1 July 1981

81-005

A Conceptual Model of Text Editing and Formatting

Peter Calingaert

The University of North Carolina
Department of Computer Science
New West 035A
Chapel Hill, NC 27514

INTRODUCTION

This report presents a conceptual model intended to facilitate the description of editing and formatting systems, whether for evaluation or for design. The texts to which the model applies are composed of characters drawn from multiple alphabets. These may include super- and subscripts, as well as italic and boldface characters, and spaces of different widths, as long as the finished text can be described in terms of lines of characters extending the full width of a print column. Although such a characterization is less general than the "boxes" and "glue" defined by Knuth [1], it does encompass most business and scholarly documents.

The model distinguishes clearly between content and format, and relies on the essential dissimilarility of a document's horizontal dimension to its vertical dimension. The model identifies three major conceptual representations of a text, describes seven editing, formatting, and other transformations on those representations, and introduces a consistent terminology.

Rather than invent a lot of new terminology to distinguish entities in the model from those in actual systems, I have chosen some familiar phrases to capitalize on their evocative power. If a reader interprets one of these phrases in the context of an actual system with which he or she is familiar, the image evoked may be more specialized or more detailed than I intend. The reader is therefore urged not to ascribe automatically to model entities the attributes of a similarly named entity in any real system.

The objective of the systems considered is to produce a physical representation of a text. There are two kinds of information embodied in the text: content and format. In the final physical document, only the characters that constitute the content appear explicitly. The format is implicit in where the content appears. In the source text, however, the formatting information occurs in the form of explicit format controls. Restructuring of the text, performed by positioning its content according to the format controls, which are simultaneously removed, is called formatting. Modification of any part of the text, whether content or format, is called editing.

FORMATTING

The source text could be as concrete as a typescript for an OCR reader or a manuscript to be copied by a keyboard operator, or it could be created dynamically by an author working at a keyboard. The final physical document consists of pages, each made up of lines, and produced by a device such as a character- or line-printer, or a phototypesetter. These devices complete the physical production of one line before proceeding to that of the next. There is thus an essential asymmetry in the two axes of each page. Formatting involves the vertical placement of lines, which have been previously defined by the horizontal placement of characters. Format specifications are therefore divided into two classes, horizontal and vertical.

Examples of horizontal specifications are line width, justification, tabulation, and initiation of paragraphs. Examples of vertical specifications are page depth, pagination, control of widows and orphans, page starts, and floating of text to avoid splitting a table or other unit across a page boundary. Some specifications have separate horizontal and vertical components. Among these are footnotes and references thereto, and headers and footers. Of the horizontal and vertical format specifications, some are global in effect, with scope to the end of the text unless sooner rescinded. Line width and page depth are examples. Others are local, to be obeyed where encountered, but without pervasive effect. Examples are tabulation and page starts.

There are four major transformations in the formatting process. The first is to convert the input text from a sequence of key strokes, or other dynamic or static form, into a computer representation as a <u>string</u> of characters. This can be thought of as <u>capture</u> of the key strokes, to obviate their being repeated if the document is edited. The second step is to format the characters horizontally into a set of lines, which in the printing industry is called a <u>galley</u>. This step is equivalent to the "setting" of type or "composition" of text. The third step is to format the lines vertically into one or more columns on each of a book of pages. This is equivalent to page "layout" or "makeup". The fourth step is to produce physical images of the pages, typically on paper or on photographic film. Figure 1 depicts the progression from input text to physical document, through three intermediate forms represented by rounded boxes. The rectangular boxes represent processing transformations.

The string of characters that result from key stroke capture includes content and format controls, both vertical and horizontal, all explicitly. The galley of lines that results from horizontal formatting includes explicit content and vertical format controls, whereas the horizontal format information is now implicit in the content. The book of pages that results from vertical formatting includes explicitly only the content. Both vertical and horizontal format specifications are implicit in the resulting document, which is now fully formatted. These changes in the degree of formatting are reflected in Fig. 2. It is true that the format controls may be retained, rather than discarded, when formatting is performed. This is an implementation convenience, however, and not a logical requirement.

EDITING

A distinct editing transformation is defined for each of the three distinct representations of the text, as depicted in Fig. 3. Whereas formatting changes the mode of representation, but not the substance of the text, editing can change both the substance and the representation. The transformation performed by formatting can be viewed as being independent of the content of the text, but governed by the format controls embedded in the text. Formatting requires no input other than the text to be transformed. Editing, on the other hand, performs a transformation on the text, governed by information provided from outside the text. Means are therefore required to capture these editing commands, as shown in Fig. 4. Dashed lines represent command flow, whereas solid lines represent text flow.

