A practical introduction to SGML

Michel Goossens and Janne Saarela

CERN, CN Division
CH-1211 Geneva 23
Switzerland
goossens@cern.ch, saarela@cern.ch

Abstract

SGML, the *Standard Generalized Markup Language*, deals with the structural markup of electronic documents. It was made an international standard by ISO in October 1986. SGML soon became very popular thanks in particular to its enthusiastic acceptance in the editing world, by large multi-national companies, governmental organizations, and, more recently, by the ubiquity of HTML, *HyperText Markup Language*, the source language of structured documents on WWW. This article discusses the basic ideas of SGML and looks at a few interesting tools. It should provide the reader with a better understanding of the latest developments in the field of electronic documents in general, and of SGML/HTML in particular.

1 Why SGML?

Since the late eighties we have witnessed an ever quickening transition from book publishing exclusively on paper to various forms of electronic media. This evolution is merely a reflection of the fact that the computer and electronics have made inroads into almost every facet of human activity. In a world in which one has to deal with an ever-increasing amount of data is support of the computer is a particularly welcome alternative, for the preparation of telephone directories, dictionaries, or law texts – to mention just a few examples. In such cases it is not only the volume of the data that is important, but also the need for it to be kept constantly up-to-date.

Once data have been stored in electronic form one can derive multiple products from a single source document. For instance, an address list can be turned into a directory on paper, but it can also be put on cdrom, as a data-base allowing interactive or e-mail access on the Internet or to print a series of labels. Using a set of law texts or a series of articles on history marked up in SGML, one can first publish a textbook containing complete law texts, or a historic encyclopedia, and then provide regular updates or

extract a series of articles on a given subject; one can also offer a consultation service on Internet, via gopher, www or develop a hypertext system on cdrom.

All these applications suppose that the information is not saved in a format that is only suited for printing (for example, WYSIWYG), but that its logical structure be clearly marked.

To recapitulate, the strong points of a generic markup (in SGML) are the following:

- the quality of the source document is improved;
- the document can be used more rationally, resulting in an improved life-cycle;
- the publishing costs are reduced;
- the information can be easily reused, yielding an added value to the document (printed, hypertext, data base).

1.1 The origins of SGML

In order to treat documents electronically it is essential that their logical structure be clearly marked. On top of that, to ensure that documents are really interchangeable, one had to develop a common language to implement this type of representation.

A big step forward was the publication by ISO (the International Standards Organization, with headquarters in Geneva, Switzerland) in October 1986 of SGML as Standard ISO8879 [15]. Because SGML had been officially endorsed by ISO, the Standard was quickly adopted by various national or international organizations and by the large software developers. One can thus be fairly confident that SGML is here to stay and that its role in electronic publishing will continue to grow.

1.2 Who uses SGML?

With the appearance of new techniques and needs linked to the constantly increasing importance of electronic data processing, the traditional way of exchanging documents has been drastically changed. Today, SGML has become an ubiquitous tool for document handling and text processing.

First among the application areas we will consider in which SGML is at present actively used is the work of the American Association of Publishers (AAP). The AAP (see [2] to [4]) selected three types of documents in the field of publishing: a book, a series publication, and an article. For each of these a *document type definition* (DTD, see below, especially Section 4) has been developed. Together, the AAP and the EPS (European Physical Society) have proposed a standard method for marking up scientific documents (especially tables and mathematical documents). This work forms the basis of ISO 12083.

Another application actively developed during the last few years is the CALS (*Computer-aided Acquisition and Logistic Support*) initiative of the American Department of Defense (DoD). This initiative aims at the replacement of paper documents by electronic media for the documentation of all arms systems. The DoD decided that all

documentation must be marked up in SGML, thus also making (the frequent) revisions a lot easier.

