 to 6, as can be seen in Fig. 1.

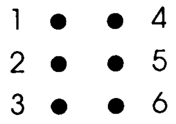

Fig. 1. A braille cell.

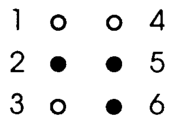

Fig. 2. A "context specific" braille cell.

These six positions, which can be raised or flat, are used in combination to give just 64 different braille "characters." This clearly means that there cannot be a one-to-one correspondence between braille characters and text. In the simplest commonly used form, called Grade 1 braille, the lower case letters A–Z and the major punctuation symbols are represented by a single braille character, with "shift" characters being used to indicate other information such as upper case, digits, and italics. For several reasons, including the size of braille characters (which are somewhat larger than normal text), the size and bulk of braille documents (which have to be embossed on rather thick paper), and the speed with which people can read information using touch [10], a number of countries have adopted a coding method, called Grade 2 braille or contracted braille. This further complicates the Grade 1 code by introducing, in a manner which is often specific to individual countries [11], context sensitive rules for the *contraction* of words and frequently used letter groups. These rules determine the correspondence between one or more braille cells and the print, so, for example, in Standard English Braille the braille symbol in Fig. 2 can stand for: "dis" when used at the start of a word (<u>dis</u>tance); "dd" when used in the middle of a word (la<u>dd</u>er); or a period when used at the end of a word (stop<u>.</u>).

Other rules can further complicate matters by insisting that the translation is not allowed across syllable boundaries. For example, "th" will be contracted in "<u>th</u>in," but not in "shor<u>th</u>and." In addition, a "letter sign" is used in braille to clarify when a single braille character represents a single print letter.

Manuscript received September 30, 1993; revised February 14, 1995.
The author is with the Department of Computation, University of Manchester Institute of Science and Technology, Manchester, M60 1QD England.
IEEE Log Number 9410561.

TABLE I
STATES, INPUT CLASSES, AND DECISION TABLE FOR STANDARD ENGLISH BRAILLE TO TEXT

The decision table has 6 states and 7 input classes.

The States are:

1 ... At the start of the word.

2 ... In punctuation at the start of the word.

3 ... After the start of the word.

4 ... Within a number.

5 ... Within members of the group "**&!(A)**".

6 ... Within the scope of a Letter Sign.

The Input classes are:

1 ... Don't care.

2 ... Valid at the start of the word.

3 ... Valid in punctuation, or at the start of the word.

4 ... Only valid after the start of the word.

5 ... Valid for members of the group "**&!(A)**".

6 ... Valid within the scope of a Letter Sign.

7 ... Valid within a number.

The Decision Table is:

```
1230000
1030000
1004000
1000007
1000500
0000060
```

TABLE II
RULE TABLE FOR STANDARD ENGLISH BRAILLE TO TEXT

```
1 [  ]=                    1
1 [!MVS]=themselves        3
3 [!]!: =the               5
5 [!]= the                 5
1 [!]=the                  3
3 ["1] ="                  1
1 ["D]=day                 3
1 ["E]=ever                3
1 ["F]=father              3
1 ["HAF]=hereafter         3
1 ["H]=here                3
1 ["K]=know                3
1 ["L]=lord                3
1 ["M]=mother              3
1 ["N]=name                3
1 ["OF]=oneself            3
1 ["O]=one                 3
1 ["P]=part                3
1 ["Q]=question            3
1 ["R]=right               3
1 ["S]=some                3
1 ["T]=time                3
1 ["U]=under               3
1 ["W]=work                3
1 ["Y]=young               3
1 ["!]=there               3
1 ["*]=character           3
1 ["?]=through             3
1 [":]=where               3
1 ["\]=ought               3
1 ["]="                    1
7 [#]=:                    4
6 [#]=                     4
4 [#]=ble                  3
1 [#']='                   4
1 [#]=                     4
6 [$]=ed                   3
1 [$]=ed                   3
1 [%D]=should              3
3 [%]: =shall              3
6 [%]=sh                   3
1 [%]=sh                   3
1 [&/OR]=and/or            3
3 [&]!: =and               5
5 [&]= and                 5
6 [&]=and                  3
1 [&]=and                  3
1 ['''0]: =..."            3
1 ['''8]: =...?            3
1 [''']=...                2
6 ['S]='s                  2
3 [']='                    2
7 ['S]='s                  2
7 [']: ='                   4
7 [']=,                    4
6 [']=,                    2
4 [']='                    3
3 [']='                    2
1 [']='                    2
3 [(]!: =of                5
5 [(]= of                  5
1 [(]=of                   3
3 [)]!: =with              5
5 [)]= with                5
1 [)]=with                 3
3 [*N]=children            3
3 [*]: =child              3
1 [*]=ch                   3
1 [+]=ing                  3
7 [,]=                     1
6 [,,]=<SHIFT_WORD>        6
6 [,]=<SHIFT_CHAR>         6
4 [,N]=ation               3
4 [,Y]=ally                3
6 [,8]='                   2
3 [,8]='                   2
3 [,7]=[                   1
7 [,G]: = grammes          3
3 [,G]: =grammes           3
3 [,G]#=grammes            3
7 [,M]: =metres            3
3 [,M]: =metres            3
3 [,M]#=metres             3
7 [,L]: =litres            3
3 [,L]: =litres            3
3 [,L]#=litres             3
1 [,,]=^^                  1
1 [,]=^                    1
7 [--]=-                   2
6 [--]=-                   2
1 [----]=----              3
1 -[-]=-                   2
1 [--]= --                 2
7 [-]=-                    4
6 [-]~: =-                 6
4 [-]=-                    2
3 [-]=com                  3
1 [-]=-                    2
4 [.D]=ound                3
4 [.E]=ance                3
4 [.N]=sion                3
4 [.S]=less                3
4 [.T]=ount                3
1 [.]=_                    1
```

