The Pursuit of Quality*

How can automated typesetting achieve the highest standards of craft typography?

Frank Mittelbach† and Chris Rowley‡

November, 1991

Abstract

This paper compares high-quality craft typography with the state of the art in automated typesetting. The first part discusses several typographical conventions which cannot be implemented by means of any formatting model currently in use. The second part explains why the current paradigms of computerized typesetting will not serve for high-quality formatting and suggests directions for the further research necessary to improve the quality of computer generated layout.

Keywords: craft typography, automated typesetting, document formatting model, paradigm, typographic rules, visual context, logical context, global optimization.

1 Introduction

The preparation of quality typeset documents requires highly skilled understanding and application of concepts developed over hundreds of years in craft typography [16]. As the constraints imposed by a particular document (i.e., its unchangeable textual and logical content) usually result in conflicts between the generally accepted rules of quality typography, some of these rules have to be violated in favour of others. The skill of the compositor lies in the resolution of such conflicts between ideals. This results in the transformation of an abstract document into a visual form which is both aesthetically pleasing and faithful to the intentions of the author.

The nature of the author's intentions, and the details of how best to realize them visually, are largely outside the scope of this paper. For us, it is sufficient to know that the placement of objects in a physical representation of some particular document is (or should be) the result of a set of rules together with some mechanism for resolving conflicts. In craft typography these conflicts are typically resolved by interaction between the typographic designer and the compositor [32].

The equivalent process for automated typesetting is a computer program whose input is a form of the document containing only the text and logical mark-up and whose output is a complete description of the detailed content of each page of the typeset form of the document. As RICHARD SOUTHALL [27] has written:

> In formatting a document on a computer-based system, the graphic characteristics of blocks of text and their arrangement on the printed page are dictated by presentation rules which are determined by the format designer. The detailed arrangement of characters and spaces within the lines of text that make up a block is also governed by rules, ...

To have any chance of emulating the traditions of craft typography a program must, in some sense, understand such rules and be able to implement them. Moreover, it must contain mechanisms for resolving conflicts between these rules. It is also important to note that the set of rules which form the basic description of the layout for a document, or a class of documents, (i.e., the concepts for positioning and shaping the contents) is potentially very large.

We have therefore split the requirements specification of a program for the production of high quality documents into two major components:

- Any of the possible rules which form the basic layout must be implementable and applicable by the program.

† EDS, Electronic Data Systems (Deutschland) GmbH, Eisenstraße 56 (N15), D-6090 Rüsselsheim, Federal Republic of Germany, Internet: mittelbach@mzdmza.zdv.uni-mainz.de

‡ Open University, Walton Hall, Milton Keynes, United Kingdom, Janet: ca_rowley@vax.acs.open.ac.uk

- Whenever the application of the rules produces a conflict, a resolving mechanism needs to be definable.

The challenge which this paper poses is, in computer science terms, to find suitable data-types whose transformations model well both the rules of the craft and the heuristics for conflict resolution. A necessary prerequisite for this is a suitable codification of the rules of craft typography and an evaluation of methods for automating its heuristics. We are not suggesting that the software should mimic exactly the skills of a craft typographer but that it should aim to produce typographical results which are, to the trained eye, indistinguishable from what would be expected from a reputable compositor.

Furthermore, we do not intend to give the impression that there is no need for well-designed systems which support interaction between a human operator and the typesetting software. However, in this paper we wish to focus on the requirements of the *automated* part of any system which aims at typographic excellence.

We start by looking briefly at a few examples of the typographer's craft which are addressed imperfectly, if at all, by existing typesetting software. Section 3 then analyses some of the basic requirements of a model for the design of software to automate craft typography and Section 4 summarizes the current state of the art; finally we suggest some directions for future research.

The authors wish to thank Richard Southall for many helpful discussions on subjects related to this work and Reinhard Wonneberger for his comments on early drafts of this paper.

2 A brief look at some rules

The rules of craft typography were developed to help the reader to understand the contents of a document [25, 26, 28, 29, 31]: good layout does not distract the reader's attention from the main aspect of the document—its message. Good typography therefore is a silent art; not its presence but rather its absence is noticeable.

