

# Chapter 3

## The mathematical Woluwe code

### 3.1 Overview of the Woluwe code

This section provides an overview of the conversion process of Braille mathematics into some other readable form.

The Woluwe code examples provided in this overview are based on the Woluwe rules for mathematical Braille.

#### 3.1.1 Short description of Braille

Braille is a system of touch reading for the blind which employs embossed dots evenly arranged in quadrangular letter spaces or cells. In each cell, it is possible to place six dots, three high and two wide. By selecting one or several dots in combination, 63 different characters can be formed.

To aid in describing the characters by their dots, the six dots of the cell are numbered 1, 2, 3 counting downwards on the left side, and 4, 5, 6 counting downwards on the right side, as illustrated in Figure 3.1.

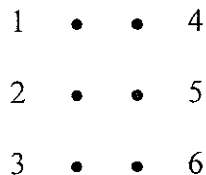


Figure 3.1: dots numbering for a Braille cell

#### 3.1.2 Braille in textual mode

Before discussing the mathematical Braille, we will first take a look at the way in which Braille is used to represent 'normal' text.

As stated before, the use of 6 dots allows us to represent 63 characters. Counting the number of characters available for normal writing, we rapidly conclude that 63 possible combinations are not enough to represent all characters (counting the numbers 0 up to 9, and the letters a,A up to z,Z, we already have 62 characters).

In order to increase the number of characters that can be represented with the six dots, two techniques are being used:

- implementing the use of special Braille characters, called prefixes or composition signs
- making the meaning a Braille character dependent on the context in which it occurs

### 3.1.2.1 Prefixes or composition signs

Composition signs are placed in front of a Braille cell in order to be able to provide some additional information on how the cell contents is to be interpreted.

Examples of some of these composition signs are:

- The number prefix is being used to represent the numbers:

⠠

EuroBraille: #

*Example:* the following combination is used to represent the number 1:

1

⠠⠠

EuroBraille: # a

- The capital-prefix use to help to represent capitals:

⠠

EuroBraille: \$

*Example:* the following is used to represent the letter B:

B

⠠⠠

EuroBraille: \$ b

### 3.1.2.2 Context dependency

Besides implementing the use of prefixes, described in the previous part, the meaning of a Braille-character can also depend on the context in which it occurs.

*Example:* There is only one Braille character (dots 2, 3, 5 and 6) available for both the open- and close round brackets ‘(‘ and ‘)’. So the context in which the Braille character occurs will determine if it should be interpreted as a ‘(‘ or a ‘)’.

### 3.1.3 Braille mathematics

The representation of mathematical information in braille is a lot more difficult than the representation of normal-text.

Some of the problems are:

- the use of special characters like  $\int$ ,  $\sum$ ,  $\in$ ,  $\subseteq$ , ...
- the use of 'formatting information' to represent subscript- and superscript characters like:  $a_1$  and  $x^2$
- making sure that formulas are correctly interpreted by automatically adding brackets



#### 3.1.3.1 Mathematical characters

A whole list of definitions were made enabling the representation of all possible mathematical characters. For most of these characters, a series of braille-cells are needed. On the following pages, you will find a brief list of the most common mathematical characters and their braille-representation.

##### *Greek Characters*

**Dots 5 and 6** are used to indicate that the information that is about to follow should be interpreted as small Greek letters. Dots 4, 5 and 6 are used for capital Greek letters.


##### *Examples*

$\pi$	 < p
$\Pi$	 8 p

##### *Characters indicating a set of numbers*

Characters which represent a set of number belonging to a specific group always start with dot 4, followed by the letter identifying the set.



##### *Example*

the set N	 " n
-----------	--

*Other special characters*

Other special character that can be formed are:  $\forall$ ,  $\exists$ , **T**, **L**, **H**,  $\Re$ . All of these characters start with dots 1, 2, 3, 4, 6. Except for the sign  $\infty$ , which starts with dots 3, 4, 5, 6.


*Examples*

character $\forall$	 & !
character $\exists$	 & ?

The other characters (T, L, H, R) are represented by a cell containing dots 1, 2, 3, 4, 6 followed by the dot-combination for the letter itself (example: T = dots 2, 3, 4, 5).

One exception for these special characters is  $\infty$ , it does not start with dots 1, 2, 3, 4, 6 but with dots 3, 4, 5, 6 (which is the same combination used for the number-prefix!).


*Example*

character $\infty$	 # %
-----------------------	--

3.1.3.2 The punctuation

A punctuation character is always followed by a space (except for the opening-” and ‘( which should start with a space).

An example of this exception is the term  $z(\cos \alpha, \sin \alpha)$  :

$z(\cos \alpha, \sin \alpha)$	 z 2 c o s < a , s i n < a '
-------------------------------	---

In this example you can see that a space was added before the cell with the dots 2, 3 and 4.