Character editing modifies the text represented as a string of characters. Because an explicit vector of indices into the character string is as long as the string, most users will prefer to specify the target of their modifications not by index, but by context. This is a very practical, but not a logical, requirement. The basic operations of character editing are to insert, delete, and replace characters or, more generally, substrings of characters. A special case of character editing is text creation by insertion into a previously empty text. This simply uses the character editor as the transducer that captures the input text. Because the text content and both horizontal and vertical format controls are present in the character string, character editing alone suffices to make any modification desired in the final document.

Line editing modifies the text represented as a galley of horizontally formatted lines. Editing commands to change horizontal format control redefine how the characters are grouped into lines. Editing commands to change vertical format control have no visible effect on the galley, but result ultimately in a different page layout. In principle, line editing commands to modify content should insert, delete, and modify lines. A newly inserted or modified line may, however, not really be a line if in fact it does not meet the horizontal format specifications. A typical example is inserting so many characters into a line that its width exceeds the maximum. In such an event, reformatting to produce new lines is requested implicitly by the editing command.

Page editing modifies the text represented as a book of horizontally and vertically formatted pages. Editing commands to change format controls clearly result in a different setting of characters into lines and a different layout of lines into pages. In principle, page editing commands to modify content should insert, delete, and modify pages. A newly inserted or modified page may, however, not really be a page for failure to meet both vertical and horizontal format specifications. In such an event, reformatting to produce new pages is requested implicitly by the editing command.

IMPLEMENTATION

Probably no existing system implements all seven transformations described by the model. Implementations of nontrivial subsets thereof are common, however, and may be editors, formatters, or combined editor-formatters.

Editors

Pure editors, to avoid issuing implicit reformatting requests, do not manipulate either lines or pages. They are character editors, even though many treat the character string as being composed of "lines" of characters. These "lines" often correspond to punched cards or card images, or to lines on a terminal. The "lines" are usually numbered for reference by the editor. For some editors, use of line numbers is the only way to direct the editor's attention to a particular location in the text. For most editors, however, context-finding commands perform that function. The "line" structure is often appropriate to text whose content is a computer program. More flexible

editors (e.g. OCCAM [2]) do not impose a "line" structure on the character string. Such editors rely on context to reach a particular location in the text.

Editors intended for batch mode operation can be implemented separately from the input transducer. Much more common, however, is the combination of input transducer and character editor into a system for the initial entry and modification of arbitrary strings of text.

Formatters

Pure formatters accept as input a string of characters, already entered and edited. They perform the horizontal and vertical formatting functions, which may or may not be clearly segregated in the implementation. Production of output may be incorporated in the formatter function, especially on a microprocessor-based system, but it is often provided separately as a service of the operating system. Some formatters (e.g. SCRIPT [3]) require an input string that is organized into "lines". In most cases this requirement is due to basing the design on the use of card images.

There is wide variation among formatters in the extent of function they provide. The simplest may provide little more than line justification and pagination, whereas the most powerful offer extensive facilities. These include left-, center-, and right-aligned tabulation, hanging indention, automatic hyphenation, avoidance of widows and orphans, footnote placement, "floating keeps", and automatic generation of index and table of contents.

Editor-Formatters

An editor-formatter system can be composed of an editor and a formatter bolted together. One example of this type of system is APLTEXT[4], which combines a context-finding character editor with a separate formatter. The division between horizontal and vertical formatting is also sharp, and it is possible for the user to view his text as a galley of lines, prior to vertical formatting.

Another type of editor-formatter is the line editor, which performs line editing and the associated horizontal reformatting. Either the line editor incorporates horizontal formatting capability, or it invokes reformatting of the original character string by a separate horizontal formatter. In the former case, the formatting component must be able to accept already formatted lines as input. In the latter, the editing component derives either formatting commands for the horizontal formatter or editing commands for a separate character editor, which then presents modified input to the formatter. These secondary commands are represented in Fig. 5 by dotted lines, to distinguish them from the primary commands supplied to the line editor. Whether line editing and horizontal formatting are implemented together or separately, need arises to determine what horizontal format controls were present in the unformatted text. Unfortunately, formatting is not uniquely reversible, because there are many equivalent ways to specify the same placement of content information. A redundant representation of the text can, however, retain the format controls for this purpose.

