Converting Formatted Documents to HTML

Jon Stephenson von Tetzchner
Norwegian Telecom Research

Abstract

The World Wide Web document language, HTML (HyperText Markup Language), is a logical language. It uses tags (markups) to define the different parts of the document, eg headers and bullets. How the different parts of the document are displayed on the screen depends on the viewer's software. This may be a problem when converting documents to HTML as most document include formatting information. Some document formats do not include any logical information, hence making the process of converting them to HTML quite tedious.

The present paper describes the current version of the fm2html document converter. The converter translates FrameMaker documents to HTML. FrameMaker documents are logically structured and include formatting information. The FrameMaker Interchange Format (MIF) is briefly described and the major differences between MIF and HTML are discussed. The description of the conversion process focuses on the problems related to converting various parts of a document: text, figures, tables and hyper links. Some selected problems are discussed in greater detail, and changes that would be desirable in HTML to solve these are indicated.

Introduction

Fm2html originated as an internal project at Norwegian Telecom Research. It was intended to be used for converting research documents between FrameMaker and HTML. There have been several upgrades. The current version of fm2html converts text, figures, tables, character formats, etc., but MIF (FrameMaker Interchange Format) still contains several features which are not converted. Normal FrameMaker documents may also be converted; they are converted to MIF before being converted to HTML.

The design of fm2html reflects many of the problems related to converting formatted documents to HTML, and the solutions may therefore also be used for writing converters for other document formats. However, some problems remain unsolved, and may be solved only by extending HTML with more formatting options. Some of the problems will be solved by the proposed HTML+ format.

Document formats

The FrameMaker document format includes a full specification of the document contents and layout. A document in MIF can roughly be divided into four sections:

1. Specification of various document structures (style sheets), eg paragraphs, tables, fonts and variables. This information is used to format actual structures later in the document.
2. Definition of tables and frames (figures).
3. Page layout information.
4. Text paragraphs with references to other paragraphs and documents.

The various sections will make reference to the other sections: Pages include paragraphs, paragraphs include references to tables and frames, and use style sheets, tables use style sheets, etc.

Although HTML includes some character formatting, the HTML format may actually be considered a logical description of the document, using logical tags to describe the document contents. This logical description suggests a way of formatting the document, but the actual formatting is decided by the writer of viewing software or by the user of the software if the software is flexible. Since the two languages differ in many aspects, some formatting information is bound to be lost on the way.

The conversion process

The current version will convert a FrameMaker document or book to HTML, including automatic generation of a table of contents. The process for converting a single file is shown in Figure 1, and the process for converting a book is shown in Figure 2.

FIGURE 1. Conversion of a single file from FrameMaker to HTML.

