ELECTRONIC PUBLISHING, VOL. 8(2 & 3), 81–93 (JUNE & SEPTEMBER 1995)

Typographic sheets and structured documents

HÉLÈNE RICHY AND JACQUES ANDRÉ

Irisa
Campus de Beaulieu
F-35042 Rennes Cedex, France

e-mail: Helene.Richy@irisa.fr, Jacques.Andre@irisa.fr

SUMMARY

Document structure provides facilities for accessing and presenting hypermedia documents, and style sheets support their layout on screen (or paper). However, little attention is given to the quality of these documents in terms of typography, in the sense, for example, of abiding by the rules of the Chicago Manual of Style. This paper shows how the specification of typographic sheets can help apply typographic control to structured documents. This approach provides a mapping between elements and typographic properties. A typographic checker based on typographic sheets is presented.

KEY WORDS Typography Correctness Quality Sheets Structured documents SGML Thot

1 INTRODUCTION

For centuries, authors have been obliged to go through the 'Gutenberg galaxy' to be published. Today, thanks to electronic documents and world-wide networks, authors and readers communicate directly. Alas, this is quite often done without the knowledge or the savoir-faire underlying the traditional activities of typographers, editors, proofreaders, and printers. Structured documents offer an alternative to this situation by separating concerns. A typical example is the definition of the graphical aspects of a document in so-called style sheets, layout style, etc., allowing the author to focus on the logical contents of his/her document.

However typography is not limited to the graphical (visual) aspects of the layout. In fact, this is only one of three levels:

1. The lower level is concerned with characters and is usually called *digital typography*. It describes how the characters are individually defined and rendered depending on the device (e.g. the use of *greyscale* for characters on screen) and on the size (optical scaling), and how these characters are grouped as fonts [1].
2. The upper level is concerned with the layout, the graphical presentation, and the template. This level is sometimes called *macro-typography* and refers to 'visual quality', a matter of esthetic judgement and legibility. The main goal is to insure a graphical

CCC 0894–3982/95/020081–13

Received 29 March 1996
Revised 17 June 1996

consistency while assembling all pieces of text (margin, titles, notes, bibliography, index) and setting all graphical elements (illustrations, figures). Traditional WYSIWYG composition systems attempt to provide for this visual representation through user control, whereas style sheets [2,3] offer another way to control layout.

3. An intermediate level concerns the text itself, not its layout. It works on the juxtaposition of letters, signs (such as punctuation signs) and words. This level is sometimes called *micro-typography*. The control of micro-typography consists of checking grammatical mistakes (syntactical or spelling errors, errors in the use of punctuation, capital letters, parentheses, ...), and typographic errors (misuse of spaces or character variants such as italic or roman). Checking micro-typography correctness requires not only typographic knowledge, but also knowledge of linguistics and an understanding of the text. Proofreaders work at this level.

In the following we are only interested in this intermediate level. For simplicity, we will call *typography* all the rules that are relevant to micro-typography.

Various studies about vision [1,4], legibility [5,6], and 'textual anatomy' [7,8] show that defining typographic rules is a very complex question. We believe that by referring to the know-how of professionals, a typographic checker for electronic documents can be developed. Thus, the aims of our work are to draw on the main concepts of typography to specify the typographic checking of electronic documents. Without expecting to replace human proofreaders, we have developed an easy to use *typo checker* which will improve the typographic style of documents. The ease of use relies on the modifiability of the rules and their adaptability to various styles of documents.

Our approach for checking structured documents is based on what we call *typographic sheets*. Just as a style sheet specifies formatting and layout characteristics (defining boxes as in TEX [9]) mapped with elements and attributes, a typographic sheet[1] specifies typographic characteristics mapped with elements and attributes. The purpose of typographic sheets is to provide:

- a consistent interpretation of the typographic rules,
- a basis for implementation of a typographic checker,
- a reference for improving and extending existing style sheets.

This paper[2] discusses the current state of development of our typo checker for structured documents. Analysis of typographic usage leads us to propose (in the next section) classification for the typographic rules. A language based on these classifications is used for defining *typo sheets* of SGML (Standard Generalized Markup Language [10]) documents, and incidentally HTML (Hypertext Markup Language [11]) documents. An overview of this language is presented in section 3. Section 4 explains the method of checking the typography using typo sheets and describes the typo checker currently available in Thot, an interactive editor derived from Grif [12].