The majority of the rules for Grade 2 English braille are the same around the world, however, several major differences do occur in their detailed application [12], [13]. The two main differences between British and American braille are concerned with capital letters and syllabification. The *capital sign*, which is the braille character with just dot 6 raised and which is used to indicate that characters are capital letters, is not commonly used in the U.K., whereas in North America it is normally used. Although both British and American braille have the rule whereby contractions should not be used across syllable boundaries, American braille is found to apply this rule more stringently. For example, the "of" contraction would be used in the word "professor" in the U.K., but not in North America. (This may have something to do with how words are pronounced in the "standard" form of English in the different countries.)

It should be noted that the Grade 1 and 2 codes are used for literary material. Additional codes have been developed for mathematical and scientific work, for music, and in specialized applications such as describing chess games.

III. COMPUTERIZED TRANSLATION OF BRAILLE

The earliest work on computerized translation of braille, reported in a number of conferences [14]–[17], was primarily concerned with the translation from text into braille so as to assist in the automatic production of braille books. Although some efforts have been directed toward the problems of translating mathematical and musical texts [18], [19], the majority of work in this field has dealt with the automatic translation of literary material into braille. However, even though some of the problems of translating literary material into braille, particularly those concerned with syllabification, placement of letter signs, and layout, have not been fully resolved there are now many working and effective systems available [20], [21].

Many of the earliest systems for braille production were pragmatic compromises of an algorithmic approach and the use of a dictionary [22]–[24]. They typically used a finite state machine to determine whether to translate a potential "window" of text into the corresponding braille characters subject to certain right contexts, such as whether the "window" was at the end or in the middle of the word. It is assumed that most systems still adopt this approach although this is now unclear since many have developed into commercial products and so detailed data on algorithms and data have become less readily available. Alternatives to this finite state machine approach have been investigated, particularly by Slaby [25], TABLE II *(Continued)*