Naturally, few typographical rules are universal: they depend on the purpose of the document, the cultural background of the prospective readers and many other things. Examples of the cultural aspect of typography range from obvious differences such as the direction of the typesetting (left-right, right-left or vertical) to more subtle distinctions such as the use of a type-face suited to the language of the document. Quite noticeable changes in the grey-values of paragraphs occur when the same type-face is applied to different languages ([25, p.43]), e.g., German with many capitals, French with its accents, etc.

The rules and concepts also depend on the school of typography and on particular house styles [5, 21, 6, 11]. Therefore, the discussion in this section is intended only to give some examples of rules which a designer *might* want to specify for particular documents and is not meant to be a complete survey of the subject; it focuses on rules accepted by most western schools of typography but which are not, to our knowledge, achievable by the currently available systems for automated typesetting.

2.1 Line breaking

Line breaking and its concomitant, word-division, provide a good example of the need for balance between conflicting ideals. The breaking of paragraphs into individual lines is not done simply because of the constraints imposed by a given page width—it has many aspects, including the following.

- The choice of page-size, layout, text-measure and type-face appropriate to the subject matter: long eye movements should be avoid as they tire the reader and impede the reading process.
- The quality and quantity of hyphenated lines: hyphenated words, especially when they appear too often or use psychologically bad breaks, make reading difficult (see Section 2.1.1).
- The shape of the paragraphs and the distribution of white-space within them: getting this right often leads to conflicts with the previous constraint. The use of ragged-right typesetting can help this conflict but leads to other problems, which are discussed in 2.1.2 below.

2.1.1 Word-division

Hyphenation of words is a process deeply entangled with paragraph construction [17]. All modern computerized typesetting systems incorporate some automated system for hyphenating words when the need arises, but most treat all the possible hyphenation-points as equally acceptable. As RONALD MCINTOSH suggests in [18] (and has implemented in the *Hyphenologist* software [8]), this is incorrect as they usually vary in their 'goodness' and these differences will influence the overall quality of the document. A special case of this aspect of hyphenation is that an inserted hyphen should ideally be distinguishable from a hyphen that is part of the word (e.g., re-cover vs. recover), allowing the designer to set up rules which take into account whether such a word should be broken at such a point [7].

Additionally, the distribution of hyphens within a paragraph is normally subject to typographical rules such as "avoid more than three hyphens on consecutive lines".

2.1.2 Paragraph shape and white space

When words have to be shaped into some template, two basically different approaches are common: ragged-right or justified blocks of text [30].

With justified text the intention is to give a uniform appearance: the excess space on individual lines

needs to be distributed as evenly as possible over the whole paragraph, which makes a global optimization algorithm for determining the break-points necessary [1, 22, 23]. To ensure a similar grey value over the whole paragraph, differences between the excess space in adjacent lines must also be taken into account. However, it is not only the *average* distribution of white space which is important: it is, for example, also essential to avoid 'rivers' of white space flowing vertically through a paragraph [31].

Another refinement is that white space between words needs to be different for different combinations of boundary characters in order to give a uniform appearance [19].

If, on the other hand, ragged-right shapes are desired then the typesetting system must be capable of producing aesthetically pleasing shapes [26, p.131], e.g., avoiding successive lines of equal length and extremely uneven shapes (if the layout description specifies this). The absence of literature on this problem and the failure of any current typesetting systems to implement such rules suggests that the underlying concepts of ragged-right typesetting are not at all well understood.

Most of the problems covered here can, with most currently available systems, be detected only by human eyes. Furthermore, their circumvention is generally difficult and, at best, involves a large amount of detailed ad hoc interaction with a skilled human operator. That they have not at present been automated is in large part because the concepts necessary to describe and evaluate them have not yet been defined.