[1] Typo sheets are used by a typographic checker (section 4), a specific tool not involved during the formatting process.

[2] We did not process our paper through this typo checker itself because its publication requirements entailed LATEX, for which we do not have an implementation. However, a first draft was checked using the typographic checker in Thot.

2 CLASSIFICATION OF TYPOGRAPHIC RULES

First, let us look at traditional typographic rules. They are published in books like *The Chicago Manual of Style* [13], or the various French *Codes typographiques* [14–16]. Some in-house publishing rules remain unpublished. Most style manuals classify typographic rules depending on the nature of the signs that are used. This classification is summarized in section 2.1. However, many rules also depend on various concepts of document structure as we describe them in section 2.2.

2.1 Classification by nature and signs

The manuals do not agree about all the rules[3], but most of them classify typographic rules as follows:

Punctuation Punctuation is governed by its function. For instance, 'close' punctuation refers to the grammatical structure of the material (sentences). Punctuation may also be governed by the logical structure (vertical list, title page, heading, . . .).

Abbreviations and acronyms Aside from publications in science and technology, abbreviations only appear in tables, notes, or bibliography. It is often an open question whether or not periods should be used with particular abbreviations. When in doubt, authoritative lists of abbreviations are consulted (e.g. in standard dictionaries).

Capitalization and names Patterns of capitalization for various categories (names, titles, offices, geographical terms, . . .) are established for helping in the use of names and of terms associated with names: several famous dictionaries are considered as accurate sources for the capitalization and for algorithms.

Distinctive treatments of words Italics, capital letters, quotation marks, . . . may be used to achieve special effects—emphasis, irony or whatever—or for isolating words in a foreign language.

Quotations Quotations may be incorporated in two ways. Whether to run in or set off a quotation is commonly determined by its length. In American practice, single quotation marks enclose quotations within quotations. British practice is often the reverse.

Numbers Numbers may be spelled out or composed of numerals, either in arabic or in roman faces depending on the context. It is difficult to be entirely consistent in the treatment of numbers in textual matter.

Hyphenation and word division When justified lines are required, some words will be hyphenated at the ends of lines. Logical hyphenation is provided by most desktop publishing systems (see algorithms in [17]).

[3] Rules depend on languages (for example, French printers conventionally put a thin space between a word and the subsequent semi-column; while English ones don't), national usage varies (even for a given language, say English, usage varies from one country to another one — as is frequently pointed out in the netnews group comp.fonts) and even from one publisher to another one. Note that consistency is required whatever the rules are.

2.2 Classification by context

Many rules described in the previous section depend heavily on the structural, syntactic, semantic, linguistic or other context. For instance:

Logical structure Some rules only apply to a specific type of elements. For example, the period is omitted after centered headings, signatures or legends.

Syntactical structure Some rules are governed by the syntactic structure of sentences. For instance, a capital letter is used at the beginning of a sentence. Nevertheless, unusual but intended punctuation and capitalization may contribute to the author's style.

Semantics Some rules are governed by the nature or the semantics of words. For example, the titles of published works are conventionally set in italic type (*The Chicago Manual of Style);* full names, and often the shortened names, of legislative or administrative departments, bureaus, and offices are capitalized: *parliament*, but *Parliament of Paris*, etc.

Language Some rules are governed by the language: usage of spaces around punctuation and use of these punctuations in the sentences, word hyphenation, etc.

3 LANGUAGE FOR SPECIFYING TYPOGRAPHIC SHEETS

Our language for specifying typographic sheets is based on the classification described in section 2. Some of the contexts are easy to identify when using an editing system: the analysis of the structural context is obvious when using a structured authoring environment; spelling checkers used in the authoring environment identify the language when considering multilingual documents (the language is a characteristic of the paragraph style or related to attributes as in SGML [10], LaTeX [18], or Thot [12]). However, identification of other contexts is not so easy and may only be solved when using linguistic tools or a syntactical analyzer [8].

For easy application on structured documents, typographic rules are first considered as properties mapped with structured elements or attributes. The other contexts are specified in conditional rules. For example, if a different capitalization rule applies to a French title and to an English title, these two punctuation rules are mapped with the title element and a condition on the language is applied to this rule.