7 [/]#=/	4	4 [6]=ff	3
7 [/]: =st	3	6 [6]=!	1
3 [/]: =still	3	1 [6]=!	1
6 [/;]=/	6	1 [7']=]	3
6 [/]=st	3	6 [7']=]	3
1 [/]=st	3	2 [7]_=were	3
4 [0']='	3	3 [7]=(	2
2 [0]_=was	3	1 [7]: =)	3
3 [0]=by	1	6 [7]: =)	3
6 [0]="	3	6 [7]=(	2
1 [0]: ="	3	4 [7]=gg	3
1 [0]="	1	1 [7]=(	1
7 [1] =,	4	1 [8''']="...	2
7 [1]=.	4	6 [8']='	3
1 [1]: =,	3	1 [8']: ='	3
6 [1]=,	3	2 [8]_=his	3
4 [1]=ea	3	3 [8]="	2
1 [1]=,	1	6 [8]: =?	3
3 [2C]: =because	3	1 [8]: =?	3
3 [2F]H=before	3	6 [8]="	1
3 [2F]: =before	3	1 [8]="	1
3 [2H]H=behind	3	6 [99]=*	1
3 [2H]: =behind	3	1 [99]=*	1
3 [2LL]: =belittle	3	3 [96]=into	1
3 [2L]: =below	3	6 [9]=in	3
3 [2N]: =beneath	3	1 [9]=in	3
3 [2SS]: =besides	3	3 [:]: =which	3
3 [2S]: =beside	3	6 [:]=wh	3
3 [2T]: =between	3	1 [:]=wh	3
3 [2Y]: =beyond	3	1 [;6]=+	1
3 [2]=be	3	1 [;_]=-	1
6 [2]=;	3	1 [;8]=*	1
1 [2]: =;	3	1 [;4]=/	1
4 [2]=bb	3	1 [;7]==	1
1 [2]=;	1	4 [;E]=ence	3
1 [3CVG]=conceiving	3	4 [;G]=ong	3
1 [3CV]=conceive	3	4 [;L]=ful	3
1 [3P#]=per cent	4	4 [;N]=tion	3
7 [3P]= per cent	4	4 [;S]=ness	3
1 [3P]=per cent	3	4 [;T]=ment	3
1 [3#]=:	4	4 [;Y]=ity	3
6 [3]=:	3	6 [;]=	6
1 [3]: =:	3	1 [;]=	6
4 [3]=cc	3	6 [<]=gh	3
3 [3]=con	3	1 [<]=gh	3
1 [3]=:	1	6 [>]=ar	3
3 [4#]=dollars	4	1 [>]=ar	3
7 [4] =.	3	1 [?YF]=thyself	3
6 [4]=.	6	3 [?]: =this	3
1 [4]: =.	3	6 [?]=th	3
4 [4]=dd	3	1 [?]=th	3
3 [4]=dis	3	6 [@]=`	1
1 [4]=.	1	1 [@]=`	1
3 [5]: =enough	3	7 [A]=1	4
6 [5]=en	3	6 [A]=a	6
1 [5]=en	3	3 [ABV]=above	3
3 [6]=to	1	3 [AB]: =about	3
1 [6]: =!	3	1 [ACLY]=accordingly	3

TABLE II *(Continued)*

3 [AC]: =according	3	7 [E]=5	4
3 [ACR]: =across	3	6 [E]=e	6
3 [AF]B=after	3	3 [EI]: =either	3
3 [AF]G=after	3	3 [EX#]=ex	4
3 [AF-]=1after-	1	7 [EX]: = example	3
3 [AFN]=afternoon	3	3 [EX]-=ex	3
3 [AFW]=afterward	3	3 [EX]: =example	3
3 [AF]?=after	3	7 [EXS]: = examples	3
3 [AF]M=after	3	3 [EXS]: =examples	3
3 [AF]D=after	3	1 [E4G4]=e.g.	3
3 [AF]: =after	3	3 [E]: =every	3
3 [AG/]=against	3	1 [E]=e	3
3 [AG]: =again	3	7 [F]=6	4
3 [ALM]: =almost	3	6 [F]=f	6
3 [ALR]: =already	3	3 [F/]=first	3
3 [AL]: =also	3	1 [FRS]=friends	3
3 [AL?]: =although	3	1 [FR]L=friend	3
3 [ALT]: =altogether	3	1 [FR]: =friend	3
3 [ALW]: =always	3	3 [FT#]=feet	4
1 [A4M4]=a.m.	3	7 [FT]: = feet	3
3 [A]!: =a	5	3 [FT]: =feet	3
5 [A]= a	5	3 [F#]=francs	4
1 [A]=a	3	3 [F]: =from	3
7 [B]=2	4	1 [F]=f	3
6 [B]=b	6	7 [G]=7	4
3 [BLLY]: =blindly	3	6 [G]=g	6
3 [BL]F=blind	3	3 [GD]=good	3
3 [BL;S]: =blindness	3	1 [GRT]=great	3
1 [BL]M=blind	3	3 [GL#]=gallons	4
3 [BL]: =blind	3	7 [GL]: = gallons	3
1 [BRL]=braille	3	3 [GL]: =gallons	3
3 [B]: =but	3	3 [G#]=guineas	4
1 [B]=b	3	3 [G]: =go	3
7 [C]=3	4	1 [G]=g	3
6 [C]=c	6	7 [H]=8	4
1 [C/O]=c/o	3	6 [H]=h	6
3 [CW#]=hundredweight	4	3 [H]F]=herself	3
7 [CW]: = hundredweight	3	3 [HMF]=himself	3
3 [CW]: =hundredweight	3	3 [HMM]=hmm	3
3 [CD]=could	3	3 [HM]=him	3
3 [C#]=cents	4	3 [HR#]=hours	4
3 [C]: =can	3	3 [HR]: =hours	3
1 [C]=c	3	3 [H]: =have	3
7 [D]=4	4	1 [H]=h	3
6 [D]=d	6	7 [I]=9	4
1 [DCVG]=deceiving	3	6 [I]=i	6
1 [DCV]=deceive	3	1 [IMM;S]=immediateness	3
1 [DCLG]=declaring	3	1 [IMMLY]=immediately	3
3 [DCL]=declare	3	3 [IMM] =immediate	3
3 [DM#]=dm (Deutch)	4	1 [I4E4]=i.e.	3
3 [DM]: =dm (Deutch)	3	3 [I#]=inches	4
7 [DM]: = dm (Deutch)	3	1 [I]=i	3
3 [DG#]=degrees	4	7 [J]=0	4
7 [DG]: = degrees	3	6 [J]=j	6
3 [DG]: =degrees	3	3 [J]: =just	3
3 [D#]=pence	4	1 [J]=j	3
3 [D]: =do	3	6 [K]=k	6
1 [D]=d	3		