We are aware that there exist automated systems which go some way towards tackling particular aspects of the production of visually optimal paragraphs. The methods they use include the following: assessment of the 'average grey value'; use of 'weighted word-division points'; control of multiple consecutive hyphenations; optically correct inter-word spacing and optical alignment of the margins. However, we know none which do all of these, and none which make any explicit attempt to avoid 'rivers'

2.2 Grid layout

The eye needs a regular grid of points on which to focus when searching for objects (e.g., the next line) on a page. Many designers therefore base all aspects of the page-layout of a document on an underlying grid [13]. Such layout constraints are fundamental to the typesetting paradigms of many of the widely used DTP software packages but, regrettably, this is the *only* concept relevant to high quality typesetting which is understood by most such systems!

The distribution of white space in such a layout is necessarily discrete, which throws optimizing systems based on the box-glue model [14, 12] out of balance.

Since such a layout limits variation in the placement of constructed objects (like broken paragraphs, constructed figures with captions, etc.), to achieve good quality within such constraints there needs to be a high level of interaction between the building mechanisms for these objects and the page make-up mechanism (or, more generally, the algorithms for constructing higher level units). Such interaction is not implemented in any current systems (Section 3 contains a more detailed discussion of this topic).

2.3 Placing floating objects

Floating objects (e.g., footnotes or figures) are linked to the main body of a document by means of cross-references or other visual clues intended to identify them for the reader. Rules for their positioning can become quite complex to implement, even when they are easy to state [21]: e.g., that the distance between the reference and the float should ideally be small, or that the float should be visible from the point of its major reference.

Since floats should be easily distinguishable by the reader as individual units, clear visual separation from the main body of the text is usually necessary. This is typically achieved by font changes, extra space, rules, etc. thus such objects can quite drastically alter the visual appearance of a page. For this reason some designers prefer, for example, to lay out all the figures on a double-page spread as a single visual unit (placing individual figures and captions according to their size and form) to produce a balanced look.

To allow for such designs the software needs to be driven by heuristic rules as opposed to the procedural routines used in all current systems. Again, to emulate craft typography it is essential that complete interaction between this part of the system and all the other formatting routines is maintained: for example, it is necessary to allow paragraphs which flow around arbitrary figure shapes to, nevertheless, have globally optimized line-breaks.

Again, no currently available software implements more than a minimal amount of automation in this area.

3 Choosing the context

Each of the rules from the last section (along with many others) poses an intrinsically demanding and interesting implementation problem worthy of research effort—but it is not only the specification of certain important typographical conventions which is difficult or impossible within current systems. There is a more fundamental limitation which will prevent these systems from approaching craft standards, however good their formatting mechanisms may be—they have all been developed under the following paradigm:

> *That the formatting parameters for any object in some document can be determined by sequentially parsing its logical form.*

In other words, that these parameter values can be predetermined in such a way that the document can be processed in one pass (except for things like forward cross-references, which are resolved in a subsequent pass).

We claim that this approach will never suffice for the high quality standards of craft typography. As RICHARD SOUTHALL [27] went on to point out:

> With complex text, the achievement of character arrangements in the printed output that help to clarify its meaning to the reader may require differing rules of composition to be used in the formatting of different parts of the text.

We would extend his observation by pointing out that each object in a document should be formatted according to the context in which it finally happens to fall. This context has two components: its logical context and its visual context.

By **logical context** we understand the placement of the object (i.e., of its tag in the logical mark-up of the document) with respect to the other objects of the document. This component of the context is, in principle, fixed for a particular document (e.g., in a simple model for the logical structure of a document, it is given by the nesting and sequencing of the logical tags) but its analysis may involve substantial look-ahead or even a preliminary pass over the whole document.

The **visual context** of an object consists of the visual concepts and rules which are active at the place in the formatted document where the object actually appears (this includes, for example, the page(s) on which it falls, whereabouts on a page it appears and what else is on nearby pages). This visual component usually cannot be determined without complete knowledge about the placement (and consequentially the formatting) of all the other objects in a particular document. For example, the placement and measure for a figure caption might depend on whether that figure falls on a verso or on a recto page or whether there are other figures present on this particular page. Thus, if a 'floating figure' floats from a page of mixed text/figures to an 'all figures' page, its logical context will not change (since this is determined by the first reference to it in the text) but its visual context may change significantly.