Yet another type of editor-formatter is the page editor, which performs page editing and the associated horizontal and vertical reformatting. Examples include many commercially-available systems dedicated to word processing. Their users can view a text in fully formatted pages, specify editorial changes to either content or format, and then examine the result. If the lines of one page can be stored as a rectangular array of characters, page formatting capability can be readily incorporated in the page editor. This organization also permits the page editor to serve as the input transducer. Otherwise, the page editor can derive secondary commands either to separate formatters or to a line editor. As for a line editor, retention of a redundant representation of the text is helpful.

CONCLUSION

The foregoing model identifies three major conceptual representations of a text, describes seven transformations on those representations, and introduces a consistent terminology. The model is intended to facilitate the description of text-preparation systems, whether for evaluation or for design. The concentration on representations and transformations provides a framework for basing implementation decisions on the logical requirements of the application. Two simple examples will suggest why this is so. (1) The model shows the domains of the languages for editing and formatting. The degree of coordination desired between the languages is clearly dependent upon the extent of integration of the editing and formatting functions. (2) The decomposition of the over-all task into functional modules can be guided by the decomposition of the model. The designer can choose whether to segregate the conceptually distinct transformations.

In comparing and discussing text-processing systems, it will be helpful to avoid the terminological confusion that has plagued the discussion of operating systems. Although I decry premature or excessive standardization, and hold no strong brief for the nomenclature presented herein, proposing it may serve as a challenge to others to suggest better.

REFERENCES

1. Donald E. Knuth. TEX and METAFONT: New Directions in Typesetting. Digital Press, Bedford (MA), 1979.
2. James Sneeringer. User-interface Design for Text Editing: A Case Study. Software -- Practice and Experience 8(5): 543-557, September-October 1978.
3. Waterloo SCRIPT REFERENCE MANUAL, University of Waterloo, Waterloo (Ontario), 12 July 1979.
4. IBM. An APL Text Editor and Composer. Document SH20-1089, 1973.