Another exception is the use of punctuation within the definition of a number value (example: 3.462,25). In this case no space should be put at the end of each punctuation character.

*Remark* Punctuation characters are only to be added when they have a significant meaning within the term.

### 3.1.3.3 Arithmetic and relational operators





Most of the arithmetic and relational operators are preceded by a space or dot 5. Some examples of these symbols are:




*Examples*

+	 ! +
-	 ! -
*	 ! )
<	 ! 9

Exceptions on this rule are the symbols  $\circ$ ,  $>$ ,  $\geq$ ,  $\in$ ,  $\varepsilon$ ,  $\uparrow$  and  $\downarrow$ . These exceptions are listed below:


*Examples*

$\circ$	 ' )
$>$	 " o
$\geq$	 " o =
$\in$	 " e

∃	 " i
↑	 " 3
↓	 " 0


### 3.1.3.5 Crossed-out symbols

In normal writing, symbols which are ‘crossed-out’ normally indicate a negation. In Braille, this negation can be accomplished by putting the following Braille codes in front of the character:

 ! *
--

*Remark:* The dot 5 in front of the actual negation-sign may be replaced by a space.



An example for the symbol  $\neq$  is:







$\neq$	 ! * =
--------	--

### 3.1.3.6 The use of brackets

First of all we will take a look at the different types of brackets and their corresponding Braille codes. In the examples given, the first term is the one to open the bracket, the other one is used to close it.

Brackets:


(	 2
)	 ~

[	 {
]	 }
{	 ! {
}	 ! }
	 4
	 4 4


### 3.1.3.7 Two dimensional constructs

These are the notations for the structures which use several lines and may be surrounded by parentheses, such as matrices, determinants and sets of equations.

- Enter the Braille code for the bracket to be opened, followed by a full sign (all dots raised).
- When you want to start a new line in normal writing, you must insert the following codes.

	 < 8
--	--

- To end the formula or mathematical term, you must enter the following code, followed by the code used to close the bracket.


	 5
--	--

### 3.1.3.8 Fractions

Basic rules for constructing fractions containing only numeric values:

- the number values for the denominator should be brailled with all dots lowered one row


*Example*

7	 # g
---	--

should be brailled like this when it is part of the denominator of a fraction


- the numeric denominator is automatically ended when the last numeric value was brailled

*Example* the fraction (7/20) - (3/24) (the brackets are being omitted)


7/20-3/24	 # g ; - # c ; /
-----------	--

Basic rules for other fractions:

- start the fraction with the following characters:

	 ! ;
--	--


- write the part for the numerator (without any spaces)
- add the fraction sign

	 8
--	--

- write part for the denominator (without any spaces)









*Example* consider the following fraction (brackets have been omitted)

$(p + q) / n$	 ! ; p ! + q 8 n 5
---------------	--


### 3.1.3.9 Subscripts, superscripts and indices


The following table gives an overview of the braille-codes that are used to form superscript- and subscript characters.


top right	
bottom right	
top left	
bottom left	
top middle	
bottom middle	

If the index is a numeric value, than its value is written as it would normally be except for the fact that all dots are lowerd one row.

*Examples*

$x^2y$	 x   < y
--------	--

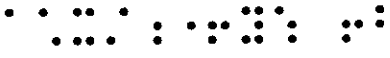
$x_1, x_2$	 x 1 , ' , x 1 <
------------	--

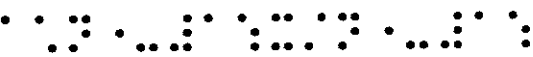
$x_2^3$	 x 1 <   :
---------	--

If the index is a non-numeric value, the following rules must be applied:

- start the index with the braille-character that were described in the table on the previous page
- write the 'index-value' stripped from all spaces
- close the term with the  $\cdot$  character (this character can be replaced by a space)

*Examples*

$a_{x^2+y} + b$	 a 1 x   < ``+ y 5 + b
-----------------	--

$a_{n-1}x^{n-1}$	 a 1 n ``- # a 5 x   n ``- # a 5
------------------	---

3.1.3.10 Integrals, summations, products and limits

The following characters are used to describe integrals, summations, products and limits:

$\cdot$	$\int$ integral-sign
$\cdot\cdot$	$\Sigma$ summation-sign
$\cdot\cdot\cdot$	$\Pi$ product-sign
$\cdot\cdot\cdot\cdot$	limits

The indices that go with these symbols are writtn down in the same way as discussed in section 3.1.3.8 in this document.

Examples

$\int_a^b f(x)dx = [F(x)]_a^b$	
	$\sim 1 a 5   b 5 f 2 x \backslash d x = \prime \prime$
	$\{ \$ f 2 x \backslash \} 1 a 5   b  $

$\sum_{i=1}^{i=6} t_i$	
	$\otimes s 1 i \prime \prime = \# a 5   i \prime \prime = \# f 5 t 1 i 5$

3.1.3.11 Avoiding potential confusions

In mathematical formulas it is possible that a term consists of both numbers and normal characters.