Whilst the correct logical context for any object can in principle be determined after a single scan through the whole document, this is certainly not possible for its visual context since this depends on the formatting of all other objects within the constraints given by the applicable typographic rules. The above paradigm must therefore be replaced by one which recognizes that:

- formatting a document is a global process which must take into account variation in the visual context of each object;
- an iterative process is required to achieve an optimal (or even a high-quality) result.

Our basic model of automated document formatting is thus very similar to the model described in Section 1.2 of [10]. However, our refinement of the model is radically different from that developed in Section 1.3 of that article: this is probably an accurate reflection of the fundamental tension between the needs of a WYSIWYG system and those of automated high-quality typography. To achieve craft quality, detailed decisions must be made concerning the typographical treatment of all aspects of a document (from the layout of a spread down to the individual character glyphs). Such decisions depend, for a particular object, not only on its logical position in the abstract document but also on its visual relationship to the rest of the formatted document.

4 The current situation

The only major project which has addressed the problems of automating high quality typesetting remains that of Knuth (and developments thereof). There seems to have been nothing published on the theoretical side of this subject since 1982. The progress to that date is well described in *Document Preparation Systems* [20] from which we would particularly recommend the comprehensive survey by RICHARD FURUTA et al [9]. In this article the distinction between the editing process and the formatting process is clarified and a number of systems are described—both pure formatters and integrated editor/formatter systems. The authors then point out, in Section 4.5.1, that since all systems leave much of the task of producing satisfactory formatting to the user, close integration of the editing and formatting is essential since this makes "the generation of the concrete document part of a single document creation process".

Perhaps this explains the rapid advances over the last decade in the development of integrated editor/formatters, resulting in the present-day sophistication and range of systems incorporating the WYSIWYG paradigm of document formatting. By contrast, there has been very little progress in automating the formatting process itself and thus rendering unnecessary the human time and skills needed to produce high quality work within the WYSIWYG paradigm. It is encouraging to note that the emerging DSSSL standard [2] does allow expression of our paradigm—but this establishes no more than a common language in which to express these ideas.

In the TEX system the concept of global formatting, allowing for variation of visual context, is realized up to a certain point. In particular, the paragraph builder implements our paradigm to a limited extent: it evaluates a large number of variant possibilities for the visual context of the objects (characters and inter-word spaces) and globally (up to the paragraph boundary) optimizes

over them within a quite complex parameter space. For example, even if a character sequence (such as `ff`) would be typeset as a single ligature glyph in the visual context of a line of type, when it is broken (by hyphenation) across a line-boundary the visual context of the sequence is changed so that two separate glyphs are typeset. TEX's paragraph builder has recently been extended by allowing, when no sufficiently good solution is found, a 'rescaling' so that a larger region of the parameter space is searched for an optimal solution [15].

Following on from the work of Knuth, but still working under the assumption that the visual context, and thus the formatting parameters, of certain entities could be predetermined, a globally optimizing pagination algorithm was analysed and implemented by MICHAEL PLASS in his thesis [24] and such an algorithm has been incorporated in the Type & Set System developed by GRAHAM ASHER [3, 4]. Both use a two step procedure in which the first run is used to produce a galley form of the document. In this form, textual items are already composed into lines, figures, etc. and their formatting is not therefore subject to any further refinement. In the second phase these preformatted objects are used to construct the final pages. This phase is controlled by an optimizing algorithm which takes into account the "distances" from references to figures together with "grey-values" (glue-stretching) for individual pages.

Even though these systems implement considerably more quality-oriented features than most other systems, they are still severely limited in many ways: e.g., they have no explicit constructs, or rules, designed to detect and prevent 'rivers'. Also, they do not conform to our paradigm in other important areas. For example, the page-building mechanism receives paragraphs already irrevocably broken into lines with no possibility of requesting a retry to find a variant which improves the page-breaking. The mechanism used by TEX itself to discourage a hyphen at the end of the last line on a page is simply to penalize a possible page-break after this line: it does not recompose the paragraph to avoid the problem hyphen by use of a slightly different sequence of line-breaks.[1]

5 New directions

The problems involved in the automation of typesetting are numerous and none of the currently available computer programs addresses more than a small number of them. As a result, documents produced by present-day software (without a significant amount of human interaction) do not come anywhere near the standards of craft typography. (The use of TEX in conjunction with the very simplest of typographic designs is perhaps an exception to this statement.)