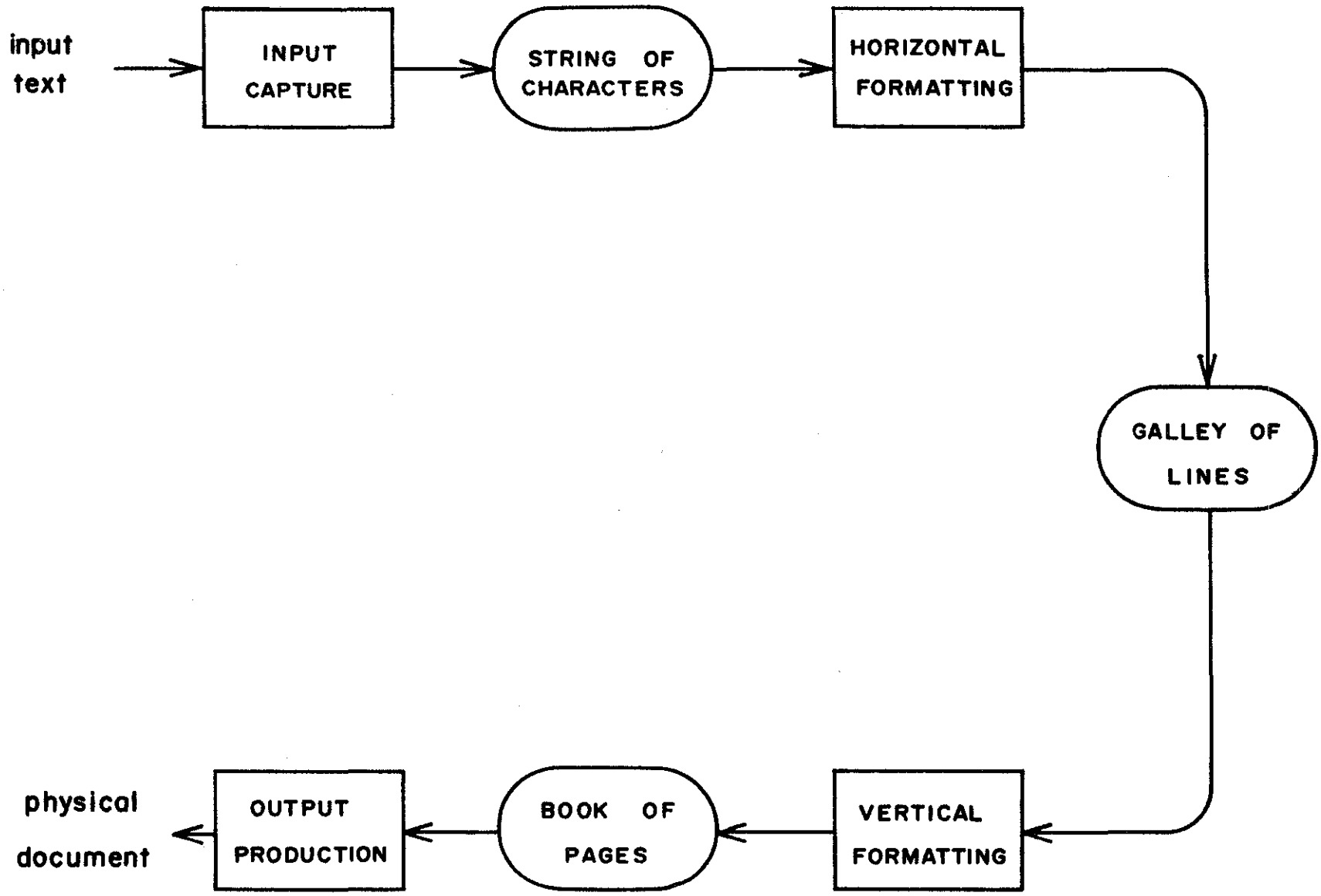

Figure 1. Progression from Input Text to Physical Document

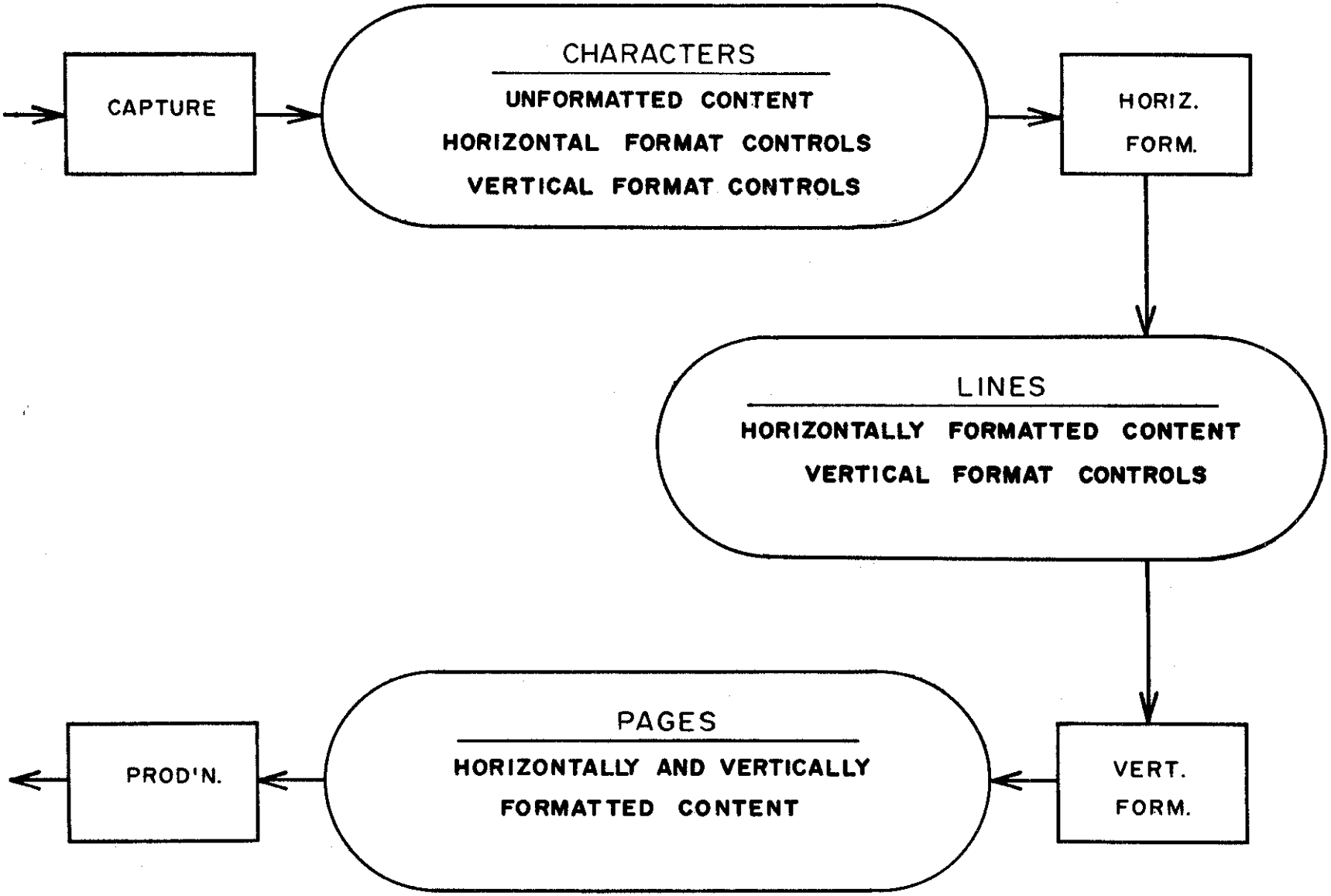

