New Graphic Symbols for EQN and NEQN

Carmela Scrocca

Bell Laboratories
Murray Hill, New Jersey 07974

ABSTRACT

There is now available on UNIX and GCOS a set of special characters frequently used in technical typing. In the past, authors have sometimes written out these symbols in English; others just assumed their secretary or typist had these symbols ready and waiting. These characters, however, are not part of the standard terminal or typesetter character sets, but are built-up of those already available. They can presently be produced for phototypesetter output by using EQN/TROFF; NEQN/NROFF can be used for computer terminal output.

This document displays these characters, shows how to use them, and discusses what is involved in making a special character.

New Graphic Symbols for EQN and NEQN

Carmela Scrocca

Bell Laboratories
Murray Hill, New Jersey 07974

Introduction

There is now available on UNIX and GCOS a set of special characters frequently used in technical typing. These characters supplement the ones that come with the typesetter and terminal which both have their own set of standard characters. These special characters are accessed through the math typesetting programs NEQN and EQN.[1] Processed through the NROFF and TROFF[2] formatting programs, the characters can be output on either a computer terminal or a phototypesetter. Using various NROFF/TROFF conventions, two or more of these existing characters can be built-up and pieced together to draw a new character.

Sections 1 and 2 of this document give a list of these characters and tell how to access and use them. In Sections 3 and 4, the reader will see what is involved in making a special character for both phototypesetter and computer terminal output.

1. The Characters

Table 1 gives a list of the characters, their meanings, and the names by which EQN recognizes them.

The user should be aware that these special characters are not built into the NEQN/EQN program, but are stored in directory /usr/pub; the filename is *eqnchar*. In order to use any of these symbols this file will have to be referenced. This can be done by using file *eqnchar* as the first filename in your output command, such as

```
neqn /usr/pub/eqnchar filenames | nroff
```

for computer terminal output. For phototypesetter output use EQN/TROFF instead of NEQN/NROFF. On GCOS, the characters are in file ./eqnchar.

Some users will find a constant need for only a few of the special characters. It may be convenient for these users to take their few selected characters, and write them into a file in their own directory. Other users may only need a few characters for use in one particular document. They could edit /usr/pub/eqnchar, copy the desired character definitions into a separate file, and read that file into the beginning of their document file.

Appendices 1a and 1b provide additional characters with their corresponding EQN names. The characters list in Appendix 1a are from the phototypesetter character set and have been assigned EQN names. /usr/pub/eqnchar must be referenced in the output command to use characters in this set. Appendix 1b contains a list of characters already built into EQN and can be used directly with the program.

character name	character	EQN name
sum of two elements	$\oplus$	ciplus
product of two elements	$\otimes$	citimes
is congruent to	$\cong$	=wig
approximately equal to	$\doteq$	=dot
equals by definition	$\triangleq$	=del
large star	$\ast$	bigstar
centered star	$\ast$	star
or	$\vee$	orsign
and	$\wedge$	andsign
for all	$\forall$	oppA
there exists	$\exists$	oppE
is included in	$\sqsubset$	incl
not a member of	$\notin$	nomem
angstrom	Å	angstrom
less than or approximately equal to	$\lesssim$	<wig
greater than or approximately equal to	$\gtrsim$	>wig
not less than	$\not<$	|<
not greater than	$\not>$	|>
left angle bracket	$\langle$	langle
right angle bracket	$\rangle$	rangle
hbar	$\hbar$	hbar
parallel	$\parallel$	||
perpendicular	$\perp$	ppd
angle	$\angle$	ang
right angle	$\llcorner$	rang
implies and is implied by	$\longleftrightarrow$	<->
implies and is implied by	$\Longleftrightarrow$	<=>
vertical ellipsis	$\vdots$	3dot
therefore	$\therefore$	thf

Table 1. The Special Characters

2. Usage

The reader should be familiar with EQN as these characters work with the program and are used in the same manner as any other mathematical symbol. An EQN name must be separated from surrounding input (with spaces) in order to be recognized as a special character. For example, just as you would say

```
pi~=~sum x sup i
```

to get

$$\pi = \sum x^i$$

you could say

x sub 3 ~=wig~ pi star y sub ppd

to give you

$$x_3 \cong \pi * y_\perp$$

of which =*wig*, *star* and *ppd* are a few of the new special characters. These symbols will work in both displayed and in-line equations. There are no bold or italic versions of the symbols.

