SGML and LaTeX*

Horst Szillat

Sella-Hasse-Str. 31,
D–12687 Berlin, Germany
szillat@berlin.snafu.de

SGML — *Standard Generalized Markup Language* — is a formal language to describe structured text documents. It should be introduced here by comparison to TeX and LaTeX.

It is interesting to have a look at how Donald E. Knuth introduces TeX in the TeXbook himself. The beginning is to simply type in the text and TeX mainly does what one expects it to do. Quite a lot of more or less complex rules have been implemented to provide these results. An example of this behaviour is the *space factor code* (`\sfcode`). Using this code TeX is able to identify most of the ends of sentences. Moreover TeX is realized in a way that one can program almost all kinds of printing layouts. In this way one can program a macro which influences the layout in any place. So TeX is a layout oriented system which is able to format texts for printing and to do a bit more.

Although LaTeX simply is TeX, too, and has all these characteristics, too, it introduces a new idea of representing the text input. The basic idea is that a text is given in the form of *embedded environments*. The layout of a text portion depends on the environment it is embedded in. Moreover, the layout of whole environments may depend on which other environment they are embedded in. The user can define new environments (`\newenvironment`) which realize a user defined layout. But the main point is that the author inputs his text on a less technical but a more abstract level. This way LaTeX enforces the idea of separating the text structure from the printing layout. Changing the layout in LaTeX means to replace the existing style files, only. One could do the same in plainTeX directly, of course. One can do structured programming in assembler, too, but assembler does not enforce it.

Now one can simply say SGML is LaTeX without TeX to be written in a slightly different manner. This means SGML is a representation of the text in its hierarchical structure without any idea of a layout. If one has lost the layout there has to be an advantage on the side of the text structuring. And so it is, indeed. LaTeX's environments are called *elements* in SGML. Within a certain model one can now define which way the elements are embedded in each other and where text is to be allowed. Within that model the amount and the order of embedded elements and text is defined.

Such a definition of a text structure is called *document type definition* (*DTD*). The 'best-known' example of a SGML document type definition is HTML (*Hypertext Markup Language*) used for the World Wide Web. While processing the document an SGML-parser is able to validate the structure of the document by the given document type definition. A simple example should illustrate this:

```
<!ELEMENT section    - - (paragraph?,subsection+)>
<!ELEMENT subsection - - (paragraph,paragraph+)>
<!ELEMENT paragraph  - - (#PCDATA)>
```

These lines are to be read as follows: An environment/element called `section` consists of maximum one `paragraph` and at least one `subsection` in this order. A `subsection` consists of exactly one `paragraph` plus at least one `paragraph`, e.g. at least two `paragraph`s. And at last, a `paragraph` consists of letters. Here it is not possible anymore — unlike in LaTeX — to put the first subsection before the first section. One could define the LaTeX environments with such control structures, too. But again, LaTeX is not designed for this goal and does not enforce it, while such validating is the nature of SGML.

Another structural advantage over LaTeX is the consequent distinction between *parameter* and *data*. The lines

```
\label{Hallo!}
\section{Errors}
\unknown{whatever}
```

show that in LaTeX one can never be sure what is human readable text (data) and what is internal technical information (parameter). On the other hand SGML has a strict idea of this distinction. As long as the SGML structures are not misused malevolently it is possible to make this distinction without even understanding the content. This is an important condition for any computer based data processing. An example will be given later.

But even in the days of total computerizing the final goal of text representing is to print the text onto paper. There are two projects/tools specially designed for the printing of SGML documents. FOSI (Formatted Output Specification Instance) and DSSSL (Document Style Semantics and Specification Language). But why not use LaTeX?

*Reprint from the Annals of the UK TeX Users Group **Baskerville**, Volume 5.2, March 1995. Published with permission of both Baskerville editor and author. Presented at the UK TeX Users Group conference 'Portable Documents: Acrobat, SGML, and TeX', on 19 January 1995, London, England.

LaTeX has some characteristics which make it the first choice.

- The structure of SGML and LaTeX are very close, so that the documents are easily to convert.
- LaTeX is a programming language and therefore can realize a wide range of unforeseen layouts.
- LaTeX has been used for many years by a large number of people. So there exists a widespread experience.

A principal scheme of the processing might look as shown in Figure 1.

Unfortunately it is not sufficient to convert the elements into environments and to write the needed style files. As already mentioned SGML and LaTeX have different ideas of what is data and parameters. So it is especially necessary to transform SGML-data to LaTeX-parameter so that LaTeX can handle it more flexibly. A typical example is the following:

```
<section label="main-section">
<title>section title</title>
section content
</section>
```

What one would like to get is something like this:

```
\section{section title}\label{main-section}
section content
```

One should note that `main-section` is a parameter before as well as after conversion while `section title` moves from being data to being a parameter. The easiest way to solve this problem is to introduce additional braces within the LaTeX environment. Depending on the number of parameters defined in the definition of the environment the data is treated as a parameter or the last parameter is treated as data:

```
<name parameter="value">data</name>
```

converts to

```
\Bsgml{name}{value}{%
data%
}\Esgml{name}
```

With some (yet still to be defined) command

```
\NewSgmlEnv{name}[n]{...}
```

one gets:

- both `value` and `data` being a parameter for $n = 2$.
- `value` being a parameter and `data` being data within the environment for $n = 1$.
- both `value` and `data` being data within the environment for $n = 0$.