Considering the generic structures of documents, we propose here a generic approach for typography: a *typo sheet* is defined for a class of documents. A typo sheet is composed of the set of all the typographic properties mapped with elements as defined in a generic structure.

Before describing this typographic language, a short review of the generic approach to structured documents is given below.

3.1 Typo sheets for structured documents

A *structured* document [19] consists of a document whose content is organized as a logical structure comprising different types of elements such as titles, paragraphs, sections, chapters, figures, etc. Some elements may have attributes (language, emphasis, etc.) describing a specific semantics within the document. For instance, the attribute *language* when applied to an element indicates in which language the corresponding text is written. Attributes may be applied to any kind of elements: a word, a section, or a whole chapter as well.

The generic approach (ODA [20], SGML [10], Thot [12]) allows the definition of typed elements and provides for their organization within a document: the generic logical structure is called a Document Type Description (DTD) in SGML. Documents with the same generic logical structure are considered as belonging to the same class of documents.

We consider that:

- A typo sheet holds all the typographic rules available for elements of documents of the same class. This feature enables a homogeneous application of the typographic rules within a document depending on the types of elements, on the position of the elements in the logical structure of the document and on the attributes which may be related to them.
- The use of the same typo sheet for checking several documents ensures consistency and homogeneity of the documents.
- Various typo sheets may be specified, reflecting, for instance, differing typographic requirements for technical documents and for philosophy books, for French papers and for English publications.
- Because typography is changeable and not standarized, the typo sheet can easily be updated to follow new styles of typography.

Thus a typo checker that relies upon various typo sheets ensures flexible control.

3.2 Overview of the language

Our typographic language is a declarative language (see the syntax in the appendix). This language is used to write typo sheets which define the typographic rules to be applied to different classes of documents: rules are associated to elements and attributes defined in a DTD. These rules only apply to textual elements which require special typographic usage.

As typographic rules may depend on various contexts (see 2.2), our typographic language allows specification of various kinds of conditions within rules:

- depending on the position of the element within the document structure: on the type of the enclosing element, of the next, or of the previous element,
- depending on the language (English, American, German, French, etc.),
- depending on the lexical unit (proper names, acronyms, etc.).

A rule can be defined at any level in the document structure. Inheritance allows rules which are associated with an element to apply to all nested elements until the terminal level (textual elements). However, some typographic functions called *alinea functions* do not apply to every textual element, but only to some of them when considered as *alinea*[4]. Thus, the language allows the definition of alinea elements for identifying a class of elements which are concerned with alinea functions: only elements which have been defined as *alinea* will be checked by these functions. For example, a rule may require that an element ends with a special punctuation mark only applies to *alinea*, i.e. only to the last textual unit that this element may contain.

A typographic rule invokes *control functions*, each of which belongs to a category (as identified in section 2.1): capitalization, abbreviation, punctuation, ... The classification

[4] *alinea* are paragraphs. The word *paragraph* is not used here because it is overridden by a restricted meaning in most document models.

of functions into categories offers several advantages. First, it allows limitation of inheritance to functions belonging to the same category. For instance, the definition of a function belonging to the category 'punctuation' on an element enables inheritance of any function belonging to this category that could be inherited from an upper level. A second advantage concerns the running of a typo checker. The categories permit judicious ordering of the evaluation of rules, e.g., checking abbreviations before checking punctuation. A last advantage concerns the checking capabilities. Based on these categories, a typo checker will be able to propose partial verification of typographic rules. The category can be considered as a filter allowing extraction of rules from an existing typo sheet. For instance, a user who wants to check the use of capitals will only indicate *Capital* and not the various functions that might otherwise be invoked according to the specific style sheet. Any number of functions can be created and referenced in a typo sheet and the naming of functions is without restriction. However, the categories of functions are fixed by the grammar of the language[5]. Thanks to this feature, a typo checker can easily be extended when developing new control programs associated to new functions[6].

3.3 Content of a typo sheet

As defined in the grammar of the language (see appendix), a typo sheet is composed of four sections (see in Figure 1, an example extracted from a typo sheet for HTML documents). The first section is a list of *alinea* elements (ALINEA). In the second section (COMPOSITION), it is possible to define *composition models* as sets of typographic rules that may be reused as typographic rules either on elements or on attributes. The typographic rules applying on elements (ELEMENT) and attributes (ATTR) are defined in the two last sections. Some of them may refer to composition models which have been previously defined in the second section.

