JÖRGEN PIND

Computers, Typesetting, and Lexicography

Abstract

As part of the general strategy of computerizing the lexicographic work process at the Institute of Lexicography, we have adopted Donald E. Knuths typesetting program TEX as our typesetting engine. The main characteristics of the program will be briefly described, followed by a discussion of its advantages for lexicographic work.

TEX has already been used for the typesetting of a 1300 page etymological dictionary of Icelandic. A number of other projects are under way.

Special notice will be paid to the problem of coding as it relates to the making of dictionaries. The advantages of a generic, or logical, coding over typographic coding will be emphasized. However, doubts will be raised about the possibility of providing a set of tags which are completely neutral with respect to typographic considerations.

1 Introduction

In this paper I want to discuss one particular aspect of computational lexicography, namely the typesetting of dictionaries. This is perhaps not an issue which is central to computational lexicography, yet it is a subject which deserves study, especially now when the arts of typesetting have been moving onto the desktop. I will show you the approach we have adopted at the Institute of Lexicography, and remark on how it fits into our overall strategy for computational lexicography.

Let me begin, in all modesty, by quoting myself. In 1986 I was invited to give a talk at the NordData Conference in Stockholm. At that time we were just embarking on widespread use of computers at the Institute, and I attempted to draw up a schematic diagram of a 'Lexicographers' workbench' (see figure 1), commenting that a number of features had not been implemented. "This holds especially for the 'manuscript writer'. Our work has not yet reached the stage where this is in great demand, but we envisage the possibility of using the database to turn out manuscripts for a typesetting program like TEX." (Pind 1986:87).

Well, this was written before we even had a version of TEX running at the Institute! As a matter of fact, though we expected that typesetting would be

Proceedings of NODALIDA 1989, pages 308-325

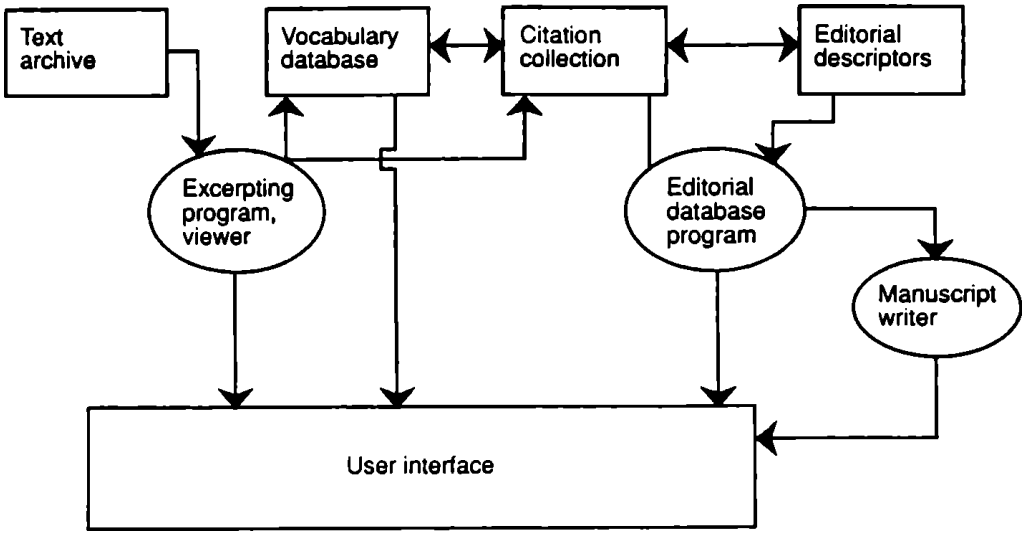

Figure 1: The lexicographer's workbench, 1986 vintage.

something that we would deal with much later, a lot of work over the past couple of years has been devoted to the typesetting side of lexicography.

There are two major reasons for this. The first is that the editor wants to be able to print proofs which are as closely related to the final form of the dictionary as possible. Thus a 'manuscript writer' has in fact been implemented as a feature of the 'workbench' we are currently working on. The relationship of this 'manuscript writer' to the work on the verbal dictionary has already been touched on in the paper by Björn Þór Svavarsson and Jörgen Pind in this volume.

The second reason is the fact that we have been engaged in producing Icelandic dictionaries from manuscripts, rather than from a database. Foremost among these is an Icelandic etymological dictionary by the late Ásgeir Blöndal Magnússon, former editor at the Institute, which will appear later this year.[1] We have also embarked upon a series of reprints of older Icelandic lexicographic works. These works have been coded in the TEX typesetting language.

2 Named Categories and Visual Formatting

In recent years a revolution has been taking place in the typesetting industry where numerous 'desktop publishing' programs have gradually been replacing the traditional tools of the printer. How far has this revolution affected the

[1] As a matter of fact, it was published on the 2nd of November 1989 as planned.

dictionary publisher and maker? I want to argue that such systems are not suited for the making of dictionaries.

Traditionally, dictionaries have been produced from collections of slips which have been used to ease the task of keeping the dictionary entries in alphabetical order and to allow them to expand as needed, without unduly affecting entries which follow alphabetically. The slips have then often been used, with minimal markup, as the manuscript for the printer. In the past few years attempts have, however, been made to use database systems to ease the arduous task of handling the collections of slips, with some success (cf. the paper by Björn Þór Svavarsson and Jörgen Pind in this volume). If we consider for a moment the nature of the database system, it is obvious that one of its major strengths is the fact that it allows the user to assign names or tags to the individual fields in the database. Thus we can easily imagine a database system for lexicographic work which knows about categories such as *headword*, *pronunciation*, *grammatical code*, *semantic field*, *usage notes*, and so on.

One of the typographical requirements for a dictionary is that *some* of these categories should be reflected in the typesetting itself. This shows for example in the use of different fonts in dictionaries, typically used to distinguish some of the categories. Note that only some of the categories will be thus reflected, since a typical dictionary contains many more categories than would be distinguished by typographic means. Some distinctions will thus be lost in the printed dictionary which are kept in the database systems.