A few other examples of the use of SGML are:[1]

- the Publications Office of the European Communities (FORMEX);
- the Association of German editors (Börsenverein des Deutschen Buchhandels);
- the British Library with "SGML: Guidelines for editors and publishers" and "SGML: Guidelines for authors";
- in France, the *Syndicat national de l'édition* and the *Cercle de la librairie*, two associations of French publishers, have defined an application for the French editing world [20];
- the ISO Publishing Department and the British Patents Office (HMSO);
- Oxford University Press and Virginia Polytechnic (PhD, USA);
- the Text Encoding Initiative (classic texts and comments);
- the technical documentation of many major computer manufacturers or scientific publishers, for instance the `DocBook` or other dedicated DTDs used by IBM, HP, OSF, O'Reilly, etc.
- many text processing and data base applications have SGML input/output modules (filters), for example, Frame, Interleaf, Microsoft, Oracle, Wordperfect;
- McGraw-Hill (Encyclopedia of Science and Technology);
- the electronics industry (Pinacle), the aerospace industry and the airlines (Boeing, Airbus, Rolls Royce, Lufthansa, etc.), the pharmaceutical industry;
- press agencies;
- text editors and tools with direct SGML interfaces, such as Arbortext, EBT, Exoterica, Grif, Softquad;
- and, of course, HTML and www!

2 SGML basic principles

SGML is a standard method of representing the information contained in a document independently of the system used for input, formatting, or output.

SGML uses the principle of logical document *markup*, and applies this principle in the form of the definition of a *generalized* markup language. SGML in itself does not define *per se* a markup language, put provides a framework to construct various kinds of markup languages, in other words SGML is a *meta-language*.

2.1 Different types of markup

The "text-processing" systems that have found their way into almost every PC or workstation nowadays are mostly of the WYSIWYG type, i.e., one specifically chooses the "presentation" or "formatting" characteristics of the various textual elements. They

1. See also the "SGML Web Page" at the URL `http://www.sil.org/sgml/sgml.html` for more information on who uses SGML and why.

can be compared to older formatting languages, where specific codes were mixed with the (printable) text of the document to control the typesetting on the micro level. For example, line and page breaks, explicit horizontal or vertical alignments or skips were frequently used to compose the various pages. Generally, these control characters were extremely application-specific, and it was difficult to treat sources marked up in one of these systems with one of the others. On the other hand, this type of markup does a very good job of defining the specific physical representation of a document, and for certain kinds of documents it might be more convenient for obtaining a given layout, in allowing a precise control of line and page breaks. This approach makes viewing and printing documents particularly easy, but re-using the source for other purposes can be difficult, even impossible.

To successfully prepare a document for use in multiple ways it is mandatory to clearly describe its logical structure by eliminating every reference to a physical representation. This is what is understood under the term *logical* or *generic* markup. The logical function of all elements of a document – title, sections, paragraphs, tables, possibly bibliographic references, or mathematical equations – as well as the structural relations between these elements, should be clearly defined.

Figure 1 shows a few examples of marking up the same text. One clearly sees the difference between *specific* markup, where precise instructions are given to the text formatter for controlling the layout (for example, the commands `\vskip` or `.sp`), and *generic* markup, where only the logical function (chapter or beginning of paragraph) is specified.

2.2 Generalized logical markup

The principle of logical markup consists in *marking* the structure of a document, and its definition has two different phases:

1. the definition of a set of "tags" identifying all elements of a document, and of formal "rules" expressing the relations between the elements and its structure (this is the role of the DTD);
2. entering the markup into the source of the document according to the rules laid out in the DTD.

Several document instances can belong to the same document "class", i.e., they are described by the same DTD – in other words they have the same logical structure. As an example let us consider two source texts of an article (see Figure 2), where the specific structures look different, but the logical structure is built according to the same pattern: a title, followed by one or more sections, each one subdivided into zero or more subsections, and a bibliography at the end. We can say that the document instances belong to the *document class* "article".