Figure 2. Changes in Degree of Formatting

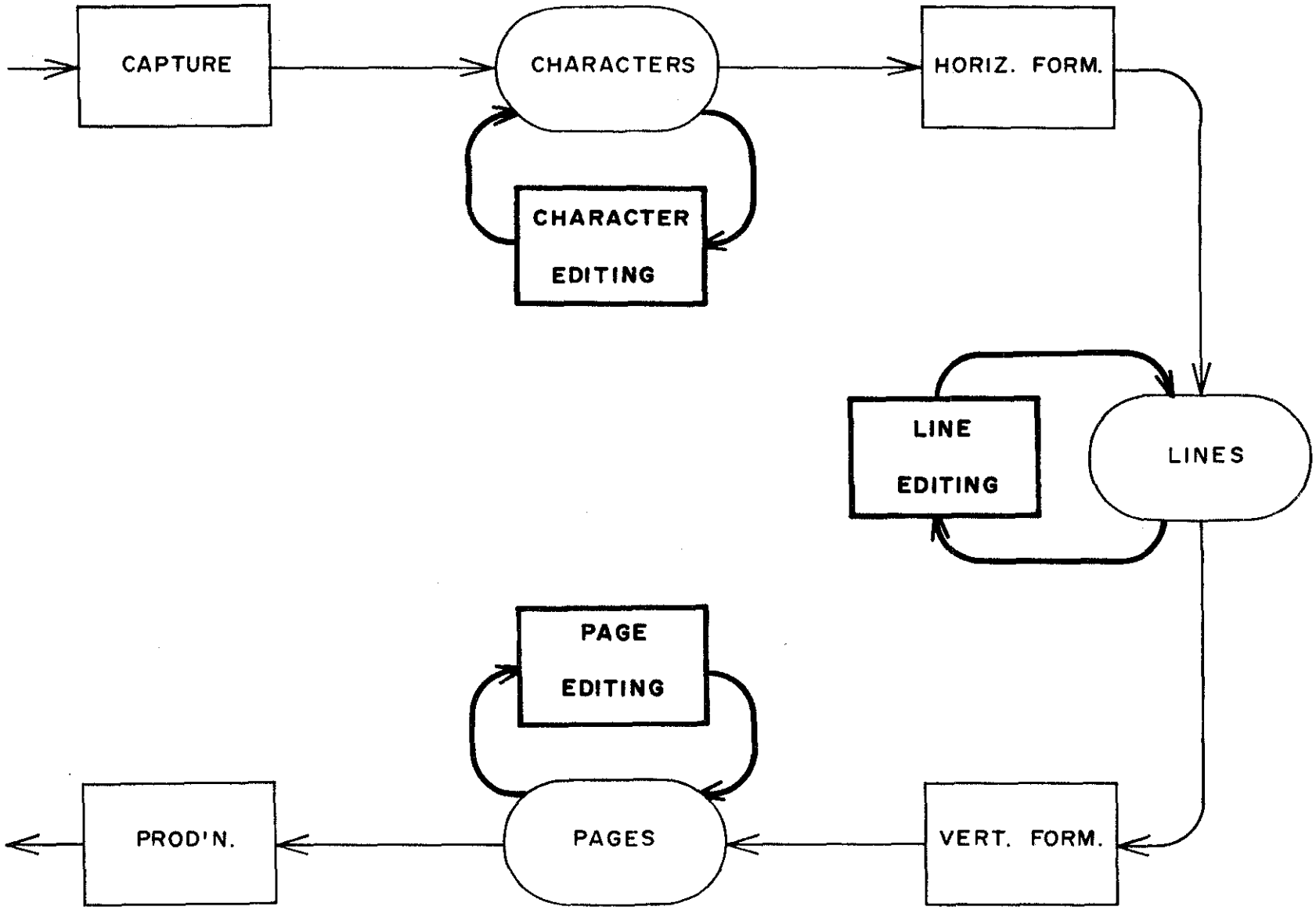

Figure 3. Logically Distinct Editing Transformations .