Consider for example the term 2za

2za	
	# b z a

Here, no confusion is possible. The line starts with the number prefix (dots 3, 4, 5 and 6), so the next character (which is normally a small b) would be interpreted by a two. The third cell shows the letter 'z'. The occurrence of the letter 'z' disables the number-prefix.


Now consider the term 2ax

When we would enter the following:

	# b a x

The result would be 21x, because the term started with the number-prefix (dots 3, 4, 5 and 6) in order to represent the number 2. When the reader reaches the third cell, being the letter a, he will interpret the cell as being the number 1 because the number-prefix is still active.


To inform the reader that the information following the number 2 is no longer to be interpreted as a numeric value, we can use dot 6. The Braille input then becomes:

2ax	 # b ' a x
-----	--


The situation described above is not the only case in which **DOT 6** can help the reader to interpret the information that is being presented. Other possible confusions that can occur are:

- The term sin a

In this case dot 6 should be used to inform the reader that the word 'sin' should be interpreted as a symbol. So the input should become:

sin	 s i n ' a
-----	--


- When switching between two fonts (example αx)

αx	 < a ' x
----	--

This line started with dots 5 and 6 which indicate that the information that follows should be interpreted as being Greek letters. Dot 6 is being used in the third cell to inform the reader that the Greek-letters have ended.

- When a punctuation-character can misinterpreted

Example: the term Is 2 + 3 = 5?

Is 2 + 3 = 5?	 \$ i s # b ! + # c = # e ' 5
---------------	--

In this case the last cell (dots 2 and 6) could be mistaken for a '?'-character. In order to inform the reader that should be interpreted as '?' we put dot 6 in front of the punctuation-character.

### 3.1.3.12 Special symbols in Braille

A list of special symbols is included in Appendix B. It contains a list of operators (binary and relational) and their translation in the Woluwe code.

## 3.2 The rendering process

### 3.2.1 The steps involved in the translation process

The renderer will navigate through the Inmath tree in depth-first manner. For each node, the element type determines which Woluwe codes need to be generated. There are in fact several types of nodes:

- container elements which don't generate any output, but are useful for determining the context (e.g. the integrandum of an integral construct).
- container elements which generate Woluwe codes, which indicate the type of construct (e.g. the boundary elements of the integral construct).
- textual elements: they contain the actual data

The resulting Braille string will always be generated using the full tree. This is because the codes which need to be generated, depend on the full context.

The translation of the data contained in the textual nodes into correct Woluwe code is not always a straightforward process. In a fraction for example, which consists only of simple numbers, one has to use lowered dots in the denominator. Sometimes a 'restore character' is needed to disambiguate what follows (for instance a latin character after a greek character...). Also the entities need to be translated correctly as defined in Appendix B.

### 3.2.2 An example of a simple translation

Suppose the user has already input the following equation (via keyboard or speech commands for example):

$$\lim_{x \rightarrow 0} f(x) \quad (1)$$

This will be exported by the Grif editor in Euromath+ format as:

```
<math>
  <lim.cst>
    <l.part.c>
      <Range>
        <Textual>x &rarr; 0<Textual>
      </Range>
    </l.part.c>
  <r.part.c>
    <Textual>f(x)</Textual>
  </r.part.c>
</lim.cst>
</math>
```

In the next step, the renderer calls the parser, which translates the previous into an inmath tree represented as follows:

```
<math>
  <lim.cst>
    <l.part.c>
      <Range>
        <Opernd><Text>x</Text><Opernd>
        <Operat><Text>&rarr;</Text><Operat>
        <Opernd><Text>0</Text><Opernd>
      </Range>
    </l.part.c>
  <r.part.c>
    <Textual>f</Textual>
    <paren.cst>
      <included>x</included>
    </paren.cst>
  </r.part.c>
</lim.cst>
</math>
```

Now, the renderer will start to analyse the tree. The first node which occurs is the <math> node. As this node is simply a container element, the renderer continues to convert its children, the first being a <lim.cst> node. This generates the characters 'l', 'i' and 'm' as part of the result string.

```
l i m
```

The next node which generates data is the <Range> node. In the woluwe code, one should indicate the range of a limit construct with a subscript sign (1). When the full <l.part.c> is being handled the result is (EuroBraille notation):

```
l i m 1 x : , # j 5
```

The '5' at the end is generated by the end tag of the <Range> node. The final result is (EuroBraille notation):

```
l i m 1 x : , # j 5 f 2 x ^
```

### 3.2.3 Representation of empty constructs

When the user edits a fraction for example, and deletes the entire contents of the denominator, Grif uses a small grey box to represent the empty place. On the Braille display, we will use something similar: dots {1234568} will indicate an empty construct.