3. Creating Special Characters for EQN

The symbols discussed here were created by taking pre-existing characters and piecing them together with the input and output conventions and escape sequences made available through NROFF and TROFF. Most commonly used here were the local horizontal and vertical motions, overstrike and zero width functions, and the point-size change function. These are used primarily for phototypesetter output; making characters for computer terminal output will be discussed in Section 4. The definitions of all the special characters are listed in Appendix 2.

Local horizontal and vertical motions are used to move a character up, down, or to the left or right depending on where you want to position your character. These motions are generated by the escape sequences \u, \d, \r, \v and \h. The motions are expressed in terms of ems; an em is approximately the width of the letter 'm'. By using ems, the amount of motion will always be in proportion to character size. The \u and \d sequences give vertical motions of ½ em up and down, respectively; \r gives an upward motion of 1 em. The \v (vertical motion) and \h (horizontal motion) escape sequences allow you to move any fraction of an em. The distance must be enclosed in ' marks and the direction of movement can be indicated by making it either positive or negative. For example, if you wanted a downward vertical motion of 3/10 of an em, you would say \v'.3m', and an upward vertical motion of 6/10 of an em would be \v'-.6m'. The same basic rules apply to horizontal motions where positive moves to the right, and negative to the left. The "is much greater than" symbol >> shows a simple horizontal motion:

>\h'-.3m'>

(>> is not a special character, but built into EQN; see Appendix 1b.)

The overstrike function \o simply overprints characters on top of another, centered on the widest character. This function was used to create the "sum of two elements" symbol ⊕ by saying

\o'\(pl\(ci'

where \(pl and \(ci are the escapes for + and O. The string of characters to be overstruck must be enclosed in ' marks. When using the overstrike function be sure not to use any motions or it will not work.

Similar to the overstrike function is the bracket building function \b. Instead of centering one character on the other, the bracket building function piles the characters vertically. The "angle brackets" were built-up using the \b function:

\s-3\b'\(sl\e'\s0 ⟨

\s-3\b'\e\(sl'\s0 ⟩

(\e and \(sl are the escapes for \ and /.) The first character in the string is positioned at the top of the pile, and on down with the last character at the bottom. \s is the escape sequence for a size change; we'll get to this in a while.

The zero width function \z enables you to print a character without moving after it is printed. \z often makes it easier to position your next character rather than figuring out where the first one moved you to. This function is used with \z*n* where *n* is the character to be printed. \z can only be applied to one character at a time so the sequence would have to be repeated. An example of this can be the definition of the "vertical ellipsis" ⋮ where \z forces the dots to remain in place; otherwise, horizontal motions would be needed to realign the dots.

```
\v'-.8m'\z.\v'.5m'\z.\v'.5m'.\v'-.2m'
```

The zero width function can also be used to darken a character: \z*xx* will print an *x*, stay in place, and print it again.

The point-size change function is used frequently to make one character fit in proportion to another. The escape sequence for this function is \s. The change in size would be indicated by however many point sizes you want to change. \s-2 will cause a reduction of 2 point sizes while \s+2 will enlarge by 2 point sizes. For example, in making the "equals by definition" symbol ≜, the Δ was made slightly smaller to fit comfortably over the =. This was done by

```
\v'.3m'\z=\v'-.6m'\h'.3m'\s-1\(*D\s+1\v'.3m'
```

hence reducing the Δ (\(*D is the escape for this) by 1 point size before printing it, and then returning back to the previous size. \s0 can also be used to bring you back to the previous size.