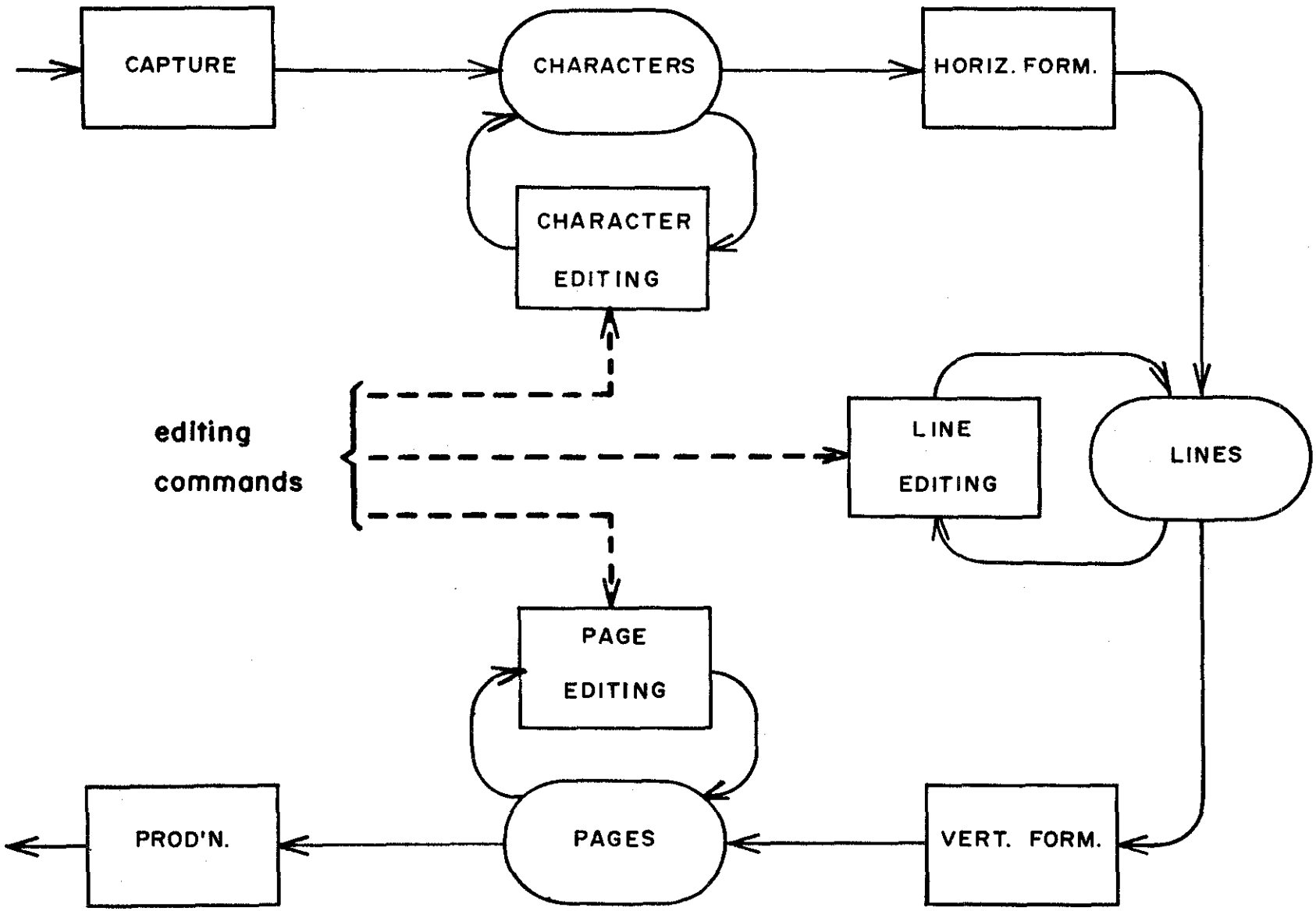

Figure 4. Primary Editing Commands