The suggestions we make in this section for future research are directed mainly at computer scientists. However, this does not mean that we believe that they can solve the problems by themselves—on the contrary, much collaborative work with designers and compositors is essential to this task.

Implementing in full the concept of global formatting either means optimizing the formatting of the document as a whole (i.e., storing information about *all* possible variants in the formatting of every entity) or it means emulating this process by means of an iterative process—the latter appears to be the only practical option.

In all currently available systems, formatting is implemented by procedural routines based on bottom-up concepts, i.e., larger units are constructed from smaller ones whose internal format is already fixed. To achieve higher quality in automated typesetting, the optimization part of the process cannot be confined in this way to individual layers of complexity, e.g., first all paragraphs, then all pages (as in Type & Set). Instead, each individual routine which formats a particular object needs to produce as its output a range of variant formattings (each locally optimized) for that object under one or more assumed context possibilities. For example, a paragraph formatting routine might return information such as "this paragraph could be laid out in the given context into 5 lines with a forbidden break after line 2, or into 6 lines with . . . ".

The information thus gathered can then be the input to a global optimization process which produces a new formatted view of the document with, in general, altered visual contexts for certain entities. This process should then be iterated until a fixed-point is reached, i.e., until successive iterations give the same mapping of entity and context pairs.

The specification of layout rules such as those discussed in the previous sections by means of a large collection of parameters is certainly not intuitive and requires a thorough understanding of the underlying algorithms in order to predict the results of even small changes to the values of the parameters. Moreover, this approach is defective because the only rules which are specifiable are those whose underlying concepts are present in the model and algorithms used.

In view of the nature and quantity of these rules, and hence the computational complexity of the parametric optimization problems to which they lead, it would seem sensible to investigate whether other areas of computer science can be applied to this task. In particular, we suggest the following questions.

- Can certain aspects of the rules and heuristics be effectively modelled using an expert-system approach?

[1] It is interesting to note that the TEX model for optimizing the line-breaks within a paragraph is flawed in this case since it erroneously takes into account the visual incompatibility of two neighbouring lines which end up separated by a page-break!

- Can conventional optimization algorithms be replaced by algorithms which will, with a high probability, quickly produce a near-optimal solution?

A necessary precursor to any such investigations must be the study of new conceptual models of document formatting together with accompanying description languages designed to capture the rules and heuristics in a natural manner whilst being precise enough to drive the formatting software.

We are working on the formulation and refinement of such a model of document formatting and we are studying the data-types and transformations required to implement it. However, much work needs to be done in collaboration with typographers and typographic designers in order to adequately understand the rules and heuristics of the craft which need to be built onto our model and its description language.

References

[1] James O. Achugbue. On the line breaking problem in text formatting. *Proc. of the ACM SIGPLAN/SIGOA*, 2(1,2), 1981.

[2] S. Adler, project editor. *ISO/IEC CD 10179: Information technology—Text and office systems—Document Style Semantics and Specification Language*. American National Standards Institute, New York, 1991.

[3] Graham Asher. Type & Set: TEX as the engine of a Friendly Publishing System. In Malcolm Clark, editor, *TEX applications, uses, methods*, pages 91–100. Ellis Horwood, Chichester, 1990.

[4] Graham Asher. Inside Type & Set. *TUGboat*, 13(1), (to appear).

[5] Judith Butcher. *Copy editing: the Cambridge handbook*. Cambridge University Press, Cambridge, second edition, 1981.

[6] *The Chicago Manual of Style: Rules for authors, Printers and Publishers*. University of Chicago Press, Chicago, 13th edition, 1982.

[7] Carl Dair. University of Toronto Press, Toronto, 1967. Paperback reprint 1985.