whose segment translation system operates by considering left and right contexts. He argues that the other approaches lead to systems which are very difficult to adapt and update due to the complications of state tables, control tables, and rules.

In addition to the above efforts in converting from text into braille, there have been a number of systems which perform the translation from braille into print [7], [21], [26], although the strategies and algorithms adopted here are somewhat less well-documented than for the early print to braille translators. Again, many of these algorithms have become embedded in a number of products including stand alone braille note-takers (e.g., see [2] and [8]).

Several problems can be identified in the conversion of braille into print, particularly to do with the context specific nature of braille. For example:

- Unit abbreviations (e.g., 20 m, 44 yds, etc.) can be ambiguous, although this situation has improved slightly since the (U.K.) English rules changed and the abbreviations no longer appear before numbers in braille.
- Initials in names (e.g., Mr. K. Smith) are written without a letter sign.
- The same braille sign is used for the oblique stroke (/) and the letter group "st."

IV. This System

The general purpose system, of which this braille to text system is a special case, has been developed to operate with a finite number of states which can hold the current context as well as capabilities for both left and right context matching. In the application of this system to the conversion of print into braille (which will be reported elsewhere), the approach taken is closer to Slaby's [26] of using left and right contexts than that of a state system. However, this system, which converts from braille into print, uses the finite state system to hold the current context with right context checking being achieved by using matching algorithms. The reason for using the finite state approach for the "left context" was that it is no simple matter to do a character/wildcard matching operation to determine