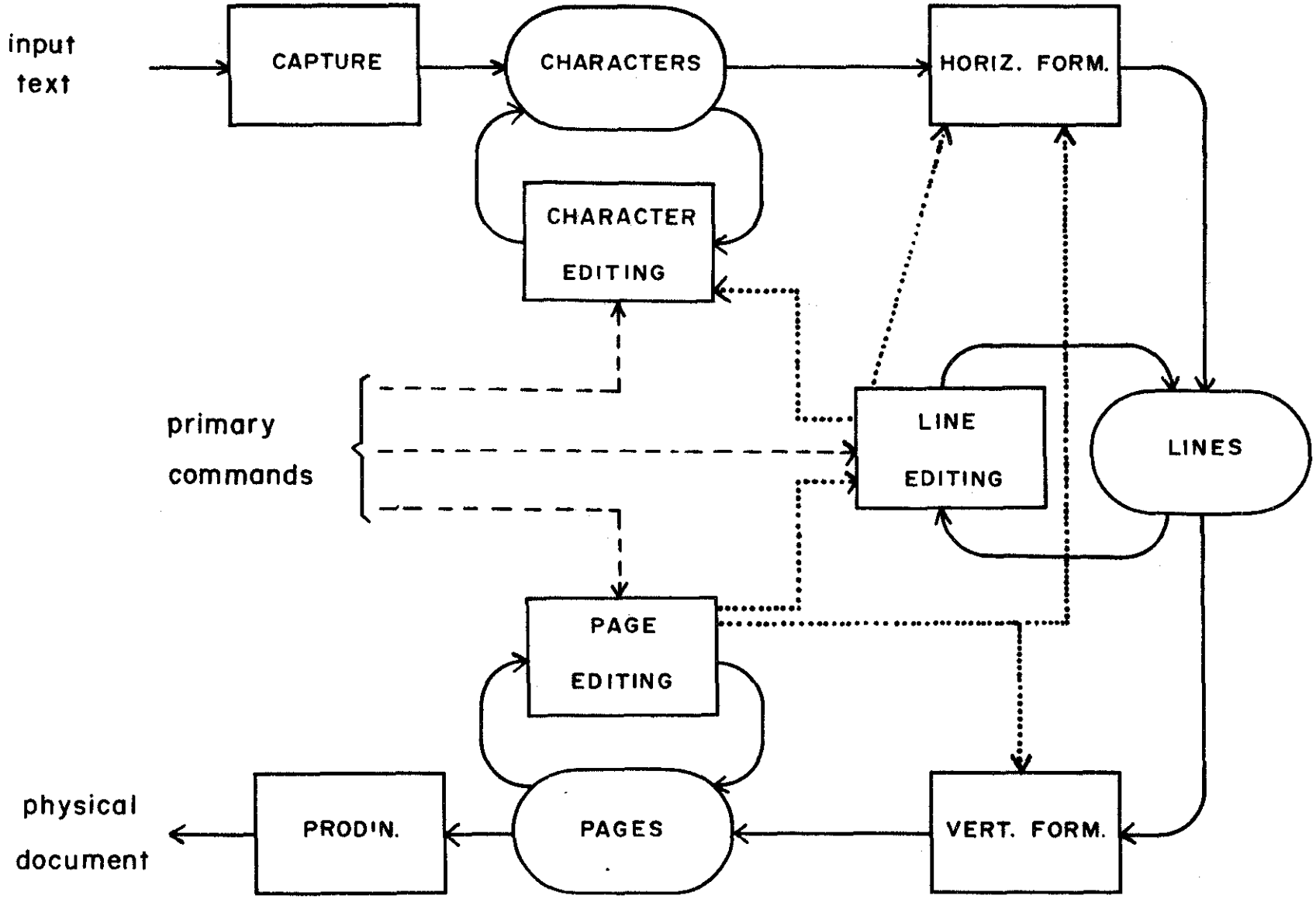

Figure 5. Derived Editing and Formatting Commands