Ideally, the lexicographer would like to use the database to automatically generate 'scripts' for typesetting, simply by instructing the database to print relevant typographic codes around some of the fields and not others. An even better approach would be to tag all the categories in the typesetting script and then instruct the typesetting system as to which ones should affect the typesetting process and which ones should not be reflected typographically. This latter approach is easy enough to accomplish if the typesetting system allows 'generic' or abstract coding of the input.

The desktop publishing systems mentioned at the beginning of this section do not allow such abstract coding (indeed very few of them are able to deal with traditional typesetting codes), since they are almost universally based on the idea of 'direct manipulation' or 'visual formatting'. The user manipulates a pointing device, such as a mouse, to mark parts of the text for, say, a font change. The notion of abstract coding plays no part at all in the formatting, and thus it is impossible in such a system to form a link between the categories of the database system and the typesetting. However, this is, of course, of the utmost importance for the lexicographer. A priori, I would have thought that this limitation of the desktop publishing systems would rule them out as being suitable for lexicographic work, and I was thus rather surprised when I came across the following description of the approach taken at the dictionary of Old English in Toronto.

> The typographical complexity of the dictionary entries—with a number of special characters, several languages, and many subsec-

> tions and cross-references which are distinguished by type—emphasizes the importance of interactive formatting. Because the working copy of the entry on the screen depicts the final appearance of a page, we hope to improve consistency...
>
> ...For example to put a keyword in bold in a citation, an editor can activate the area to be formatted by ranging over it with the mouse, and then use the mouse to select and apply the property bold from the Character Looks Menu (Healy 1985:248).

The system being described is a Xerox workstation, running publishing software similar to programs running on the Macintosh computer.

As mentioned earlier, this approach is severely handicapped by the fact that there is no easy way in a visual formatting system to form links to the categories of the database system being used. I would therefore like to argue that the requirements which need to be made of a typesetting system for lexicographic work are twofold.

- The typography should be of the highest order.
- The system must be able to work with generic or logical markup.

These requirements are met by a number of systems. We have chosen to work with TEX. In the following pages I will describe the way we have used TEX. While some of you are undoubtedly familiar with TEX, I will presume that not everyone is, and ask those knowledgeable to bear with me while I give a short tutorial introduction to TEX.

3 What is TEX? A Tutorial Introduction

TEX is a typesetting system 'intented for the creation of beautiful books' to quote TEX's author, Professor Donald E. Knuth of Stanford University. Those who have read his *TEXbook* will also know that the previous quote continues with 'and especially books that contain a lot of mathematics'.

TEX is indeed the premier system for typesetting mathematics available in the world today, so it is perhaps somewhat surprising to find it used for the making of dictionaries, indeed dictionaries which contain *no* mathematics at all! I will attempt to describe why we have found TEX to be eminently suitable for the typesetting of our dictionaries.

3.1 The Beginnings of TEX

It is perhaps rather surprising that we should be able to use TEX at all considering that it was created for one express purpose, viz. to allow Don Knuth to typeset his own magisterial treatise on the *Art of Programming* in what he felt would be an acceptable manner. These books started out being typeset in lead in the time-honoured manner of many generations of printers. When subsequently revisions of the original volumes were being prepared, the computer had made inroads into the field of typesetting and, to quote Knuth,

> ... when I received galley proofs they looked awful—because printing technology had changed drastically since the first edition had been published. The books where now done with phototypesetting instead of hot lead Monotype machines; and (alas!) they were being done with the help of computers instead of by hand (Knuth 1986f:96).

This was in 1977. This lead Knuth to temporarily abandon the project of writing the *Art of Computer Programming* while he would make up his own system for the typesetting, a task which he estimated would take about one year. In fact it took nine years of concentrated work to finish TEX and it's companion program METAFONT, which is a system for generations of letterforms.

The source code for the TEX system has graciously been put in the public domain by Knuth. The programs are written in WEB which is a special system for 'literate programming' (Knuth 1984b). A WEB program is processed by two programs. TANGLE makes a Pascal program from the WEB source which can then be compiled by a Pascal compiler, while WEAVE makes a TEX script from the same source, containing the source code with comments and detailed indices. Running this script through TEX produces a typeset version of the program. Knuth has thoroughly documented the TEX and METAFONT programs in his five volume work *Computers and Typesetting* (Knuth 1986a-e).

3.2 The Nature of TEX

TEX can be described as a document compiler or a typesetting language. Both terms require some clarification.

In the history of computer science, many computer languages have evolved. Some of these have been general purpose languages like Pascal or C, others have been specifically crafted for some particular task. TEX is an example of a special purpose language, and so is METAFONT. TEX as a language has primitive constructs which relate to the traditional art of printing.

The objects which TEX handles are 'boxes' and 'glue', to use Knuth's terminology (see figure 2). The smallest boxes which TEX manipulates are those surrounding the individual letters. Larger boxes can be built out of the undecomposable boxes surrounding the letters. Thus a line of type is also considered a box from TEX's point of view. Glue is the stuff which gets put between words and other boxes (though not between the boxes making up individual words). Leading, the distance between consecutive lines of type, is implemented in TEX through interline glue. This 'boxes and glue' model turns out to be surprisingly powerful and enables TEX to perform extraordinary feats of typesetting for example in the typesetting of mathematics.

Some of TEX's algorithms are quite well known. This is especially true for the paragraph setting algorithm (Plass and Knuth 1982), as well as the hyphenation algorithm devised by Frank Liang (Liang 1983).