Another handy tool is the font change function \f. Since EQN automatically sets its characters in italics, it will be necessary to specify any other font. To change fonts, use \f*x* where *x* is the desired font (R for roman, B for bold, I for italic, etc.). \fP will revert back to the previous font. A prime example of this would be the "angstrom" symbol Å:

```
\fR\zA\v'-.3m'\h'.2m'\(de\v'.3m'\fP\h'.2m'
```

where the capital "A" is always printed in the Roman font.

A few words of caution. If in building your character you have used vertical motions or point-size or font changes, you must remember to undo them, or whatever follows will be off the main line or in the wrong size or font. It may also be necessary at times to use a horizontal motion to ensure enough space before and after your character. Also, you should test your character before putting it to actual use. Try it out with changes in original font or point size and see how it reacts to the change; don't be surprised if your character falls apart.

When you are satisfied with your character and it is ready to use, introduce it into your file with the define facility provided by EQN. For example, you would define the "not greater than" symbol (≯) as

```
define |>  % "\o'>\(or" %
```

Now by using |> in an equation, you will get ≯.

In the previous example, the %'s and "'s are two different kinds of delimiters. The outside pair are essential to mark the beginning and end of the definition. (%'s were used, although any character will do as long as it is not used in the definition.) Also, any definition containing TROFF commands should be enclosed in " marks, so EQN will treat the TROFF commands as a unit.

4. Creating Special Characters for NEQN

To get a special character to print out on a computer terminal is not quite as involved as on the phototypesetter. You are restricted to working with only the characters available on the print wheel and movement is limited. Vertical motion can be obtained by \u and \d but this will only give you a motion of ½ line space per escape sequence. Spacing and backspacing is about all the horizontal motion you'll get. You can, however, use NEQN to define a character as was done with the "less than or approximately equal to" symbol ≲:

ndefine <wig % < from "~" %

All TROFF conventions work in NROFF, but because of the lack of characters, they may produce strange effects when using NEQN/NROFF. Therefore, it may be necessary to separately define characters for phototypesetter and computer terminal output. *tdefine* applies to definitions for EQN; *ndefine* works with NEQN. *define* uses the same definition for both.

Built-up characters for terminal output are usually just good enough for identification's sake; generally, they look lousy. However, given the time and additional effort, these characters can be refined so that their output on the terminal is quite satisfactory.

Acknowledgements

I would like to thank J. F. Ossanna, M. E. Lesk and other members of Center 127 for all their help; and special thanks to B. W. Kernighan for his teaching, guidance and patience throughout. I am especially grateful to S. P. Morgan for his encouragement and shared enthusiasm; without his aid and the concurrence of Dept. 7133K supervision, none of this would have been possible.

References

1. "A System for Typesetting Mathematics," B. W. Kernighan and L. L. Cherry, Computing Science Technical Report #17.
2. NROFF/TROFF User's Manual, J. F. Ossanna, BTL internal memorandum.

Appendix 1a

Additional Symbols from Phototypesetter Character Set

	character	EQN name
	¼	quarter
	¾	3quarter
	°	degree
	□	square
	○	circle
	■	blot
	●	bullet
	≃	−wig
	~	wig
	∝	prop
	∅	empty
	∈	member
	∪	cup
*	∩	cap
*	⊂	subset
*	⊃	supset
*	⊆	!subset
*	⊇	!supset

* NEQN/NROFF does not produce these symbols.

Appendix 1b

Additional Symbols Provided by EQN

	character	EQN name
*	½	half
	≈	approx
	≥	>=
	≤	<=
	≫	>>
	≪	<<
	→	->
	←	<-
	≡	==
	≠	!=
	±	+-
	∞	inf
	∂	partial
	′	prime
	·	cdot
	×	times
	∇	grad
	. . .	...
	, ... ,	,....,
	Σ	sum
	∫	int
	∏	prod
	∪	union
*	∩	inter

* NEQN/NROFF does not produce these symbols.