TABLE II *(Continued)*

3 [KC/S#]=kilocycles per second	4
7 [KC/S]: = kilocycles per second	3
3 [KC/S]: =kilocycles per second	3
1 [KC#]=kilocycles	4
7 [KC]: = kilocycles	3
3 [KC]: =kilocycles	3
3 [KW#]=kilowatts	4
7 [KW]: = kilowatts	3
3 [KW]: =kilowatts	3
3 [K]: =knowledge	3
1 [K]=k	3
3 [L#]=£	4
6 [L]=l	6
3 [LR]=letter	3
3 [LL]A=ll	3
3 [LL]E=ll	3
3 [LL]I=ll	3
3 [LL]O=ll	3
3 [LL]U=ll	3
3 [LL]=little	3
3 [LB#]=pounds	4
7 [LB]: = pounds	3
3 [LB]: =pounds	3
3 [L]: =like	3
1 [L]=l	3
6 [M]=m	6
1 [M*]=much	3
1 [M/]=must	3
1 [MYF]=myself	3
3 [MN#]=minutes	4
3 [MN]: =minutes	3
3 [MC/S#]=megacycles per second	4
7 [MC/S]: = megacycles per second	3
3 [MC/S]: =megacycles per second	3
3 [MC#]=megacycles	3
7 [MC]: = megacycles	3
3 [MC]: =megacycles	3
3 [M#]=miles	4
3 [M]: =more	3
1 [M]=m	3
6 [N]=n	6
1 [NEC]: =necessary	3
3 [NEI]: =neither	3
1 [NEWSLR]=newsletter	3
3 [N]: =not	3
1 [N]=n	3
6 [O]=o	6
3 [OZ#]=ounces	4
7 [OZ]: = ounces	3
3 [OZ]: =ounces	3
3 [O'C]: =o'clock	3
1 [O]=o	3
6 [P]=p	6
1 [PD]: =paid	3
1 [P}CVG]=perceiving	3
1 [P}CV]=perceive	3
1 [P}H]=perhaps	3
3 [PT#]=pt	4
3 [PT]: =pt	3
3 [P>#]=paragraph	4
7 [P>]: = paragraph	3
3 [P>]: =paragraph	3
1 [P4M4]=p.m.	3
3 [P#]=p.	4
3 [P]: =people	3
1 [P]=p	3
6 [Q]=q	6
1 [QT#]=quarts	4
7 [QT]: = quarts	3
3 [QT]: =quarts	3
1 [QR#]=quarters	4
7 [QR]: = quarters	3
3 [QR]: =quarters	3
1 [QK]=quick	3
3 [Q]: =quite	3
1 [Q]=q	3
6 [R]=r	6
1 [R4I4P4]=r.i.p.	3
1 [RCVG]=receiving	3
1 [RCV]=receive	3
1 [RJCG]=rejoicing	3
1 [RJC]=rejoice	3
3 [R#]=rupees	4
3 [R]: =rather	3
1 [R]=r	3
6 [S]=s	6
1 [SD]: =said	3
1 [S*]: =such	3
3 [ST#]=stones	4
7 [ST]: = stones	3
3 [SE#]=seconds	4
3 [SE]: =seconds	3
3 [S#]=shillings	4
3 [S'#]=section	4
3 [S]: =so	3
1 [S]=s	3
6 [T]=t	6
3 [TD]=today	3
3 [TGR]=together	3
3 [TM]=tomorrow	3
3 [TN]=tonight	3
3 [T#]=tons	4
3 [T]: =that	3
1 [T]=t	3
3 [U4K4]=U.K.	6
6 [U]=u	6
3 [U]: =us	3
1 [U]=u	3
6 [V]=v	6
3 [V]: =very	3

TABLE II *(Continued)*

1 [V]=v	3
6 [W]=w	6
3 [WD]=would	3
3 [W]: =will	3
1 [W]=w	3
6 [X]=x	6
3 [XS]: =its	3
1 [XF]=itself	3
3 [X]: =it	3
1 [X]=x	3
6 [Y]=y	6
1 [YRF]=yourself	3
1 [YRVS]=yourselves	3
3 [YR]=your	3
3 [YD#]=yards	4
7 [YD]: = yards	3
3 [YD]: =yards	3
3 [Y]: =you	3
1 [Y]=y	3
6 [Z]=z	6
3 [Z]: =as	3
1 [Z]=z	3
1 [\RVS]=ourselves	3
3 [\]: =out	3
6 [\]=ou	3
1 [\]=ou	3
1 [^U]=upon	3
1 [^W]=word	3
1 [^!]=these	3
1 [^?]=those	3
1 [^:]=whose	3
1 [^]=	1
1 [_C]=cannot	3
1 [_H]=had	3
1 [_M]=many	3
1 [_S]=spirit	3
1 [_W]=world	3
1 [_!]=their	3
6 [_]=_	1
1 [_]=_	1
1 [{O]=•	3
6 [{]=ow	3
1 [{]=ow	3
3 [|]!: =for	5
5 [|]= for	5
6 [|]=for	3
1 [|]=for	3
6 [}]=er	3
1 [}]=er	3

whether the previous character is, for example, a letter group, punctuation character, etc. (Whereas in print the character "." is always a period, the braille character used for a period could represent a number of possibilities.)

One of the aims of this system was to provide the facility whereby braille to print conversion could be achieved for a wide range of languages [11], and although there is some uncertainty as to whether the system will cope with the complications of German braille as described by Seiller and Oberleitner [26], it is anticipated that this will be achieved for a considerable number of languages. As a result, the system has been designed so that a wide range of options and data can be input using a set of tables, including braille rules, which are presented in a clear manner. It is intended that this will ease the difficulties associated with the enhancement and updating of existing tables for particular languages, and will facilitate the production of translators for new languages.