The algorithm for setting paragraphs minimizes the 'demerits' associated with the setting of a particular paragraph. These demerits reflect, among other things, the 'badness' of individual lines of the paragraph which are calculated

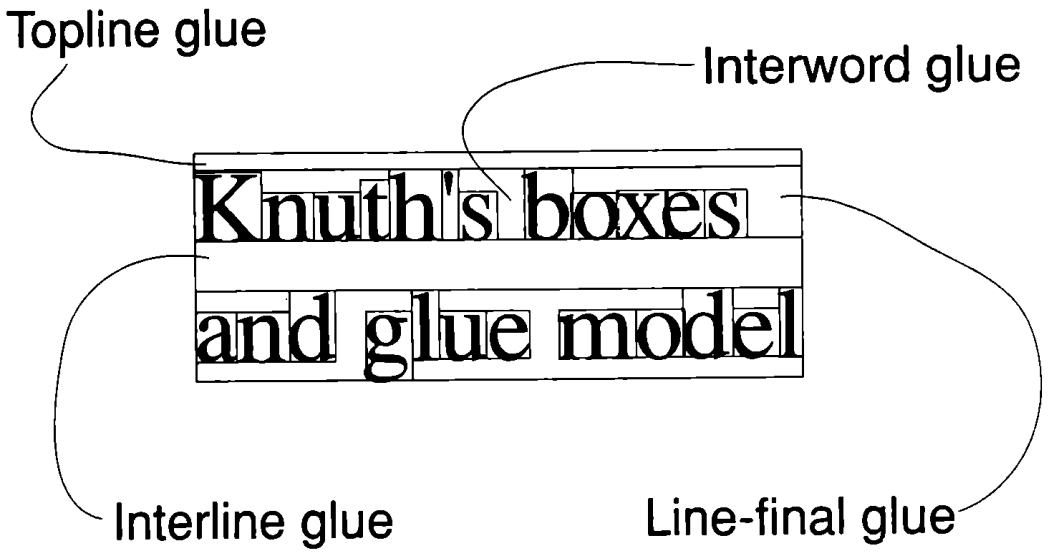

Figure 2: TEX's boxes-and-glue model

by noting the extent to which the inter-word glue has to stretch or shrink. TEX sets the paragraph by minimizing these demerits. The interesting thing to note is that this means that the paragraph as a whole is typeset in one go and a word coming late in a paragraph can influence the setting of lines coming earlier in the paragraph.

Liang's algorithm for word hyphenation is pattern-based, but departs from older versions by using both variable length patterns and patterns which both allow and inhibit hyphenation points. I will not discuss this any further here, but simply note that his method gives excellent results in a number of languages besides English. In particular the Icelandic hyphenation table does a very creditable job of hyphenating.

TEX has numerous primitives (around 300) for dealing with typesetting and also a very powerful macro programming language. It is this latter which gives TEX its status as a programming language.

Here are a few examples of the primitive operations which TEX operates with. Note that primitives and TEX macros are expressed with 'control sequences'. These usually start with a special 'escape character' which is typically \, the backslash.

- **\kern**. This command is followed by a dimension specification (e.g. in printers' points) and moves the placement of two boxes relatively to each other. Note that boxes come in horizontal and vertical versions and **\kern** can be used to position boxes both vertically and horizontally, depending on the 'mode' TEX is in. Usually kern is used to bring boxes closer together

(e.g., letter pairs like 'V' and 'A' which, because of kerning, are printed as 'VA' rather than 'VA'.

- `\looseness`. Changes to looseness mean that TEX will attempt to set a particular paragraph in more or fewer lines than the optimal setting calls for. By setting `\looseness=1` an attempt is made to open the paragraph and set it one line longer than would be the case if no `\looseness` is specified.

- `\fontdimen`. This command enables one to query the 'current font' for font parameters like the *x-height*, normal spacing, etc.

- `\penalty`. Controls the desirability of breaking at a particular point. Penalties can be both positive (making a break less likely) and negative (indicating desirable break points). Infinite penalties (having a value greater than 10000) either force (`\penalty=-10000`) or prohibit (`\penalty=10000`) a break at a particular point.

- `\sfcode`. The 'space factor code' is used to control the stretching of spaces after individual characters. Using the `\sfcode` makes it possible to, say, stretch spaces after periods more than after ordinary characters.

- `\hyphenchar`. Very few things are hard-wired into TEX. Even the hyphen# ation character can be changed. By setting `\hyphenchar\tenrm='\#`, TEX will use the hash-mark as the hyphenation character for the 10 pt Roman font (witness the first line of this paragraph).

3.3 TEX as a Programming Language

TEX has a very powerful macro language which can be used to write macros at almost any level of abstraction. The execution of these macros takes place through a process of macro expansion, where the macros are gradually reduced to primitives of the TEX language. Since macros can call other macros, it is possible to structure the code in a systematic way by gradually moving from primitive constructs to more abstract ones.

TEX observes a block structure, like most other programming languages. The block structure is achieved by using the symbols for 'open' and 'close group' which are usually the curly braces { and }. Using grouping, it is a simple matter to structure code, such that the likelihood of naming conflicts are lessened.

The TEX macro language, like most other macro languages, uses registers for the different 'data types' which are available. These registers come in five varieties:

Count registers are used for keeping integer values (32 bit). TEX has primitive operations for integer arithmetic only, but this is usually not a problem. The following piece of TEX code declares a count register named `\figno` which is initialized to 0:

```
\newcount\figno
\figno=0
```

The code for the figure macro would then take care of placing the figure and assigning a number to it which would be incremented for each figure. This last operation is achieved by:

```
\advance\figno by 1
```

Dimension registers are used for printers' dimensions, points, picas, millimeters, etc. The following piece of code declares a dimension register and then initializes it.

```
\newdimen\pagewidth
\pagewidth=170mm
```

Next come the glue registers or 'skip' registers. These contain glue specifications. The following example illustrates the definition of the `\smallskip` macro which makes use of the `smallskipamount` glue register:

```
\newskip\smallskipamount
\smallskipamount=3pt plus 1pt minus 1pt
\def\smallskip{\vskip\smallskipamount}
```

The `\smallskipamount` register is set to 3pt plus 1pt minus 1pt. The macro `\smallskip` is defined as a vertical skip (`\vskip`) of `\smallskipamount`.

Finally, we come to the box registers which are used for holding the boxes gradually acccumulated for each page. Boxes have three dimensions, as mentioned before. These can be queried or set, using the primitives `\wd`, `\ht`, and `\dp` for the width, height, and depth, respectively.

3.4 Defining Macros

We have already seen one example of how macros are defined. This is done with the `\def` primitive. Macros can take arguments, it is even possible to have macros which check for optional arguments, a higly useful feature. A typical macro with arguments is the following simple macro for setting headwords in bold face. (The percent sign % is usually a comment character in TEX. Anything coming after the % on a line is ignored by TEX.)

```
\def\hword#1{% macro for the headword
    {\bf#1\mark{#1}}}
```

This sets the headword in boldface (`\bf`) and defines a 'mark'. This mark can, for instance, be used to establish the range of entries on a particular page of a dictionary. The parameters are denoted by # and they are numbered consecutively, starting with `#1`.