To describe the formal structure of all documents of type "article" one has to construct the *Document Type Definition* (or DTD). of the document class "article". A DTD is expressed in a language defined by the SGML Standard and identifies all

Specific markup

TEX

```
\vfil\eject
\par\noindent
{\bf Chapter 2: Title of Chapter}
\par\vskip\baselineskip
```

Script

```
.pa
.bd Chapter 2: Title of Chapter
.sp
```

Generic or logical markup

LATEX

```
\chapter{Title of Chapter}
\par
```

HTML (SGML)

```
<H1>Title of Chapter</H1>
<P>
```

Figure 1: Different kinds of markup

the elements that are allowed in a document belonging to the document class being defined (sections, subsections, etc). The DTD assigns a name to each such structural element, often an abbreviation conveying the function of the element in question (for example, "sec" for a section). If needed, the DTD also associates one or more descriptive *attributes* to each element, and describes the relations between elements (for example, the bibliography always comes at end of the document, while sections can, but need not contain subsections). Note that the relations between elements do not always have to be hierarchical, for instance the relation between a section title and a cross-reference to that title three sections further down is not a hierarchical type of relation. In general, DTDs use element attributes to express these kinds of cross-link.

Having defined the DTD one can then start marking up the document source itself (article A or article B), using the "short" names defined for each document element. For instance, with "sec" on form the *tag* `<sec>` for marking the start of a section and `</sec>` to mark its end, and similarly one has `<ssec>` and `</ssec>` for subsection, and so on.

```
    Article A            Article B
    =========            =========
Title                Title
Section 1            Section 1
   Subsection 1.1       Subsection 1.1
   Subsection 1.2       Subsection 1.2
Section 2               Subsection 1.3
Section 3            Section 2
   Subsection 3.1       Subsection 2.1
   Subsection 3.2       Subsection 2.2
   Subsection 3.3
   Subsection 3.4
Bibliography         Bibliography
```

Figure 2: Two instances of the same document class "article"

```
<article>
<tit>An introduction to SGML</tit>
<sec>SGML: the basic principles</sec>
<P>  ...
<ssec>Generalized logical markup</ssec>
<P>  ...
```

2.3 A few words about the DTD

If one wants to apply the latest powerful data processing techniques to electronic documents, using the information about their structure, one must have ways to ensure that they are marked up without mistakes. One must also ensure that the structure of a document instance is coherent: a document must obey the rules laid out for documents of the given document class, according to the DTD for that class.

To fulfil all these aims a DTD defines:

- the *name* of the elements that can be used;
- the *contents* of each element (Section 4.2);
- *how often* and in what order each element can occur (Section 4.2);
- if the begin or end tag can be *omitted* (Section 4.2);
- possible *attributes* and their default values (Section 4.3);
- the name of the *entities* that can be used (Section 4.4).

3 Transmitting the information relative to a document

The aim of SGML is to represent the information contained in a document. Already in Section 2.2 we have explained that SGML operates in two stages to define the structure of a document:

- a declaration phase;
- a utilization phase, where the document source is marked up using declared elements, attributes and entities.

This basic principle is used for the transmission of *all the information related to the document to be exchanged*.

The basic character set is ASCII, as defined by international Standard ISO/IEC 646. One can change the character set by changing this declaration at the beginning of the parsing of the document, when the SGML declaration associated to the DTD is read in (see Appendix B.)

A document can contain symbols or characters that cannot be entered directly on the keyboard, such as Greek letters or mathematical symbols, or even illustrations, photos, or parts of another document. This functionality is implemented through the use of entity references (see Section 4.4).

The markup system is based on a set of delimiters, special symbols, and keywords with special meaning.[2] For instance when "`sec`" identifies the element "Section", then in the document source `<sec>` is the tag marking the beginning of a Section, with the delimiters "<" and ">" indicating, respectively, the tag start and end. Similarly, the formal structure of the document (described by the DTD) has its own language defined by the SGML Standard.

More generally, the SGML Standard does not define once and for all the structure of a document and all elements that it can contain, i.e., the delimiters and special symbols, but merely specifies the construction rules they have to follow. Also, SGML does not fix the markup language, but offers an *abstract syntax*, allowing one to construct particular syntax instances as needed. The Standard proposes an example syntax, called the *reference concrete syntax*, used throughout this article. We can thus safely state that SGML is a *meta-language*.