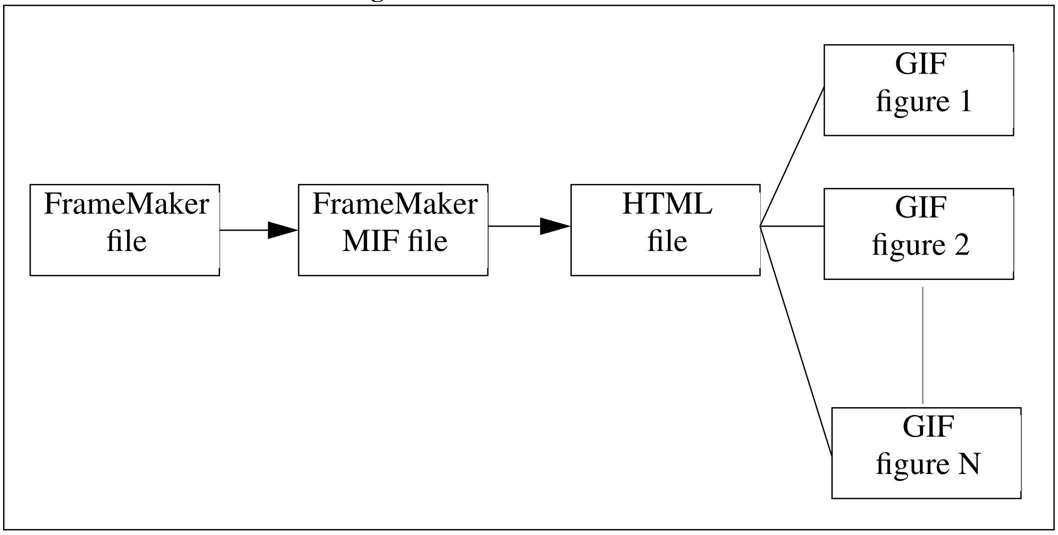
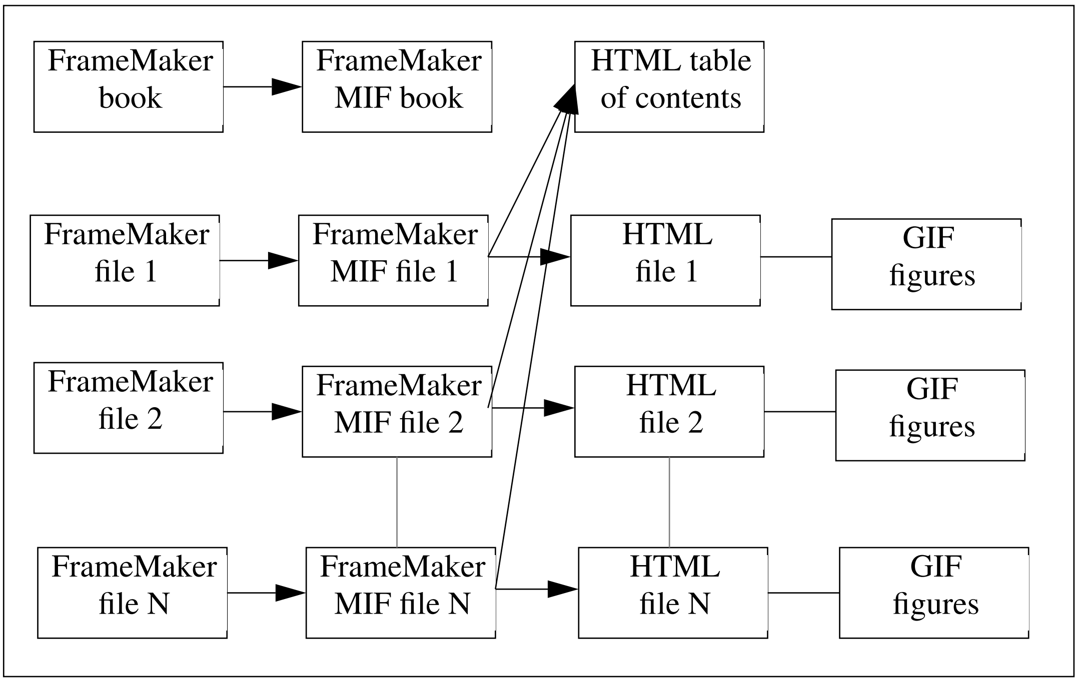


FIGURE 2. Conversion of a FrameMaker book to HTML.



The first section of the MIF document is mostly ignored by the converter since it includes information that cannot be converted to HTML. The information in this section could have been used to guess the type of paragraph by looking at its definition. Big fonts, for example, could indicate headings. This approach was not considered viable because writing styles vary greatly. Some information may be taken from this section at a later stage, though, to enable better formatting, with added formatting options in HTML.

The second section of the MIF document is converted. This conversion process is described further later in this paper.

Section three of the MIF document is mostly discarded since it includes information about page layout; pages are not defined in HTML, and page layout with multiple columns, etc., is not possible.

Section four of the MIF document contains the text of the document divided into paragraphs. The paragraphs include references to tables and frames in section 2. Most of this section is translated.

Converting a FrameMaker file

To convert a FrameMaker file, it must first be converted to the FrameMaker Interchange Format (MIF). This is done by calling the FrameMaker program fmbatch. Fm2html then converts the MIF file to HTML (Figure 1).