Like any good programming language, TEX offers the user a conditional testing mechanism. One application of this is to print different types of proofs. For instance, it is possible to redefine the `\hword` macro in such a manner that TEX will write the headwords to a special file when the dictionary is being proofed. It

is then a straightforward matter to check whether the list of headwords thus generated is in correct alphabetical order. This can be accomplished in the following manner:

```
\newwrite\hwordfile % first a file is defined
\newif\ifproofmode  % A conditional is declared
\proofmodetrue      % Are we printing proofs? Yes we are.
\ifproofmode \message{**** Printing proofs *****}
\immediate\openout\outfile=\jobname.hwrd

\def\hword#1{% macro for the headword
    {\bf#1\mark{#1}}
    \immediate\write\outfile{#1}}
    ...
\else \message{**** Final run *****}
\def\hword#1{% macro for the headword
    {\bf#1\mark{#1}}}
\fi
   ...
```

The conditional construction

```
\if
   ...
\else
   ...
\fi
```

thus makes it easy to print different versions of the same manuscript according to need.

This has only been the briefest of introductions to TEX as a programming language, but it should, I hope, reveal to the reader something of the flavour of the TEX language.

3.5 TEX in Iceland

The Institute has been responsible for introducing TEX into Iceland. I have earlier described the steps undertaken to make TEX work with Icelandic (Jörgen Pind 1988). In particular:

- It was necessary to make a set of patterns for TEX to achieve correct (or nearly correct) hyphenation. The patterns were generated by Frank Liang's program PATGEN, using as input a 210.000 word dictionary made by the Institute for IBM in Iceland to use in IBM spelling checkers.[2]

[2] I am very grateful to Mr. Gunnar M. Hansson, general manager of IBM Iceland, for allowing us to use this material for this purpose.

- The Computer Modern Fonts had to be adapted to Icelandic by adding a few characters (e.g., 'ð' (eth) and 'þ' (thorn)).

- Changes had to be made to the standard macro collections to allow for new fonts and some differences in character definitions.

With these changes, TEX has been found to work admirably for Icelandic and has already been used to typeset a number of books. I guess Iceland must be unique in having brought out a number of TEXed books and yet no mathematics book has been typeset with the Icelandic version of TEX as yet!

4 Typography and Dictionaries

4.1 Some General Observations

The typesetting of dictionaries usually presents few problems. Dictionaries are usually set in two or three columns which are rather narrow. This can often lead to difficulties with line-breaking, since the narrow columns leave relatively little latitude for the paragraph-breaking algorithm. For this reason, it is advantageous to choose a font with a narrow set width, and, secondly, it is necessary to allow the typesetting program more flexibility in stretching and compressing interword spaces than is normal in books which are set to the full width of the page. In TEX this flexibility is controlled with the primitive `\tolerance`.

When the columns are set in register, as is usually the case, widow lines are bound to occur because the leading (interline glue in TEX) is not allowed to vary. These can be got rid of by stretching or shrinking the paragraph (or paragraphs on the previous page or pages). In TEX this is controlled by the `\looseness` primitive. If one is prepared to accept *full* widow lines (as we occasionally did in the etymological dictionary), it is possible to achieve this in TEX by setting the glue register `\parfillskip` equal to 0 pt, thus drawing the last line of a paragraph out to the full width of the column.

If the columns are not set in register (as is, for example, the case in the Oxford English Dictionary where the quotations are set in smaller type, thus forcing variable leading), it is much easier to control for widow lines since the space between paragraphs can easily be varied (this is done in TEX with the `\parskip` primitive).

It is custumary in dictionaries to print words at the top of the page, showing the range of the entries on that page. This process can very easily be automated in TEX, using the `\mark`. By `\marking` all headword entries and defining suitable macros for the outputting of the headlines, this process becomes completely automatic. Note that though I mention here the necessity of `\marking` the headwords, it is in fact *not* necessary to mark them individually. By a suitable definiton of the `\hword` macro this can be programmed (see the previous definitions of the `\hword` macros).

5 Work Finished and in Progress

The major performance test of TEX for lexicographic work was the typesetting of the etymological dictionary by Ásgeir Blöndal Magnússon. This book runs to 1231 two-column pages with forty pages of introductory material. TEX took care of the typesetting of all the pages except for two pages which contain illustrations demonstrating the use of the dictionary. These two pages were designed with a drawing program.

Originally, it was never intended that the etymological dictionary would be typeset with TEX. When keyboarding of the manuscript began in 1985, we did not have TEX, and the coding of the manuscript was such that it would be easy to transfer it to a printer for typesetting with traditional printers' typesetting codes. However, in January 1989, when we were ready to turn the manuscript over to the printer, it turned out that they did not have all the characters needed for the typesetting, and would also have difficulties with all the diverse floating accents which the book contains. At that point I decided to make some trial runs with TEX, using PostScript fonts (Adobe Times Roman). It turned out that no problems where encountered which could not rather easily be solved. Even the fact that PostScript has a fairly limited character repertoire could be remedied by drawing the missing characters with Fontographer, a font generating program running on the Macintosh (Altsys Corporation 1989).

Figure 3 shows a sample page from the dictionary.

Our major project in the future will, of course, be the dictionary of verbs outlined in the paper by Jón Hilmar Jónsson in this volume. The editing will take place in a database system, and the output of that system, a TEX script, will be generically coded.

Additionally, we have just embarked on a project to reprint some older Icelandic lexicographic works. Work is now in progress on four older dictionaries. These are all coded in the TEX language, and the intention is to bring these out in new editions. These are dealt with as textual objects, though the generic coding would, of course, considerably ease the task of putting them online, if that should be decided at a later stage (cf. Alshawi et al. 1989).

6 Issues of Coding

In recent years, more and more attempts have been made to use database systems for the creation of dictionaries. When a database is used for a dictionary, it becomes possible to *name* the fields which are being entered. The database programmer has quite a lot of freedom in the choice of these names and therefore in the choice of categories which are dealt with in the dictionary. I shall assume here that the final aim of the project is to produce a printed dictionary, though, of course, if it is made up using a database system it becomes possible to 'publish' it in computerized form, say, on a CD-ROM disk.

hreyfingu í leðju eða for, sbr. *lóna* af *lón*. Sjá so. *öðla*.