Note that this conversion can be done without any conversion parameters. All programming, e.g. replacements are done in LaTeX. This is a major difference to the widely used SGML-to-whatever converter `format` which works with replacement tables.

But the real reason for why I started to develop my own SGML to LaTeX converter is that I felt the necessity to manipulate the data within the conversion process.

The main questions are what information about the used words are needed for typesetting and where this information comes from. Again this seems to be a typical non-English problem. In German there are two similar problems: hyphenation and (wrong) ligatures.

Basically German hyphenation rules are easily to be adapted for pattern matching and ligatures can be applied. (Hyphenation is allowed before the last consonant out of a group of consonants. There is no hyphenation within a group of consonants at the very beginning or end of a word. Certain combinations of consonants count as one single consonant. Easy, isn't it?) At the present there is a problem with the umlauts. But this problem should disappear with the `dc`–fonts. The real problem raises with complex words, e.g. words which are composed of several words but look like one. These words have to be hyphenated between the elements of the compound. This fools every pattern matching. Moreover, there should not be any ligature in these places. The reason is that one does not want to have less space 'between words'.

An example of a rather unsuspicious word is `aufflammen`. One would guess the hypenation `auff\-lam\-men`, which is wrong, of course. The english translation gives a hint: *flame up*. Within terms of `german.sty` one should write `auf"|flam\-men`, where `"|` means: hyphenation is allowed but no ligature is allowed. The printing result is 'aufflammen' instead of 'aufflammen'.

Unfortunately TeX is unable to store this information neither in the hyphenation table nor in the document preamble by `\hyphenation`. Maybe a successor of TeX will be able to do so. So far an author writing in LaTeX has to input this information directly into the document, well — if he cares...

Using a conversion from SGML to LaTeX the converter would be the right place to insert the additional hyphenation and ligature information. The converter has to use two dictionaries — a standard dictionary and a special dictionary. It is not unusual that special matters need special terms and consequently special dictionaries. But in German the problem is that one can create new complex words ad hoc. These new compounds may be specific to a particular document. So it would be a nice idea to ship this special dictionary as a structural part of the document!

In this way the author does not have to care about every single hyphenation and ligature exception, but additionally has a spell checker.

But unfortunately there is even a worse case which needs special treatment. It is the word `Baumast`, which can be `Bau\-mast` (*mast used in building*) `Baum\-ast` (*bough of a tree*), both made of wood, of course. This is a really rare case that a word must be tagged with an additional information where it occurs within the document. This information should explain which word is to be meant. One could do that in the form of an explicit hyphenation information. In SGML it could look like

```
<word which="Baum\-ast">Baumast</word>
```

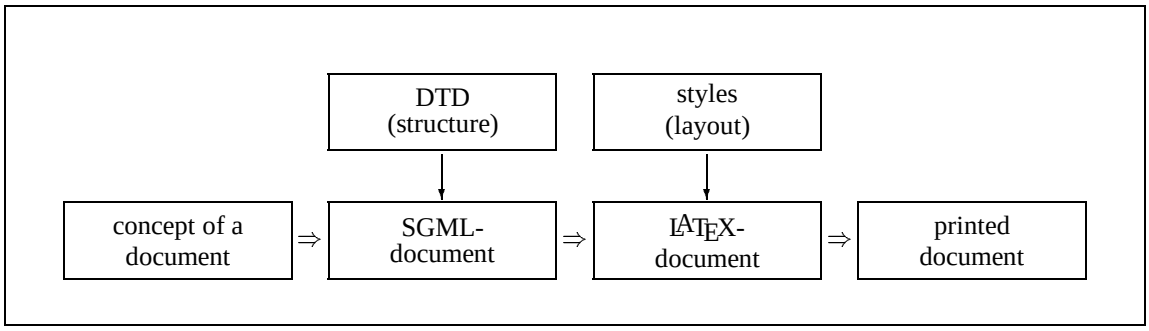

Figure 1: *Processing of a SGML document*

(This example is simplified. It would be more correct to use a SDATA-Entity so that the LaTeX-specifics are hidden.)

Note that the hyphenation information on the words `auf-flammen` and `Baumast` are totally different things. The first one is part of the layout information (how to print out?), while the second one is a structural part of the document (which word?).

Summarizing one can state that SGML and LaTeX are a good pair. Using the specifics of both systems one can do a lot of things correctly in an easier way.

Further reading

- H. Szillat: *SGML — Eine praktische Einführung* ISBN 3-929821-75-3, Int. Thomson Publ.
- ftp-server: `ftp.ifi.uio.no`
- news groups: `comp.text.sgml`, `sgml-l`