During the process of converting the MIF file, figures are extracted to separate files included in the HTML document, and a table of contents is automatically generated. This is described in more detail later in this paper.

Converting a FrameMaker book

The process of converting a FrameMaker book is very similar to that of converting a single file. The FrameMaker book file is converted to MIF. The names of all the files in the book are extracted and converted to MIF. A single table of contents file is generated for all the files in the book, and figures are extracted in the same way as for single files. Each MIF file becomes a single HTML file (Figure 2).

Converting from MIF to HTML

As already mentioned, MIF is very rich. It includes as lot of data that is useless in the context of an HTML document. It was therefore decided to extract only the needed information from the document and discard the rest. The extent of the information that is needed has grown since the first version of fm2html was written, but the principle is the same: A lot of information is discarded.

This does have some effect on the parsing. The converter is quite tolerant regarding different versions of FrameMaker. It will not break when a new line of information is introduced; it will just ignore it. However, this also means that fm2html is dependent on comments after the end of structure marker “>”, otherwise it will not be able to tell what type of end marker it is. This means that MIF files generated by hand or by other word processors are likely to fail. Since it is very unlikely that anybody will write MIF files by hand and the number of sources of MIF files is small, this has not been considered a problem.

Converting text

Fm2html makes use of the paragraph styles in the FrameMaker document. It discards information about the paragraph format (except local character formatting) and uses the name of the paragraph style to decide how to format the paragraph in HTML. This procedure is logical since the resulting HTML document may be displayed in a number of ways, and it will often be quite impossible to get the HTML document formatted the same way as the original FrameMaker document.

Figure 3 shows two paragraphs in MIF format. The paragraphs have paragraph styles 1Heading and Body respectively. The layout of these paragraphs is defined earlier in the document. The paragraph styles are bound to internal tags HEADING1 and BODY, which in turn are bound to the formatting shown in Figure 4.

FIGURE 3. MIF paragraphs.

```
<Para
<PgfTag ‘1Heading’>
<ParaLine
  <String ‘A look at the document formats’>
>
> # end of Para
<Para
<PgfTag ‘Body’>
 <ParaLine
  <String ‘The aim of the converter is to convert documents in the
FrameMaker Interchange ’>
 >
 <ParaLine
  <String ‘Format (MIF) to HTML. These two languages differ quite a lot, and
therefore some ’>
 >
 <ParaLine
  <String ‘formatting information is bound to be lost on the way. ’>
 >
 > # end of Para
```

FIGURE 4. HTML paragraphs corresponding to the MIF paragraphs in Figure 3.

```
<H2>A look at the document formats</H2>
The aim of the converter is to convert documents in the FrameMaker
Interchange Format (MIF) to HTML. These two languages differ quite a lot,
and therefore some formatting information is bound to be lost on the way. <p>
```

The reason why the internal format is used instead of converting directly to HTML is that the internal format can combine several HTML tags and thus make a much richer language.

The formatting of the paragraph in HTML is chosen by the user. By editing a tag file, the user chooses between different internal tags. Each paragraph type in FrameMaker can be bound to any of the internal tags.

This method of converting documents makes the conversion process simple since there is no need for the converter to deduce the type of text from font types and similar sources of less exact information. However, this also means that if the user decides not to make use of paragraph styles in FrameMaker, the conversion process is likely to have all the text formatted as simple text.

One of the problems of converting to HTML is that HTML does not have any notion of tabulators and tab stops. FrameMaker can have tabulators at any location, but it would be very difficult to get the same effect in HTML. Tabulators are therefore removed, except in the case of paragraphs bound to the special HTML construct <PRE>. This means that the text will be displayed with a non-proportional character type and that tabulators are placed at even intervals. Tabulators may also be used when the paragraph format is bound to an internal format, which makes use of the tabs explicit. Fm2html uses constructs in HTML, which in some cases can simulate the use of tabulators.