4 The structure of a DTD

To better understand how SGML works we propose to examine a real example of a modern SGML application, namely HTML level 2, which corresponds to the functionality offered by popular HTML viewing programs, such as Mosaic, Netscape or Lynx. The complete DTD of HTML2 is shown in Appendix A starting on page 76. To make it easier to identify the various parts of the DTD the lines have been numbered.

---

2. These symbols can also be redefined at the beginning of the document

Before starting to parse a DTD the SGML declaration is read in by the parser. For HTML this declaration is shown in Appendix B on page 86. It defines the character set, special characters and option settings used in the DTD and allowed in the document instance. For instance, in the area of markup minimization, the parameter `OMITTAG` (Line 66) has the value `YES`, which allows tag minimization, i.e., under certain circumstances (specified in the DTD) tags can be omitted, as explained in Section 4.2. If, on the other hand, the value is specified as `NO` then tag minimization is disallowed altogether.

The DTD defines all elements, their possible attributes and the entities associated with a given document class (HTML2 in our example).

Inside a DTD the start of a declaration is noted by the sequence "<!" and its termination by '>". Certain sections of a DTD are identified (marked) by a keyword to ensure they are handled correctly, or to (de)activate their contents according to the value of the keyword (`IGNORE` or `INCLUDE`). The notation for the beginning, respectively the end of such a *marked section* is "`<![ keyword [`" and "`]]>`" (see Lines 37–39, and 303–305).

4.1 Comments

It is always a good idea to include comment lines inside document sources or DTDs, whose presence will make them more readable and help in their future maintenance.

An SGML comment has the form:

```
<!-- text of the comment -->
```

The comment is limited by the double hyphen signs, `--`, and can span several lines, as seen, for instance in Lines 1–11 and 28–35.

4.2 The elements

An element declaration

Each element belonging to the logical structure of a document must be declared. This declaration specifies the *name* of the element, as well as, between parentheses, its *content model*, i.e., which elements can or must be part of the element in question.

```
<!ELEMENT name n m (content model)>
```

For instance Lines 614 and 616 are equivalent to the declaration:[3]

```
<!ELEMENT HTML O O  (HEAD, BODY)>
```

The part between the element name "`HTML`" and the content model "`(HEAD, BODY)`" describes the minimization possibilities for the `<HTML>` tag (see "Omitting tags" below). The present declaration specifies that an HTML document contains a "`HEAD`" *followed* by a "`BODY`". Line 533 and the definition of the parameter entity on Lines 548–551 specify further that the document head must contain a "`TITLE`" and can contain a few more elements (`ISINDEX`, `BASE`, `META`, etc).

3. The form used in the DTD at line 616 uses a parameter entity, see Section 4.4.

symbol	*description*
,	all must appear and in the order indicated (ordered "and")
&	all must appear but any order is allowed (unordered "and")
|	one and only one can appear (exclusive "or")
+	element must appear once or more
?	optional element (0 or one)
*	element can appear once or more

Table 1: Order and choice operators

Omitting tags

It is possible that under certain circumstances one can infer automatically from the context that an omitted tag is present. This possibility must be declared for each element between the element's name and its content model in the form of two blank separated characters, corresponding, respectively, to the omittag characteristics of the start and end tag. There are only two possible values, namely a hyphen "-" indicating that the tag *must* be present (cannot be omitted), and an uppercase letter O "O" signifying that it may be omitted. For example, for numbered (OL) and unnumbered (UL) lists and their elements (LI) one has (from Lines 379 and 411, resp.):[4]

```
<!ELEMENT (OL|UL) - -  (LI)+>
<!ELEMENT LI      - O  %flow>
```

The two blank-separated hyphens, "- -", on the first line specify that one must *always* use the begin and end tags for the list declarations (<OL>...</OL> and <UL>...</UL>) while the "- O" on the second line indicate that the end tag for the members of a list (<LI>...) may be omitted.