A typographic language may provide the following categories of control functions:

- *Abbreviation*: use of abbreviations,
- *Attribute*: use of attributes and presentation,
- *Capital*: use of capitals,
- *Distance*: spacing between elements,
- *Exponent*: use of superscripts,
- *InsertPair*: quotations and parentheses,
- *Punctuation*: punctuation (appearance and position),
- *SpaceTable*: spacing between characters.
- *Word*: typography of words (capitals, italics, . . .).

Several functions may be defined in each category. For instance, in the category *Distance*, relevant functions for checking spacing are listed below:

- `DistAlineaNone` may check no final space left,
- `DistNextHard` may check 'hard' (em-) space after the element,
- `DistPrevSoft` may check 'soft' space before the element,
- `DistNextNone` may check no space between elements,
- `DistPrevPunctPar` may check the relative position of a final punctuation and an element such as reference to a note, etc.

[5] In the first implementation within Thot nine categories are implemented.

[6] The use of an interpreted language (such as Tcl) for writing such functions will allow a faster development and experimentation with new typographic functions.

```
TYPOGRAPHY HTML
ALINEA
  H1, H2, H3, H4, H5, H6, Toggle_Item;
  ...
COMPOSITION
  HeadingMod =
    BEGIN
    IF (Language = English) Capital: CapAllWord;
    IF (Language = Francais) Capital: CapFirstElem;
    Punctuation: PunctHighNone;
    END;
   ...
ELEMENT
  Form:
    BEGIN
    IF (IS Acronym) Word: CapitAll;
    InsertPair: PairNone;
    END;
  H1: HeadingMod;
  Toggle_Item:
    BEGIN
    Capital: CapFirstElem;
    Punctuation: PunctHighNone;
    END;
  Paragraph:
    BEGIN
    IF (IN Form AND Next = Toggle_Menu)
      Punctuation: PunctColon;
    IF (IN Form AND NOT Next = Toggle_Menu)
      Punctuation: PunctNone;
    IF (NOT IN Form) Punctuation: PunctEnd;
      InsertPair: FunctNil;
    END;
   ...
ATTR
   Default_Value (Submit_Input):
     Capital: CapFirstElem;
 ...
END.
```

Figure 1. A typo sheet for an HTML Form (extract)

The syntax of typographic rules is identical whether they are associated to elements or to attributes. A global *null* rule — available for any category of function (`FunctNil` in Figure 1) — can be used to supress inheritance without defining a new rule on the current element. The rules may be either imperative or conditional and a composition model can be used as a typographic rule. A composition model associates a name to a set of typographic rules. For instance, a composition model called `HeadingMod` can be defined and reused for defining the typographic rules associated with H1 element (or H2 element, as well). Various conditional rules may be specified: rules depending on the identified lexical unit, on the position in the logical structure or on the language. The rules to be applied on attributes may apply either for any value of the attribute or only for special values. In the latter case, one typographic rule may be written for each value of the attribute.

A special treatment concerns the attribute *Language*. The *Language* attribute can be involved in conditional rules by means of a *Language* condition. Thus, when using conditional rules depending on the *Language* attribute value, only one typo sheet is used for checking multilingual documents. Nevertheless, it is possible to define a typo sheet for each language and each style of document, if wanted.

4 IMPLEMENTATION OF A TYPO CHECKER

The language for typography described above has been implemented and used for writing typo sheets for structured documents. This section describes a typo checker based on the typo sheets and developed in Thot (more details are given in [21]). This typo checker detects typographic errors such as capitalization or punctuation errors within structured documents. In the same way as a spelling checker does [22], the typo checker analyses the text and detects errors concerning micro-typography within the textual parts of the document. This checker may also suggest a list of corrections to help users recover from errors. A generic approach has been adopted. This approach consists of describing homogeneous sets of typographic properties related to a document class [12].

4.1 Specification of the typo sheets

A typo sheet specifies typo characteristics mapped with elements and attributes that are defined in a DTD. Several style sheets and several typo sheets may be defined for the same class of documents (the same DTD). It is generally useful to define various typo sheets for different classes of documents: for example, use of punctuation changes when considering a *technical report* or when considering a *set of conference slides*. In order to avoid proliferation of typo sheets for a class of document, conditional typographic rules are defined. For example, conditional rules depending on the language attribute are defined for checking multilingual documents.