Generating a table of contents

The table of contents is generated on the basis of certain internal converter tags. The FrameMaker generated table of contents is discarded because it is based on page numbers. Page numbers do not have meaning in HTML as the whole text is in one flow.

By editing the tag file, the user of the converter may decide which headings that are included in the table of contents.

Handling and generation of HyperText links

FrameMaker uses HyperText links. Some of these links are page based and are therefore of little interest when creating a HTML document. Others are more generic and they are therefore converted. These include links to named anchors, start and end of a document, as well as links to other documents.

Footnotes in FrameMaker are also converted to HyperText links. The footnotes are moved to the end of the HTML document that is generated. References in FrameMaker are also converted to HyperText links. References are used in FrameMaker to automatically update parts of the text, which reference other parts of the text. As most of these references point to figures, tables and headings (eg the text “see chapter 5”, where “chapter 5” is the reference), it was considered useful to convert them to links.

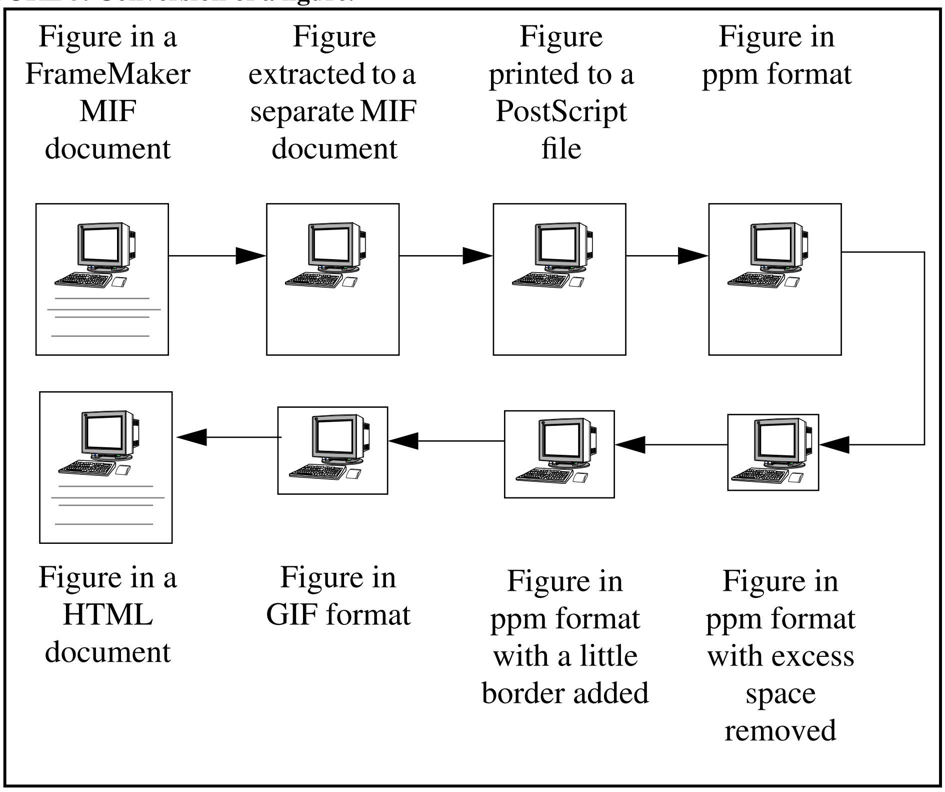
Converting figures

Different versions of FrameMaker (FrameMaker for Unix, FrameMaker for Windows, etc.) may include figures in several different formats. The figures can be drawn using FrameMaker tools or be included. Most WWW viewers can display GIF figures, but other formats may only be displayed by a few of the viewers. This means that all the different figure formats should be translated to GIF, which normally would mean a lot of work finding suitable converter programs. Instead it was decided to use a unified procedure for converting figures. This does not give the best result, but it means that most figures may be converted. As a bonus, the same method may be used to convert mathematical formulas.