The contents model

As already mentioned, the content model uses order and choice operators (see Table 1 for a list).

We already encountered the operator of choice (|), which specifies that one of the elements can be present (but not more than one at a time). Let us now turn our attention to another example with a description list (<DL>) as declared on Line 357 as:

```
<!ELEMENT DL    - -  (DT*, DD?)+>
```

This indicates that for a description list the start tag <DL> *and* end tag </DL> must always be present, and that the list can contain one or more occurrences ((...)+) of zero or more <DT> tags (DT*) that can be *followed* (,) by at most one <DD> tag (DD?).

An element with multiple members that can appear in any order is defined on Lines 548–553. These lines essentially stipulate that an HTML head can contain, in any order,

4. The meaning of the symbols | and + is explained in Section 4.2, see especially Table 1; the definition of the parameter entity %flow can be found on Line 313, see also Section 4.2.

a title (TITLE), zero or one <ISINDEX>, <BASE>, and <NEXTID> tags, and zero or more <META> and <LINK>:

```
<!ELEMENT HEAD O O  (%head.content)>
<!ENTITY % head.content
     "TITLE & ISINDEX? & BASE? &
                    (%head.extra)">
<!ENTITY % head.extra
          "NEXTID? & META* & LINK*">
```

An element can contain other elements, characters, or both (in the latter case one speaks of a *mixed content*).

One can specify to the SGML parser the type of characters that can be used. The following reserved names are defined for that purpose:

PCDATA *parsed character data*.
The characters are supposed to have been treated by the parser and can thus no longer contain entity references or tags. For instance, on Line 557 an HTML title is defined as:

```
<!ELEMENT TITLE - -  (#PCDATA)>
```

RCDATA *replaceable character data*.
The parser can expect to find only characters or entity references, i.e., (begin and end) tags are forbidden.

CDATA *character data*.
No further processing is needed by the SGML parser (nevertheless, the data might be processed by another program, for instance PostScript). A telephone number in a letterhead could be declared thus:

```
<!ELEMENT TEL CDATA>
```

ANY The element can contain data of type PCDATA or *any* other element defined in the DTD.

EMPTY The element has an *empty content*. It can, however, be qualified by possible attributes (see Section 4.3). An example of this is the <IMG> tag and its attributes as defined on Lines 233–240.

Certain elements can be used anywhere in the document source. In this case it is convenient to declare them as *included* in the element document. More generally, an element can be contained in the content model of another element and can be part of any of the element's constituents. In this case the syntax +(...) is used. Similarly, one can *exclude* certain elements from the element being defined by using the syntax -(...). For instance, the electronic HTML form is defined on Line 457 as follows:

```
<!ELEMENT FORM - - %body.content
    -(FORM) +(INPUT|SELECT|TEXTAREA)>
```

This states that the <FORM> element can contain everything specified by the parameter entity %body.content (Lines 430, 267, 146, and 309–311). Moreover, all these elements

keyword	*value of attribute*
CDATA	textual data (any characters)
ENTITY(IES)	general entity name(s)
ID	an SGML element identifier
IDREF(S)	value(s) of element identifier reference(s)
NAME(S)	SGML name(s)
NMTOKEN(S)	nominal lexical token(s)
NOTATION	notation name
NUMBER(S)	number(s)
NUTOKEN(S)	numeric lexical token(s)

Table 2: Keywords for attribute types

can contain, *at any level* the tags <INPUT>, <SELECT>, or <TEXTAREA>. On the other hand, forms are not recursive, since the <FORM> tag cannot contain itself (-(FORM)).

4.3 Attributes

All possible attributes of all elements in a DTD must be explicitly declared in the same DTD. For reasons of clarity and convenience, attribute declarations normally immediately follow the declaration of the element they refer to.

An attribute declaration consists of:

- the name of the element(s) that it refers to;
- the name of the attribute;
- either the *attribute type*, specified as one of the keywords shown in Table 2, or, between parentheses, the list of values the attribute can take;
- a default value (one of the possible values specified between quotes, or one of the keywords shown in Table 3).