ádess. ⊙**ádeis** h. (18. öld) 'óhreinindi; óhapp; ádrepa'; af fs. *á* og *dess* af so. *dessa* (< **det(t)sa* < **dantisōn*), sbr. *að dessa niður á e-m* 'þagga niður í e-m' og *dessast* 'saurgast, versna'. Eiginl. 'það sem dettur á e-n eða skellur á e-m'. Sjá *dess*.

aðili k. 'hlutaðeigandi'; **aðild** kv. 'hlutdeild', sbr. *sakaraðild*, *réttaraðild* o.s.frv. Orð þessi lutu í öndverðu að skyldu og rétti ættingja (eða tengdamanna) í málaferlum, sk. *aðal* (1) og *aðall*.

Aðill k. fnorr. karlmannsnafn, sbr. *aðall* og *aðili*.

Aðils k. karlmannsnafn; sbr. sæ. *Adils*, sæ. rúnar. *Aþisl* < **Aðgísl*, fe. *Eadgils*. Forliðurinn *að-* á skylt við *aðal-* (2) og *óðal*, sbr. fsæ. pn. *Adi*; um viðliðinn sjá *gísl* (1).

adíu, adjö uh. (18. öld) 'kveðjuorð'. To. úr d. *adjø* < fr. *adieu* < *a Dieu*, eiginl. 'guð veri með þér'.

admíráll, aðmíráll k. (nísl.) 'sjóliðsforingi'. To. úr d. *admiral* < ffr. *a(d)miral* (s.m.) < arab. *amir* 'höfðingi'. Sjá *emír*.

Adólf k. karlmannsnafn; tökunafn, líkl. ættað úr þ., sbr. nhþ. *Adolf*, fhþ. *Athalwolf*, *Athulf*, gotn. *Athaulfs*; líkl. < **aþa-wulfaz*. Sjá *aðall* og *úlfur*.

adressa kv. (19. öld) 'heimilisfang'; **adressera** s. 'skrifa heimilisfang, ...'. To. úr d. *adresse*, *adressere* ættuð úr fr. *adresser*, sbr. lat. *ad* 'til' og *directum* (l.h.) 'beint'.

aðsjáll l. 'níðskur, naumur í útlátum' < **at-séall*; e.t.v. leitt af gamalli forskeyttri so., sbr. gotn. *atsaihwan* 'gaumgæfa' og ísl. *sjá að sér*.

-aður, †-aðr k. viðsk. no. eins og *munaður*, *unaður*. Skiptist á við *-uður* (s.þ.) og er komið af germ. **-ō-þu-*. Þetta viðsk. er runnið af verknaðarviðsk. **-þu-* < ie. **-tu-* sem skeytt var við stofn ō-sagna. Víxl *-að-* og *-uð-* eru upphaflega háð sérhljóði eftirfarandi endingar, t.d. nf. et. **-aþuʀ* > *-uðr*, en ef. et. **-aþaʀ* > *-aðar*, og gegndu þessar tvær myndir viðsk. í upphafi sama hlutverki, en síðar hefur *-að-* verið að mestu sérhæft í verknaðarmerkingu, en *-uð-* að mestu í gerandmerkingu. Sjá *-uður*, *-naður* og *-nuður*.

aðventa kv. 'jólafasta'. To., komið úr lat. *adventus* 'koma', ɔ: koma eða fæðing Krists í heiminn.

aðventistar k.ft. kristinn trúflokkur; nafngiftin lýtur að trú þeirra á endurkomu Krists.

aðvífandi lh.nt.: *koma a.* 'koma að eins og af tilviljun'. Sjá **vífa* (2).

1 af fs. (ao.) 'frá, burt'; sbr. fær., nno. og sæ. *av*, d. *af*, gotn. *af*, fe. *æf*, *of*, fhþ. *ab(a)*, lat. *ab* (< **ap*), gr. *ápo/apó*; sk. *afar*, *afr* (2), *aftur*, *at* (4), *efja*, *eftir*, *efsa*, *öfund*, *öfugur* og e.t.v. *aftann*. Sjá *af-* (2).

2 af- forskeyti; sbr. fær., nno. og sæ. *av*, d. *af-*, gotn. *af-*, fe. *of-*, fhþ. *ab-*, *aba-*, *abo-*, lat. *ab-*, gr. *apo-*, fi. *apa-*. Sjá fs. *af*. Ýmist gamalt forskeyti eins og t.d. í *afbragð*, *aflát*, *afráð*, *afrek* o.s.frv. eða síðar forskeytt fs. eða ao., sbr. t.d. *afdráttur*, *afhýða*, *afrækja* o.fl. Forskeytið heldur oft eiginlegri (staðarlegri) merkingu sinni, sbr. t.d. *afbjarga*, *affjalla*, *afhús*, *afhvarf*, en stundum verður tákngildi þess niðrandi eða herðandi, t.d. *afgelja*, *afgera*, *afát* 'ofát', *afgamall*, *afkostir*, *afstopi* 'ofstopi', eða meira eða minna óeiginlegt, t.d. í *afráð*, *afrek*.

áfa kv., merking ekki fullljós, en líkl. 'fjandskapur, mein', sbr. físl. *iǫll ok ófu / færik ása ſonum* (Lokas.). Sumir telja að *áfa* sé í ætt við lo. *æfur* og *ófa* kv., en stofnsérhljóðið, germ. **ē*, er annars óþekkt í þeirri orðsift. Aðrir ætla að *ófu* (í Lokas.) sé eiginl. s.o. og *áfá* og þá < **ófo* < **áfǫ́*. Enn aðrir tengja orðið við *vofa* kv.; lítt sennilegt; *áfa* er stakorð og ritháttur ekki öruggur, e.t.v. stendur *ófu* fyrir *ófu* og orðið þá s.o. og *ófa* og tengt lo. *æfur*. Allt óvíst.