Figure 5 shows the process of converting a figure. The process has seven stages:

1. Extract the figure from the document into a separate MIF file.
2. Convert the figure to PostScript format.
3. Convert the figure to the ppm format.
4. Remove excess space from the figure.
5. Add a small border to the figure.
6. Convert the figure to GIF.
7. Include the figure in the resulting HTML document.

FIGURE 5. Conversion of a figure.



The first two steps extract the figure from the document and print it to a PostScript file. This means that all figures are handled in the same manner, without regard to the original format.

In step 3, the figure is converted to ppm format using GhostScript and pstoppm.ps, a script that comes with GhostScript. Having the figure in ppm format is useful as several programs exist to manipulate figures in this format (PbmPlus and NetPbm packages). One of these programs is used to remove excess space from the figure. This is necessary because excess space was introduced in the conversion to PostScript format. Another program is used to add a little border to the figure. Otherwise, the figure may look strange, given that the background colour of the figure is different from the background colour used in the client software. The resulting figure is converted to GIF and included in the HTML document.

There are some problems related to the use of this method. The main one is that GhostScript does not seem to be able to handle all figures well, and may change some figures. However, this does not seem to be a major problem. Most figures are translated correctly. It is a more serious problem that figures which have small parts (for example text in a small font) are not translated well. In some cases, the text may not be readable. This is a consequence of the lower resolution of computer screens compared to that of paper. One solution may be to use more screen space, but that may not always be acceptable. This problem still has to be solved.

The above conversion process will function well as long as the figures can be easily extracted. This is easy for figures in anchored frames since fm2html only needs to recognise the start and end of the frame, and physically remove it from the document. Figures which are not in anchored frames and consequently not located in the text flow are more difficult to handle and they are currently not converted.

As mentioned above, mathematical formulas are handled in the same way as figures. It is easy to extract the contents of a mathematical formula into a separate document. The process of Figure 5 may then be used to convert the formula to a GIF file, which in turn is included in the HTML document. However, this also means that the same problem applies to mathematical formulas as for figures. If the formula has small text, it will not be easily readable.

Converting tables

HTML has no explicit structure for including tables. This is added in the current draft for HTML+. For the time of being, other methods need to be used. Two solutions are possible:

1. Converting the table into a figure.
2. Using the preformatted text option in HTML (mono-spaced text).

The first solution would give an exact representation of the table, but would also include the current limitations of figure conversion. In the worst case, this may lead to the contents of the table not being readable.

The second solution would display the contents of the table in a format that is not optimal, but the contents would be readable. However, figures may not be shown directly since character counting is used to format the table. A link to the figure may be used instead. The second solution is more in the spirit of WWW, as the user of the viewer may choose the font type and size. If the first solution is applied, the user of the viewer has no control. In order to ensure readability, the second solution was chosen.

Unsolved problems

In the current version of fm2html, a number of problems remain unsolved. Some of them may only be solved by extending the HTML language. This includes the problem of special characters. Characters which are available in FrameMaker but not in HTML, include mathematical characters and hence Greek characters. This is considered a major problem since many potential users of fm2html write technical papers that include such characters. Although a solution was found for full mathematical formulas, this solution may not be optimal for single characters since the size of these characters may differ from the size of the main text. HTML+ will include an extended character set that will solve this problem. The converter currently generates HTML+ compatible output, and mathematical characters will therefore be available in HTML+ compatible viewers (provided that HTML+ is not changed).

Another character formatting option that is not available in HTML is super- and subscript. This is likely to be a major problem with technical papers since they tend to use these formatting options extensively. The current solution is replace them with italics, but since HTML+ currently includes both superscript and subscript, this problem should soon be history.

Paragraph formatting, indents, tabulators, multiple columns, etc., are not available in HTML and as long as this is the case, all such information is removed. The only present solution to this problem is to upgrade HTML

The future

Upgrading of the converter will continue in order to extend its value to the users. HTML+ will be supported and the conversion process improved. At present, new features are being added every month. This will continue, at least in the foreseeable future.