Each control function[7] searches for one category of typographic error within a text: a missing capital, a misused punctuation, a misused abbreviation, etc. Each function is composed of a detection part and of a correction part.

[7] Control functions are written in the C language. A set of procedures (for selection, search or correction) is provided to help writing new functions. The ability to define new functions in TCL [23] and to write them directly in the typography models is planned.

4.2 Operating mode and user interface

The typo checker works as follows: considering a structured document, it first reads the typo sheet(s) related to the class of this document. It then checks the rules issued from these sheets, element by element, using any inherited functions and functions invoked based on attributes. When an attribute is defined, the typographic rules associated with this attribute override the others.

The typo checker works in the same way as the Thot spelling checker does [22]. The checking process is interactive. A dialog between the human proofreader and the typo checker supports resolution of ambiguities. All text belonging to the *checking area* of the document is checked. Each control function is executed one by one on the current leaf (text only)—in the logical tree structure of the document. If no typographic error is detected, the next text leaf is checked until the entire checking area has been examined.

A dialog form displayed in a window on the screen allows the user to define the checking area (before or after the current selection, within the selection or in the whole document). When a typographic error is detected, the suspect text is highlighted and displayed on the screen within the document window. A special message in the dialog window indicates the kind of error that has been detected. If the option 'correction on demand' is selected, the dialogue indicates which kind of correction is wanted and displays the related proposals (see Figure 2).

At present the typo checker can propose the following corrections:

- addition or suppression of a character (punctuation, parenthesis, apostrophe, space, etc.)
- replacement of a character or a word by another character or word (punctuation, abbreviation, or acronym, etc.)
- insertion or suppression of an attribute (italics, bold, etc.)
- moving a character (a period or a parenthesis) before a reference,
- capitalization (caps or small caps) or lowercase.

The user can either confirm, stop, or ask for explanation. It is also possible to modify the text directly within the document window or continue searching for the next error.

In addition, an option window (see Figure 3) gives options for controlling the checking process: categories of control functions to be checked, typo sheet to be used, types of elements or types of characters to be checked.

4.3 Initial results

The success of a typo checker based on typo sheets depends on a set of features:

- quality typo sheets: whether, for a particular category and function, inheritance should be specified or explicitly avoided.
- efficient control functions: it may be more efficient to gather several functions than to split them; for example, for checking small caps and initial caps, it will be better to use only one function that does both, and thus propose only one correction.
- accurate sources for recognition of lexical units: for example, lexicons of historical and cultural terms when working on historical documents, or lexicons of scientific terminology (plants, animals, geological terms, astronomical terms, etc.) when checking scientific documents for capitalization of names.