[8] David Fawthrop. *Hyphenation by algorithm of English/American and other languages*. Computer Hyphenation, Bradford, 1990.

[9] Richard Furuta, Jeffrey Scofield and Alan Shaw. Document Formatting Systems: Survey, Concepts and Issues. In [20].

[10] Bo Stig Hansen. A function-based formatting model. *Electronic Publishing*, 3(1):3–28, 1990.

[11] Sue Heinemann and Virginia Croft, editors. *Xerox Publishing Standards: A Manual of Style and Design*. Watson-Guptill, New York, 1988.

[12] Mary Anne Holstege. *Marking and the Design of Notations*. PhD thesis, Stanford University, Department of Computer Science, Stanford, CA 94305, June 1989. Report No. STAN-CS-89-1270.

[13] Allen Hurlburt. *The grid: A modular system for the design and production of newspapers, magazines, and books*. Van Nostrand Reinhold, New York, 1978.

[14] Donald E. Knuth. *The TEXbook*, volume A of *Computers & Typesetting*. Addison-Wesley, Reading, Massachusetts, May 1989. Eighth printing.

[15] Donald E. Knuth. The new versions of TEX and METAFONT. *TUGboat*, 10(3):325–328, 1989.

[16] Hans-Joachim Koppitz, editor. *Gutenberg Jahrbuch*. Gutenberg-Gesellschaft, Internationale Vereinigung für Geschichte und Gegenwart der Druckkunst e.V., Mainz. Contains results on the past and present history of the art of printing. Published since 1926.

[17] Franklin Mark Liang. *Word Hy-phen-a-tion by Com-put-er*. PhD thesis, Stanford University, Department of Computer Science, Stanford, CA 94305, August 1983. Report No. STAN-CS-83-977.

[18] Ronald McIntosh. *Hyphenation*. Computer Hyphenation, Bradford, 1990.

[19] Frank Mittelbach. E-TEX: Guidelines to future TEX extensions. In Lincoln K. Durst, editor, *1990 Conference Proceedings*, pages 337–345, September 1990. Published as TUGboat 11#3.

[20] Jurg Nievergelt, Giovanni Coray, Jean-Daniel Nicoud and Alan C. Shaw, editors. *Document Preparation Systems*. North-Holland, Amsterdam, 1982.

[21] *Hart's Rules; For Compositors and Readers at the University Press, Oxford*. Oxford University Press, London, 39th edition, 1991.

[22] Michael F. Plass and Donald E. Knuth. Breaking paragraphs into lines. *Software—Practice and Experience*, 11:1119–1184, 1981.

[23] Michael F. Plass and Donald E. Knuth. Choosing Better Line Breaks. In [20].

[24] Michael Frederick Plass. *Optimal Pagination Techniques for Automatic Typesetting Systems*. PhD thesis, Stanford University, Department of Computer Science, Stanford, CA 94305, June 1981. Report No. STAN-CS-81-970.

[25] Emil Ruder. *Typographie; Ein Gestaltungsbuch*. Niggli/Hatje, Heiden, Stuttgart, fifth revised edition, 1988. Contains complete English and French translation.

[26] Manfred Siemoneit. *Typographisches Gestalten*. Polygraph Verlag, Frankfurt am Main, second edition, 1989.

[27] Richard Southall. Presentation rules and rules of composition in the formatting of complex text. In *SGML & TEX Conference 1990, Groningen*. Abstract of talk on Page 18 of the conference program.

[28] Jan Tschichold. *Ausgewählte Aufsätze über Fragen der Gestalt des Buches*. Birkhäuser Verlag, Basel, 1987. Second printing.

[29] Jan Tschichold. *Leben und Werk des Typographen Jan Tschichold*. Saur, München; New York; London; Paris, second edition, 1988.

[30] Alex White. *How to spec type*. Watson-Guptill, New York, 1987.
[31] Jan White. *Graphic Design for the Electronic Age*. Watson-Guptill, Xerox Press, New York, 1988.
[32] Hugh Williamson. *Methods of Book Design*. Yale University Press, New Haven, London, third edition, 1983.