The algorithm used for the conversion can be found in the Appendix. Table I shows the decision table used for the conversion of Standard English Braille.

Note: A nonzero value in the decision table indicates that a rule should "fire" for a given input class and current_state. A value of zero indicates that the rule should not "fire."

V. Results

The rules for the transcription of Standard English Braille are listed in Table II.

Notes:

- The format of the rules in Table II is: Input class ⟨TAB⟩ rule ⟨TAB⟩ new_state

  The input class is set for each rule and is used in conjunction with the decision table to determine if a rule "fires."

  The rule is in the format: [focus]right_context = output_text

  Several wildcards can be used in the right_context. These are:

"!"	··· one or more of the set "&!(A)."
" "	··· any white space character.
"~"	··· one or more roman letters (e.g. I, IV, MCM).
":"	··· zero or more potential punctuation characters.
"_"	··· actual space character.

- Any characters which are not in the tables go through the system and result in the new_state being set to 1.
- The character "^" is used in the output of this system to indicate that a character should be converted to uppercase and that when used twice the " ^ " indicates that the following word should be converted to upper case.
- This algorithm should work for the braille codes used in the U.K. and North America as the differences between the British and American codes (for example, the two ways to contract the "of" in professor) will both result in the same print output from this system.
- The system used to represent the braille characters in ASCII format is American Computer Braille.

To illustrate how these rules work, the word "adds" is considered. The braille equivalent is shown in Fig. 3. In American Computer Braille the word is: A4S.

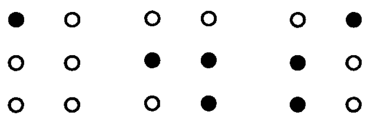

Fig. 3. The braille words "adds."

The main points involved in its translation are now detailed:

The system will search through the tables starting with the entry: 7 [A]=1 4. The focus matches for [A] and so the state is checked for input_class 7 and current_state 1. The decision table has a 0 and so the system goes to the next rule: 6 [A]=a 6.

Similarly, the decision table has 0. The next rule is: 3 [ABV]=above 3. Here, the focus (i.e., ABV) does not match and so the rule does not fire. The system continues to the rule 1 [A]=a 3. This matches and the decision table entry for input_class 1 and current_state 1 is 1. The right_context of the rule is blank and so the rule fires, the right-hand side of the rule (i.e., "a") is output, the current_state is set to 3 (i.e., after the start of the word) and the system moves one character along the braille word, leaving "4S."

The system now searches starting with the entry: 3 [4#]=dollars 4. The focus does not match. The next rule is: 7 [4]= . 4. The decision table entry is 0 for the input_class 7 and current_state 3. The next rule (6 [4]= . 3) also fails to satisfy the decision table. The next rule is: 1 [4]: = . 3. The focus matches, the decision table for input_class 1 and current_state 3 is 1, and so the right context is checked. The right context of the rule is ":_" which can be interpreted as looking for a space (i.e., the end of the word) after zero or more characters which are "potential punctuation characters." The input buffer has an "S" to the right of the "4." "S" is not a potential punctuation character and so the rule fails. The next rule is: 4 [4]=dd 3. The focus clearly matches and the decision table for input_class 4 and current_state 3 is 4, and so the right context is checked. The right context of the rule is blank and so the rule fires, the right-hand side of the rule (i.e., "dd") is output, the current_state is set to 3 (i.e., after the start of the word), and the system moves one character along the braille word, leaving "S."

Similarly the rule: 1 [S]=s 3 will fire for "S" giving the total output "adds," as required.

The system detailed here has been tested on a set of Standard English Braille words which were designed to test all of the rules found in the Braille Primer [27]. In addition, extensive tests have been carried out with braille files which were produced by Torch Trust for the Blind.

Overall the system performs well on contracted English Braille. However, there are a number of problems which have been identified:

- The poetry sign is not dealt with in a satisfactory manner. It is currently converted as the contraction "ar" which is represented by the same braille character.
- Specialist biblical references containing chapters of the bible and verse numbers are not converted correctly. (However, this is simply a matter of adding the biblical names to the main tables for the system.)
- Some of the old style unit abbreviations are not converted correctly.
- Punctuation within words can cause problems. For example, the system cannot distinguish between the braille equivalents of "adds." and "a.s." and would convert both into "adds."
- The oblique stroke "/" is not always converted properly when it occurs in the middle of a word.
- Isolated letters without letter signs (for example, the braille equivalent of "Mr. M. Jones") are not converted correctly.