áfá kv. (18. öld) 'áhrif, t.d. af vínanda', sk. *áfengur* l. 'sem hrífur á'; **áfengi** h. 'vínandi' og *áfang* h. E.t.v. < **anfa(n)hō* dregið af forskeyttri so. **anfa(n)han*, sbr. fhþ. *anafāhan* 'byrja' (eiginl. 'grípa á'), eða myndað af so. *fá* (1) eða öllu heldur samb. *fá á*.

áfang h. † 'átak, hnjask, ofbeldi'; e.t.v. leitt af forskeyttri so. **anfa(n)han* 'grípa í, byrja', sbr. fhþ. *anafang* 'átak, hrifs, byrjun'; sk. *áfá* og *áfengur*. Sjá *fá* (1).

áfangi, †áfangr k. Sjá *áivangr*.

afar ao. 'mjög', einnig forskeyti **afar-**, sbr. **afarkostir**; líklega sama orð og gotn. *afar* 'á eftir, síðar', fhþ. *avar*, *abur* 'aftur', sbr. nísl. **afur-** (< **afr-*) sem notað er sem forskeyti í líkri merk. og *afar-* (*afurverði*, *afurnagandi*) og *af-* (2) sem stundum er haft í herðandi merkingu, t.d. *afkostir* s.s. afarkostir, *afgamall* 'mjög gamall'; *afar* sýnist vera einsk. miðstig af fs. eða ao. *af*, sbr. fi. *ápara-* 'aftari, síðari'. Aðrir telja að *afar* sé sk. gotn. *abrs* 'sterkur'. Sjá *afr* (2).

af-baka s. (16. öld) 'aflaga, skekkja'; sbr. nno. *avbakleg* 'öfugsnúinn, óhægur, erfiður, afskekktur', *avbekt* 'þver, öfugur', sæ. máll. *åbäklig* 'luralegur, ólögulegur', fær. *avbeklaður* 'illa troðinn, aflagaður (um skó)'. Myndun orðsins er óljós, þótt það sé sýnilega tengt no. *bak*. F.J. (1914) ætlar að það merki í öndverðu 'að bakfletta trjávið, höggva ávala af trjám' og styðst þar m.a. við umsögn B.H., en það samræmist lítt merkingu og formi nno. og sæ. orðmyndanna. Sjá *bak* og *bekill*; ath. *bækill*. **-baldi** k. (nísl.) 'ofsafenginn maður', sk. *baldinn* l. og *ofbeldi* h. **-bragð** h. 'e-ð frábært'; sbr. nno. *avbragd* og fær. *avbragd-* í *avbragdsstyrki* 'mikið afl'. Leitt af so. **ab-bregðan* eða *bregða af*, sbr. *afbrugðinn* 'frábrugðinn, ólíkur' og *afbrúðig(u)r*. **-brúðig(u)r** l., **af-brýði** (†**af-brygði**) kv. Sjá *ábrúðig(u)r*. **-danka** s. (nísl.) 'svipta metorðum eða stöðu'; **-dankaður** l.

Figure 3: A sample page from the etymological dictionary

The traditional way of making a dictionary has been to proceed in a somewhat different manner, writing the dictionary entries on slips of paper.[3]

While the comparison between slips of paper, a file cabinet, and a database system is often made, this comparison is somewhat misleading since categories on the written sheets or slips are usually *not* named. In the case of dictionaries this is most clearly the case. An example will show this. Figure 4 shows a slip from the collection which was used in the making of the first standard dictionary of

letur (-urs, pl. ds.) [le:dœɋ, lɛ:tœɋ] n. **1. a.** Skrift, Typer: *gotneskt, latneskt l.; færa e-ð í letur,* optegne n-t, føre i Pennen; *sett l.,* en Slags Halvfraktur, nærmende sig til Schwabacherlypen. — **b.** *leturs land,* Papir (BóluHj. 255); *letra rolla* (egl. Typefaar) (BóluHj. 217) = *prentsmiðja.* — **2.** Indskrift: *l. á steini.* **-band** [-r-ban·t] n. Forkortelse, Abbreviatur. **-breyting** [-brei:diŋk, -brɛi:t-] f. Udhævelse. **-gerð, -gjörð** [-ɋer·ð, -ɋör·ð] f. **1.** Bogstavskrift, Typernes Karakter: *leturgerðin er alt önnur,* Typerne er af en helt anden Karakter. — **2.** Skrivning: *hvorugur þeirra hafði numið svo mikið í leturgjörð, að þeir mættu rita nöfn sín* (JThMk. 382). — **3. a.** *(samning rits)* Optegnelse, Affattelse af et Skrift.

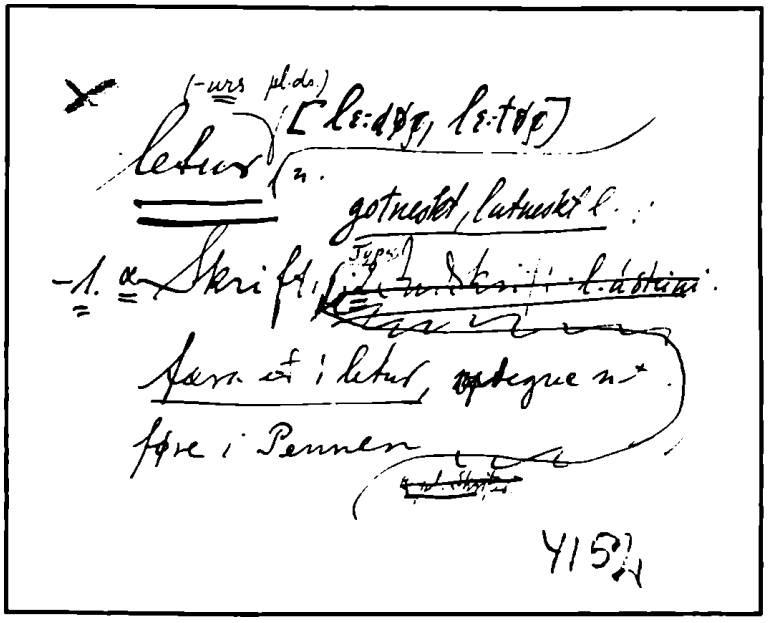

Figure 4: A sample entry from the dictionary by Sigfús Blöndal *and one of the dictionary slips on which it is based*

modern Icelandic, Sigfús Blöndal's Icelandic-Danish dictionary (Blöndal 1923). It is quite obvious that no categories as such are marked on the slip. They can, however, be *inferred* from the slip by the use of markings which indicate different fonts. The slip implicitly marks categories by the use of underlining and other typographical marks.