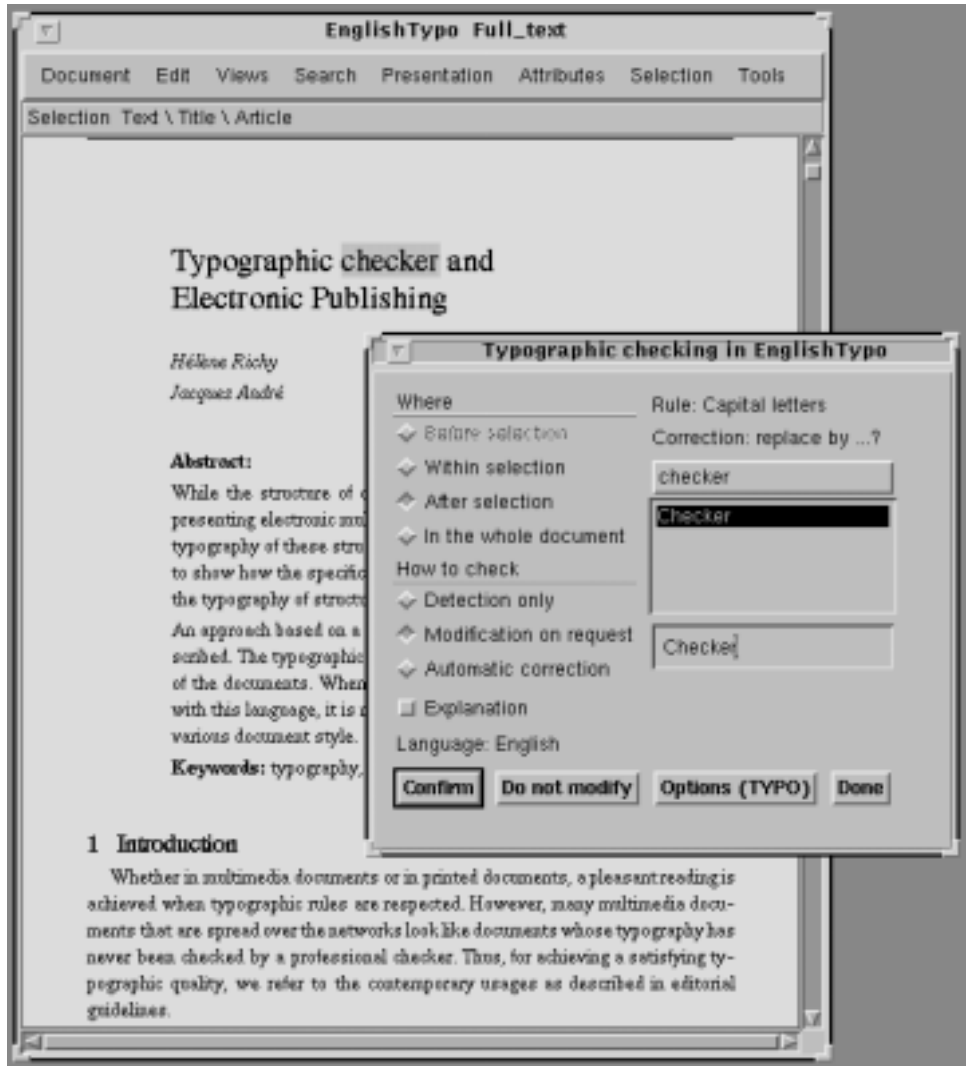

Figure 2. Dialog window of the typo checker within the Thot authoring environment. (Here a capital is proposed for the word 'checker' because it is in a title.)

The first experiments are very promising. We found that a checking process based only on structural and on some linguistic considerations detected many typographic errors and that the proposed corrections were pertinent. See footnote 2.

Some rules, such as the usage of spacing, could be checked using the automatic correction mode. However, we found it inadvisable to ask for automatic correction of all typographic rules: because the user may better 'understand' the document content than the typo checker does, the user can resolve any ambiguities by a dialogue with the typographic checker as provided by Thot. For example, ambiguities could be detected when checking capitals or abbreviations, but their resolution often requires an understanding of the content. Despite these successes, a precise measure of the results is difficult because there is no

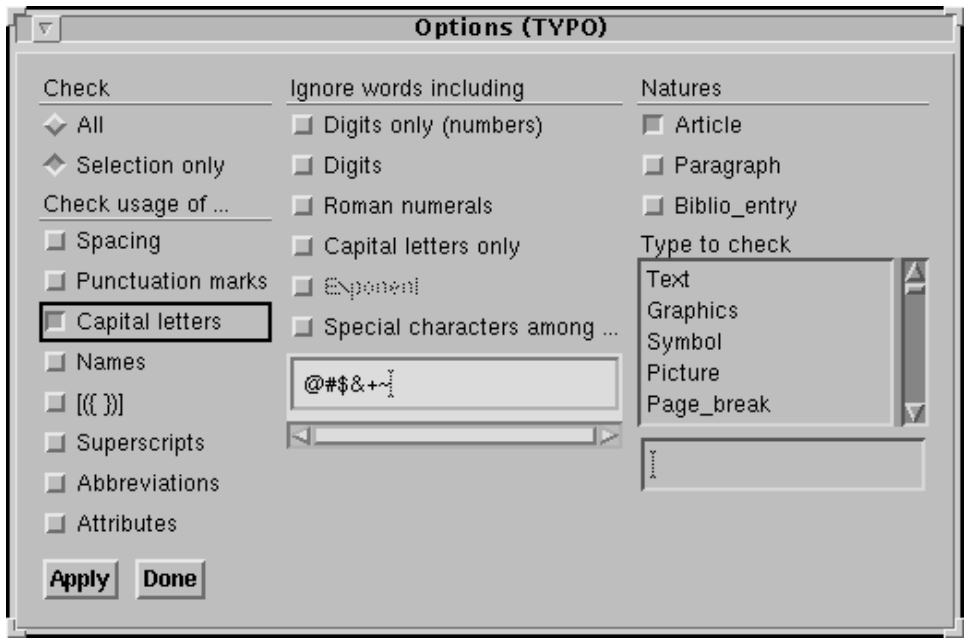

Figure 3. Option window for the typo checker

'standard' set of erroneous documents for testing and many corrections are a matter of taste.