VI. Concluding Remarks

Although the system described here performs quite well, it is intended to test and develop the system further by attempting to produce further tables for the conversion of braille into print for languages other than English.

Appendix
The Conversion Algorithm

The algorithm is described below using Structured English.

```
program convert
  begin
  do
      read_word
      convert word into normal form   // use table to convert lower to upper case.
                                      // tidy up graphics characters etc.
      convert_braille_into_print
  while not end_of_input
  end  // of main program

procedure convert_braille_into_print
  begin                               // turn braille word into print
  set current_state to 1
  set current_character to first character in word
  while still converting do           // do the whole word
      begin
      set match to FALSE              // initialize for the loop
      start search in rule table at rule defined by current_character
```

```
        repeat
            if focus_matches and state_ok and right_context_ok
             and left_context_ok then // left context not used for braille to text
                begin
                output right-hand side of rule  // i.e., the text after the equals sign
                set current_state to new_state  // get new state from end of the rule
                move along word by size of current rule focus
                set match to TRUE
            end
            else go to next rule
            if not match
               and new rule does not start with same letter as current_character then
                  begin                         // no more rules for that character
                  output current_character      // so use default option
                  set current_state to 1        // and output braille character
                  set match to TRUE
                  end
        until match       // keep going round until done current character
        set current_character to first character in word
        end               // while still converting—keep going until done whole word
    end // of convert braille into print

function focus_matches
  begin
  set match to TRUE
  set input_index to index into input_buffer position for current_character
  set rule_index to index start of focus for rule
  do
      if input_buffer[input_index] != rule[rule_index] then    // not got a match
          set match to FALSE
      increment rule_index                       // move along rule
      increment input_index                      // move along input
  while match and (rule[rule_index] != "]")      // Note: "]" terminates focus
  return match
  end // of focus_matches

function state_ok
  begin   // nonzero entry fires state
  if decision_table[input_class of current rule, current_state] > 0 then
      return FALSE
  else
      return TRUE
  end // of state_ok

function right_context_ok
  begin
  set match to TRUE
  increment input_index                                     // step over "]"
  do
      if rule[rule_index] is a wildcard then         // "!," " ", "~," ":," or "-"
          begin
          if not valid_wildcard_match then
                                             // Note: this will move along input buffer
              set match to FALSE             // and increment input_index appropriately
          else do wildcard match
          end
      else
          begin
          if input_buffer[input_index] != rule[rule_index] then // not got a match
              set match to FALSE
          increment input_index                          // move along rule
          end
      increment rule_index                               // move along input
  while match and (rule[rule_index] != TAB)    // Note: TAB terminates
                                               // right hand context of rule
  return match
  end // of right_context_ok
```

ACKNOWLEDGMENT

Thanks are due to Torch Trust for the Blind who have helped in providing some of the data files for testing this system.

REFERENCES

[1] P. L. Blenkhorn, "Personal transcription systems," in *Computerised Braille Production. The Proceedings of the Fifth International Workshop, Winterthur October 30–November 1, 1985*, J. M. Ebersold, T. Schwyter, and W. A. Slaby, Eds. Eichstatt: Katholische Univ., 1986, pp. 33–38.

[2] D. Blazie, "Braille 'N' speak: A story, an update," *Braille Monitor*, pp. 174–176, Apr. 1988.

[3] J. D. Leventhal, E. M. Schreier, and M. M. Uslan, "Electronic braille displays for personal computers," *J. Visual Impairment Blindness*, vol. 84, pp. 423–427, 1990.

[4] A. Meyers and E. Schreier, "An evaluation of speech access programs," *J. Visual Impairment Blindness*, vol. 84, pp. 26–38, 1990.

[5] N. Davies, "Computers at work: Making computers work for you," *New Beacon*, vol. LXXV(887), pp. 202–204, 1991.

[6] A. T. Vincent and S. Smith, "A talking brailler," in *Learning to Cope*. London: Educational Computing, 1983, pp. 33, 35.

[7] J. Spragg, "Interfacing a Perkins brailler to a BBC micro," *Microprocessors Microsyst.*, vol 8, pp. 524–527, 1984.

[8] M. Priwler, "Eureka A4: The arrival of a new concept," *Braille Monitor*, pp. 164–167, Apr. 1988.