[3] Actually, using dictionary slips was quite a breakthrough in the making of dictionaries. This can be seen if one has a look at the 18th century monumental Icelandic dictionary by Jón Ólafsson from Grunnavík (which was never finished). This was written out as a single manuscript, at first having reasonable space between the entries, but gradually deteriorating into complete chaos as entries were added to the manuscript.

This approach is quite natural, considering that dictionary editors, such as Sigfús Blöndal, were working with the sole aim of producing a printed dictionary. They thought of their work as that of producing a *text*, and their approach was quite plainly a 'typographical' one where the only things they needed to keep distinct in the manuscripts were changes which would show up on the printed page, like font changes.

This approach has no doubt been almost universally followed, at least until quite recently. Some published dictionaries have been made available to researchers. These are generally typographically coded and bringing them online has often proved to be a formidable task (Alshawi et al. 1989).

This discrepancy between the database representation of a dictionary and the printed, typographical, representation is quite unfortunate and various steps have been taken to close the gap. This is currently not too difficult a task and I want to discuss here briefly how one could achieve this aim with TEX.

A programming language such as TEX makes it possible to code the manuscript at any level of abstraction which one finds most convenient. The primitives which TEX deals with are for the most part typographical ones, as already discussed. However, it is by no means necessary to use these primitives directly. Let me illustrate this by taking the entry from the Icelandic-Danish dictionary shown in figure 4 as an example. This can be typographically coded as follows in TEX (the phonetic transcription has been left out):

```
\bold{letur (-urs,} pl. ds.) [...] n. 1. a. Skrift, Typer:
\ital{gotneskt, latneskt l.; føra e-ð í letur}, optegne n-t,
føre i Pennen; \ital{sett l.}, en slags Halvfraktur,
nærmende sig til Schwabachertypen. --- \bold{*b.}
\ital{leturs land}, Papir (BóluHj. 255);
\ital{letra rolla} (egl. Typefaar) (BóluHj. 217)
= \ital{prentsmiðja}.
```

This example should be mostly self-explanatory. The instructions `\bold` and `\ital` change respectively to the bold and italic fonts. This representation is fairly close to the one given on the slips themselves, as depicted in figure 4. Note incidentally the somewhat strange use of fonts in the first line where parentheses do not balance correctly with respects to fonts. This use is probably quite natural for the printer (who has, after all, been taught that a delimiter character, for example, should belong to the same font as the preceding text). To someone accustomed to the notions of 'blocking' and 'environments' from computer science this manner of font change does seem illogical.

If we care to analyze the example from a functional perspective, we can easily see that it contains a number of different categories. There is the headword, which is printed in bold type, and so is the grammatical ending signifying the genitive. Here we have an example, ever so common in dictionaries, of one font being used for disparate categories. Additionally, there are examples of use and phrases shown in italic type, of sectioning (using numbers and letters of the alphabet), and of source references ('BóluHj.' being the Icelandic 19th century poet Hjálmar Jónsson).

A different way of coding would be to code the categories directly without any reference whatsoever to their typographical implementation. This approach, which has quite a short history, has been variously named 'logical' or 'generic' coding, and can thus be distinguished from the *visual* coding shown above. Generic coding has recently received increased attention through the standardization of the SGML (Standard Generalized Markup Language) (ISO 1986, Barron 1989, Bryan 1989). Similar concepts have been expressed in other languages and formatters, though SGML carries it to its logical conclusion: SGML is simply a manner of coding a manuscript, and has really nothing do do with typesetting, or database manipulation. It does, however, embody a manner of representing the structures which are to be found in a particular document.

In particular, as regards TEX, Leslie Lamport's macro package LATEX is very much geared towards logical coding (Lamport 1986; see also Lamport 1988). LATEX is a macro package used for general document processing. It uses the concept of separate 'style files' to capture the different formatting needs of reports, articles, books, etc. Furthermore, it defines categories such as 'titles', 'sections', 'chapters', 'footnotes', and so forth to express the different logical categories of documents.

The TEX macro language is such that one can easily implement macros to any degree af abstraction required. Using such an approach, it would be easy enough to code the above example from Sigfús Blöndal's dictionary in the following manner (I have formatted it here for easier readability):

```
\hword{letur} (\decl{-urs}, \xx{pl. ds.}) \phon{[...]}
\pos{n.}
\sense{1.}
 \subsense{a.} \trans{Skrift, Typer}:\V
  \exampl{\ic{gotneskt, latneskt l.; færa e-ð   letur},
  \da{optegne n-t, føre i Pennen}};
  \exampl{\ic{sett l.},
  \da{en slags Halvfraktur, nærmende sig til
             Schwabachertypen}}. ---
 \subsense{*b.}
  \exampl{\ic{leturs land}, \da{Papir} \source{BóluHj. 255};
  \exampl{\ic{letra rolla} \da{(egl. Typefaar)}
         \source{BóluHj. 217)}
  = \xrf{prentsmiðja}.
\sense{2.}
```

This, I hasten to add, is just a demonstration of the manner by which it would be possible to proceed. In particular, in no way is this coding based upon a study of the entries in this dictionary, a study which it would be necessary to undertake if it were desired to code the dictionary in this manner.

The categories mentioned above should be easy enough to understand since they have been given names which are fairly self-explanatory (the categories `\ic` and `\da` stand respectively for 'Icelandic' and 'Danish') and it will thus not be

necessary to give detailed explanations for each of them. It is, of course, immediately apparent that the manuscript gets considerably more complicated when such a system of coding is employed. After all, a lot of categories are delimited which will not find any particular realization in the printed text. By working from such a manuscript it is much easier to set up a one-to-one relationship with a database representation which of course is considerably more difficult when dealing only with a visually coded manuscript.