Appendix 2

Note: — represents a backspace in ndefined characters.

```
.EQ
tdefine ciplus % "\o'\(pl\(ci'" %
ndefine ciplus % O—+ %
tdefine citimes % "\o'\(mu\(ci'" %
ndefine citimes % O—x %
tdefine =wig % "\(eq\h'-\w'\(eq'u-\w'\s-2\(ap'u/2u'\v'-.4m'\s-2\z\(ap\(ap\s+2\v'.4m'\h'\w'\(eq'u-\w'\s-2\(ap'u/2u'" %
ndefine =wig % =—"" %
tdefine bigstar % "\o'\(pl\(mu'" %
ndefine bigstar % X—|—— %
tdefine =dot % "\z\(eq\v'-.6m'\h'.2m'\s+2.\s-2\v'.6m'\h'.1m'" %
ndefine =dot % = dot %
tdefine orsign % "\s-2\v'-.15m'\z\e\e\h'-.05m'\z\(sl\(sl\v'.15m'\s+2" %
ndefine orsign % \e/ %
tdefine andsign % "\s-2\v'-.15m'\z\(sl\(sl\h'-.05m'\z\e\e\v'.15m'\s+2" %
ndefine andsign % /\e %
tdefine =del % "\v'.3m'\z=\v'-.6m'\h'.3m'\s-1\("D\s+1\v'.3m'" %
ndefine =del % = to DELTA %
tdefine oppA % "\s-2\v'-.15m'\z\e\e\h'-.05m'\z\(sl\(sl\v'-.15m'\h'-.75m'\z-\z-\h'.2m'\z-\z-\v'.3m'\h'.4m'\s+2" %
ndefine oppA % V—— %
tdefine oppE %"\s-3\v'.2m'\z\(em\v'-.5m'\z\(em\v'-.5m'\z\(em\v'.55m'\h'.9m'\z\(br\z\(br\v'.25m'\s+3" %
ndefine oppE % E—/ %
tdefine incl % "\s-1\z\(or\h'-.1m'\v'-.45m'\z\(em\v'.7m'\z\(em\v'.2m'\(em\v'-.45m'\s+1" %
ndefine incl % C—_ %
tdefine nomem % "\o'\(mo\(sl'" %
ndefine nomem % C———/ %
tdefine angstrom % "\fR\zA\v'-.3m'\h'.2m'\(de\v'.3m'\fP\h'.2m'" %
ndefine angstrom % A to o %
tdefine star %{ roman "\v'.5m'\s+3*\s-3\v'-.5m'"}%
ndefine star % * %
tdefine || % \(or\(or %
tdefine <wig % "\z<\v'.4m'\(ap\v'-.4m'" %
ndefine <wig %{ < from "" }%
tdefine >wig % "\z>\v'.4m'\(ap\v'-.4m'" %
ndefine >wig %{ > from "" }%
tdefine langle % "\s-3\b'\(sl\e'\s0" %
ndefine langle %<%
tdefine rangle % "\s-3\b'\e\(sl'\s0" %
ndefine rangle %>%
tdefine hbar % "\zh\v'-.6m'\h'.05m'\(ru\v'.6m'" %
ndefine hbar % h—\u—\d %
ndefine ppd % _—| %
tdefine ppd % "\o'\(ru\s-2\(or\s+2'" %
tdefine <-> % "\o'\(<-\(->'" %
ndefine <-> % "<——>" %
tdefine <=> % "\s-2\z<\v'.05m'\h'.2m'\z=\h'.55m'=\h'-.6m'\v'-.05m'>\s+2" %
ndefine <=> % "<=>" %
tdefine |< % "\o'<\(or'" %
ndefine |< % <—| %
tdefine |> % "\o'>\(or'" %
ndefine |> % |—> %
tdefine ang % "\v'-.15m'\z\s-2\(sl\s+2\v'.15m'\(ru" %
ndefine ang % /—_ %
tdefine rang % "\z\(or\h'.15m'\(ru" %
ndefine rang % L %
tdefine 3dot % "\v'-.8m'\z.\v'.5m'\z.\v'.5m'.\v'-.2m'" %
ndefine 3dot % .—\u.—\u.\d\d %
tdefine thf % ".\v'-.5m'.\v'.5m'." %
ndefine thf % ..—\u.\d %
.EN
```