[9] W. Bledsoe, "Braille: 'A success story,' " in *Evaluation of Sensory Aids for the Visually Handicapped*. Washington, D.C.: National Academy of Sciences, 1972, pp. 3–36.

[10] E. Foulke, "Reading braille," in *Tactual Perception: A Sourcebook*, W. Schiff and E. Foulke, Eds. Cambridge: Cambridge, 1982, pp. 168–208.

[11] *World Braille Usage*. UNESCO, Paris: The National Library Service for the Blind and Physically Handicapped. Washington, D.C.: Library of Congress, 1990.

[12] British National Uniform Type Committee, *A Restatement of the Layout, Definitions and Rules of the Standard English Braille System, Issued by the British National Uniform Type Committee, 1952*. London: Royal National Institute for the Blind, 1955. Revised 1968, published 1969.

[13] American Association of Workers for the Blind and Association for the Education of the Visually Handicapped, *English Braille, American Edition 1959*. Louisville, KY: American Printing House for the Blind, 1970. Revised 1962, 1966, 1968, 1970.

[14] R. A. J. Gildea, G. Hubner, and H. Werner, Eds., *Computerized Braille Production: Proceedings of the First International Workshop in Munster (Germany), March 1973*. Munster: Rechenzentrum Univ. Munster, Dec. 1974.

[15] H. Werner, Ed., *Computerized Braille Production: Proceedings of the Second International Workshop in Copenhagen (Denmark), September 1974*. Munster: Rechenzentrum Univ. Munster, June 1978.

[16] J. M. Ebersold, T. Schwyter, and W. A. Slaby, Eds., *Computerised Braille Production. Proceedings of the Fifth International Workshop, Winterthur October 30–November 1, 1985*. Eichstatt: Katholische Univ., 1986.

[17] G. Francois and J. Engelen, Eds., *Computerised Braille Production. Proceedings of the Sixth International Workshop on Computer Applications for the Visually Handicapped, September 19–21, 1990*. Leuven, Belgium, 1990.

[18] T. Wesley and J. Wallace, "The application of information technology to the access of mathematical information for the blind," in *Proc. 3rd Int. Conf. Comput. Handicapped People*, Vienna, pp. 562–569, July 7–9, 1992.

[19] J. B. Humphries, "Computerised braille music production using CIMBAL," *Braille Res. Newslett.*, no. 10, pp. 6–15, 1979.

[20] A. M. Goldberg, E. M. Schreier, J. D. Leventhal, and J. C. De Witt, "An evaluation of braille translation programs," *J. Visual Impairment Blindness*, vol. 81, pp. 487–492, 1987.

[21] "Braille and computers," in *Aids and Appliances Review*, no. 11. The Carroll Centre for the Blind, Winter 1984.

[22] E. Sullivan, "Braille translation," in *Uses of Computers in Aiding the Disabled*, J. Raviv, Ed. Amsterdam: North Holland, 1982, pp. 351–366.

[23] H. Werner, "Automatic braille production by means of computer," in *Uses of Computers in Aiding the Disabled*, J. Raviv, Ed. Amsterdam: North Holland, 1982, pp. 321–336.

[24] P. A. Fortier, D. Keeping, and D. R. Young, "Braille: A bilingual (French/English) system for computer aided braille translation," in *Research Report 2*. Winnipeg: Univ. Manitoba, 1977.

[25] W. A. Slaby, "Computerised braille translation," *J. Microcomput. Applicat.*, vol. 13, pp. 107–113, 1990.

[26] F. P. Seiler and W. Oberleitner, "WineTU: German language grade 2 to ASCII braille translator," *J. Microcomput. Applicat.*, vol. 13, pp. 185–191, 1990.

[27] Royal National Institute for the Blind, *Braille Primer with Exercises, Based on the Restatement of Standard English Braille*. London: RNIB, 1969.

Paul Blenkhorn was as a Research Fellow at the Research Centre for the Education of the Visually Handicapped, Birmingham University (England).

He has been involved in assistive technology for a little over 10 years. He was one of the founders of Dolphin Systems for People with Disabilities Ltd., which developed, manufactured, and marketed a range of assistive devices for disabled people. For the past three and a half years, he has been a Lecturer in the Department of Computation at UMIST and a member of the Technology for Disabled People Unit there. His research interests remain in the application of technology to meet some of the needs of disabled people.