The astute reader will probably object to the choice of terms for the entries labelled **\sense** and **\subsense** in the above extract, since these only refer to numbers and letters and cannot strictly be said to denote the sense. This is, of course, true. In this case it would have been better to label the whole passage belonging to the particular sense, leaving out the numbers and letters and letting TEX assign these automatically. The point here is simply that it is possible to approach the task of coding in different ways, and it is difficult to specify once and for all a finite set of categories that will take care of all the entities one could conceivably want to code.[4]

One example will illustrate this. The etymogical dictionary, like all of its kind, contains *n* different accents which have to be coded for. In TEX, accents are expressed with special macros which make use of an **\accent** primitive. Thus one would write **\=a** to get 'ā', where the **\=** signifies a floating bar accent, or **\'a** to get á etc. But this command will not always give the correct result. Thus if one attempts to put an acute accent on top of a 'k' by writing **\'{k}** the result is ḱ. The correct version should look like 'ǩ'. This reflects a limitation of the **\accent** primitive in TEX which can be circumvented by writing special purpose macros for letters like 'ǩ'.

To obtain this effect it is necessary to write a special purpose macro in TEX. However, in that case, it is of course necessary to know about the fonts being used for typesetting. One of the major premises of generic markup is that such knowledge is not necessary, indeed it is not necessary to know how the text will eventually be used, say, whether it will be printed or put into a database.

6.1 Visual Coding and Direct Manipulation

The approach to coding which has been described here, is language-based and thus contrasts very much with the 'direct manipulation' approach which has in recent years been popularized especially on the Macintosh computer. As regards typography, the direct manipulation approach entails that the user points to or 'clicks' on words or letters on the screen and then typically chooses the relevant font from a menu. This was the pattern of usage which was embodied in MacWrite, the archetypical Macintosh word-processing program. The effects of the font changes could be immediately seen on the screen, in a WYSIWYG 'What you see is what you get' representation. The user interface was immediately hailed as a breakthrough, which of course it was, and yet, as time has shown, it has its problems. This can be seen in the evolution of word-processing programs

[4] I guess The DANLEX Group (1987) would want to argue differently, since they have attempted to provide a taxonomy of all the different categories which can occur in a dictionary.

for the Macintosh which tend to move them closer to a language-based representation. Thus the notion of 'style sheets', an idea borrowed from Brian Reid's program *Scribe*, has now been carried over into almost every word-processing program for the Macintosh (Reid and Walker 1980). Using style sheets, it becomes possible to mark sections in a semi-generic or logical manner. Unfortunately the notion of style sheets only applies to paragraphs, and is thus useless for the making of dictionaries where one is mainly interested in categories at a much finer granularity (i.e. sub-paragraph categories).

As demonstrated in this paper, a language-based formatter like TEX can easily be accommodated to a manuscript generated from a database and thus it can deal with categories at any level. I can state without hesitation that our experience using TEX has shown that it is eminently suited for lexicographic work.

References

Alshawi, Hiyan, Bran Boguraev, and David Carter. 1989. Placing the Dictionary On-Line. Bran Boguraev and Ted Briscoe [Eds.]. *Computational Lexicography for Natural Language Processing*:41–63. Longman, London.

Altsys Corporation. 1989. Fontographer, Users's Guide. Plano, Texas.

Barron, David. 1989. Why use SGML? *Electronic Publishing*, 2(1):3–24.

Blöndal, Sigfús. 1923. *Íslensk-dönsk orðabók*. Reykjavík.

Bryan, Martin. 1988. *SGML: An Author's Guide to the Standard Generalized Markup Language*. Wokingham, Addison-Wesley.

The DANLEX Group. 1987. *Descriptive Tools for the Electronic Processing of Dictionary Data*. Lexicographica, Series Major, 20. Max Niemeyer Verlag, Tübingen.

Healy, A. diPaolo. 1985. The Dictionary of Old English and the Final Design of its Computer System. *Computers and the Humanities*, 19:245–249.

ISO. 1986. International Standard 8879: Standard Generalized Markup Language (SGML). s.l.

Knuth, Donald E. 1984a. Literate Programming. *Computer Journal*, 27(2):97–111.

Knuth, Donald E. 1984b. *The TEXbook*. Addison-Wesley, Reading, Massachusetts.

Knuth, Donald E. 1986a. *TEX: The Program*. Computers and Typesetting, vol B. Addison-Wesley, Reading, Massachusetts.

Knuth, Donald E. 1986b. *The* METAFONT*book*. Computers and Typesetting, vol C. Addison-Wesley, Reading, Massachusetts.

Knuth, Donald E. 1986c. METAFONT: *The Program*. Computers and Typesetting, vol D. Addison-Wesley, Reading, Massachusetts.

Knuth, Donald E. 1986d. *Computer Modern Typefaces*. Computers and Typesetting, vol E. Addison-Wesley, Reading, Massachusetts.

Knuth, Donald E. 1986e. Remarks to Celebrate the Publication of Computers and Typesetting, *TUGboat* 7:95–98.

Lamport, Leslie. 1986. LATEX. *A Document Preparation System*. Addison-Wesley, Reading, Massachusetts.

Lamport, Leslie. 1988. Document Production: Visual or Logical. *TUGboat* 9:8–10.

Liang, Franklin M. 1983. *Word Hy-phe-na-tion by Computer.* Report STAN-CS-83-977. Stanford University, Department of Computer Science.

Pind, Jörgen. 1986. The Computer Meets the Historical Dictionary. *Nordisk DATAnytt* 16(10):41–43.

Pind, Jörgen. 1988. Umbrotsforritið TEX. Íslenskun þess og gildi við orðabókargerð. *Orð og tunga*, 1:175–219.

Plass, Michael, and Donald E. Knuth. 1982. Choosing Better Line Breaks. Jurg Nievergelt, Giovanni Coray, Jean-Daniel Nicoud, and Alan C. Shaw [Eds.]. *Document Preparation Systems: A Collection of Survey Articles*:221–242. North-Holland, Amsterdam.

Reid, Brian K., and Janet H. Walker. 1980. *Scribe: Introductory Users's Manual.* [3. ed.] Unilogic, Pittsburgh.

Institute of Lexicography
University of Iceland
101 Reykjavík
Iceland
jorgen@lexis.hi.is