In short, the use of the typo checker insures the homogeneity of documents as to the application of the typographic rules. Further studies and experiments for validating the typo checker are required, particularly for preventing some rules from interfering with each other. The integration of this typo checker within Tamaya [24] a structured authoring environment for the Web, based on Thot, is under consideration. Such a tool will provide easy checking of the typography correctness of HTML documents.

5 CONCLUSION

Desktop publishing and editing systems still have much to learn from the experience of professional editors and proofreaders who have spent many years improving the art of printing. Users are far from having gained such lengthy experience, and few of them feel concerned with pleasing typography either for printed documents, or for text appearing on displays. Because many users ignore typographic rules, a tool such as our typo checker is welcome, especially when documents are distributed in electronic form, without guidelines from professional editors.

ACKNOWLEDGEMENTS

The authors would like to thank Jeanine Grimault for her experience in proofreading and her help in understanding typographic rules, the anonymous reviewers and especially Robert A. Morris for his comments and his careful review of this paper and for his editorial assistance. Merci!

REFERENCES

1. *Visual and Technical Aspects of type*, ed., R. Hersch, Cambridge University Press, Cambridge, UK, 1993.
2. H. Lie and B. Bos. Cascading Style Sheets, level 1, W3C Working Draft, May 1996. http://www.w3.org/pub/WWW/TR/WD-css1.html.
3. ISO, *Information technology – Text and office systems – Document Style Semantics and Specification Language (DSSSL)*, 1991. ISO/IEC DIS 10179.
4. G. Legge, 'Psychophysics of reading', *Vision research*, 239–252, (1985).
5. R. A. Morris, K. Berry, and K. Hargreaves, 'Towards quantification of the effects of typographic variation on readability', *SID93 Digest*, (1993).
6. M. Tinker, *Legibility of Print*, Iowa State University Press, USA, 1963.
7. P. Norrish, 'Semantic structures of text', in *Structured documents*, eds., J. André, R. Furuta, and V. Quint, pp. 143–159, Cambridge, UK, (1989). Cambridge University Press.
8. J. Virbel, 'The Contribution of Linguistic Knowledge to the Interpretation of Text Structures', in *Structured Documents*, eds., J. André, R. Furuta, and V. Quint, pp. 161–180, Cambridge, UK, (1989). Cambridge University Press.
9. D. E. Knuth, *The TeXbook*, Addison-Wesley, Reading, Massachusetts, USA, 1984.
10. ISO, *Information processing – Text and office systems – Standard Generalized Markup Language (SGML)*, October 1986. ISO 8879–1986(E).
11. T. Berners-Lee and D. Connoly. Hypertext Markup Language - 2.0, November 1995. ftp://ds.internic.net/rfc/rfc1866.txt.
12. V. Quint and I. Vatton, 'Grif: an Interactive System for Structured Document manipulation', in *Text Processing and Document Manipulation*, ed., J. C. van Vliet, pp. 200–213, Cambridge, UK, (1986). Cambridge University Press.
13. *The Chicago Manual of Style*, The University of Chicago Press, Chicago, Illinois, USA, 1993.
14. *Lexique des règles typographiques en usage à l'Imprimerie nationale*, Imprimerie nationale, France, October 1990.
15. *Code typographique*, Fédération nationale du personnel d'encadrement des industries polygraphiques et de la communication, Paris, France, 1981.
16. *Guide du typographe romand*, Groupe de Lausanne de l'Association suisse des compositeurs à la machine, Switzerland, 1994.
17. F. M. Liang, *Word Hy-phen-a-tion by computer*, Ph.D. dissertation, Standford University, USA, June 1983.
18. L. Lamport, LaTeX, *A Document Preparation System*, Addison-Wesley, Reading, Massachusetts, USA, 1986.
19. R. Furuta, V. Quint, and J. André, 'Interactively Editing Structured Documents', *Electronic Publishing-Origination, Dissemination, and Design*, **1**(1), 19–44, (April 1988).
20. ISO, *Information processing – Text and office systems – Office Document Architecture (ODA)*, 1989. ISO 8613.
21. H. Richy and J. André, 'Correcteur typographique pour l'édition électronique', Research report 2562, Inria, Irisa, Campus universitaire de Beaulieu, 35042 Rennes Cedex (France), (April 1995). [Richy95a] at http://opera.inrialpes.fr/OPERA/BibOpera.html.
22. H. Richy, P. Frison, and É. Picheral, 'Multilingual String-to-String Correction in Grif, a structured editor', in *EP'92*, eds., C. Vanoirbeek and G. Coray, pp. 183–198, Cambridge, UK, (1992). Cambridge University Press. [Richy92a] at http://opera.inrialpes.fr/OPERA/BibOpera.html.
23. J. K. Ousterhout, 'Tcl: An Embeddable Command Language', in *USENIX Conference*, (Winter 1990).
24. V. Quint, C. Roisin, and I. Vatton, 'A structured authoring environment for the World-Wide Web', *Computer Networks and ISDN Systems*, **27**(6), 831–840, (April 1995). [Quint95b] at http://opera.inrialpes.fr/OPERA/BibOpera.html.
25. V. Quint, 'The Languages of Thot', Int. Report, Opera project, Imag, Grenoble, France, (December 1995). Translated by E. Munson, http://opera.inrialpes.fr/OPERA/BibOpera.html.

APPENDIX: Grammar of the typo language

This appendix gives the syntax of the grammar of the typo language of Thot. It is described here in the M meta-language [25].

```
TypoSheet = 'TYPOGRAPHY' IdentType ';' [ 'ALINEA' SeqOfType ]
           [ 'COMPOSITION' SeqOfMod ] [ 'ELEMENT' SeqOfTypo ]
           [ 'ATTR' SeqOfTypoAttr ] 'END' .
SeqOfType = IdentType < ',' IdentType > ';' .
TypeFunct = 'Abbreviation' /'Attribute' /'Capital' /'Distance' /
            'Exponent'/'InsertPair' /'Punctuation' /'SpaceTable'/
            'Word' .
IdentFunct = NAME .
TypoRule = [ 'IF' '(' SeqOfCondition ')' ] SeqOfParam ';'.
SeqOfParam = 'BEGIN' ParamTypo < ParamTypo > 'END' ';'/ParamTypo .
ParamTypo = TypeFunct ':' IdFunct / IdentMod .
IdFunct = IdentFunct [ '=' DefFunct ] / 'FunctNil' .
DefFunct = 'BEGIN' CmdTcl 'END'.
CmdTcl = STRING .
SeqOfCondition = Condition < 'AND' Condition > .
Condition = [ 'NOT' ] ConditionElem .
ConditionElem = 'First' / 'Last' / RelType OpType IdentType /
                'IN' IdentType / 'LANGUAGE' '=' Language /
                'IS' LexicalUnit .
OpType = 'IN' / '=' .
RelType = 'Next' / 'Previous' .
LexicalUnit = NAME .
Language = NAME .
SeqOfMod = Mod < Mod > .
Mod = IdentMod '=' SeqOfRules .
IdentMod = NAME .
SeqOfRules = 'BEGIN' TypoRule < TypoRule > 'END' ';' / TypoRule .
SeqOfTypo = Typo < Typo > .
Typo = IdentType ':' SeqOfRules .
SeqOfTypoAttr = TransAttr < TransAttr > .
TransAttr = AttrIdent [ '(' IdentType ')' ] [ AttrRelat ] ':'
            SeqOfRules .
AttrIdent = NAME .
AttrRelat = '=' AttrValue/ '>' [ '-' ] MinVal/ '<' [ '-' ] MaxVal/
            'IN' '[' ['-'] MinInterval '..' ['-'] MaxInterval']'.
AttrValue = [ '-' ] ValEqual / TextEqual / AttrValIdent .
MinVal = NUMBER .
MaxVal = NUMBER .
MinInterval = NUMBER .
MaxInterval = NUMBER .
ValEqual = NUMBER .
TextEqual = STRING .
AttrValIdent = NAME .
IdentType = NAME .
```