An attribute declaration thus takes the following form:

```
<!ATTLIST element_name
      attribute_1 (values) "default"
      attribute_2 (values) "default"
      ...                          >
```

For instance, the list declaration (<DL>) (Lines 357–362) defines an attribute "compact" to indicate that the members of a list should be typeset more densely.

```
<!ATTLIST DL COMPACT (COMPACT) #IMPLIED
```

This declaration specifies that the only possible value is COMPACT and that the system (the parser) will provide a default value (#IMPLIED, see Table 3).

One might also wish to specify numeric information, for instance, the <PRE> tag (Lines 317–320) has an attribute to specify the width of the line:

```
<!ATTLIST PRE WIDTH NUMBER #implied
```

keyword	*description*
`#FIXED`	The attribute has a fixed value and can take only that value.
`#REQUIRED`	The value is mandatory and must be specified by the use.
`#CURRENT`	If no value is specified, then the default value will be the the last specified value.
`#CONREF`	The value will be used for cross-references.
`#IMPLIED`	If no value is specified, the parser will assign a value.

Table 3: Keywords for attribute default values

The attribute type is an "(integer) number" (keyword: `NUMBER`) and if no value is specified then the parser will supply a default (`#implied`).

As a last example let us once more look at the element `<IMG>` (image) and its attributes (Lines 234–240), whose definitions correspond essentially to the following declaration:

```
<!ATTLIST IMG
     SRC   %URI;                 #REQUIRED
     ALT   CDATA                 #IMPLIED
     ALIGN (top|middle|bottom) #IMPLIED
     ISMAP (ISMAP)               #IMPLIED
     ....
```

The first line references the parameter entity `%URI` (see Lines 73–84) that defines a *Uniform Resource Identifier*. This attribute is *mandatory* (`#REQUIRED`). The other attributes are optional and have a system-defined default value (`#IMPLIED`). In the case of the alignment attribute (`ALIGN`) a choice of any of three values if possible.

4.4 Entities

Entities can be used for the following purposes:

- The definitions of abbreviated notations to ease repetitive text strings (general entities); for example,
  ```
  <!ENTITY TUG "\TeX{} Users Group">
  ```
- The definition of notations to input special characters, accents or symbols (general character entities). An example of character entities can be found on Lines 102–105;
  ```
  <!ENTITY amp CDATA "&#38;"
                -- "&" (ampersand) -->
  ```
  ISO has defined several standard character entity sets, for instance, for national characters (see Appendix D), graphical symbols, mathematics, etc.
- The inclusion of external files (external entities).
- The definition of variables in a DTD (parameter entities).

It is important to note that, contrary to element and attribute names, which are case insensitive and can be specified in upper, lower, or mixed case, entity names are *case-sensitive*, and one must take care to specify them precisely as they are defined.

General entities are declared in the DTD. An entity declaration first specifies a symbolic name for the entity, followed by its contents. The latter can contain tags, entity references, etc., that will be interpreted when the entity is expanded.

To refer to an entity one makes use of an *entity reference*, which takes the form:

```
&entity_name;
```

For example, if one wants to use the entity "TUG" defined above, one should type in the document source the string of characters `&TUG;` and the parser replaces this by the string "TEX Users Group".

The data associated with an entity can be in another (external) file (*external* entity). This kind of entity can be used to include in the source document being parsed a table or figure (or any kind of data) that was prepared with another application. Instead of including the complete contents of the file in the declaration, one merely specifies the name of the file where the data is stored. The filename must be preceded by the keyword `"SYSTEM"`, for example, for the unix operating system one might have a declaration of the form:

```
<!ENTITY article SYSTEM
    "/usr/goossens/tug/sgmlart.sgml">
```

Inside a DTD one frequently uses *parameter* entities that allow one to considerably increase the modularity of the definition of the various elements defined in the DTD. Simple examples are (Lines 89, 91, 175);

```
<!ENTITY % heading "H1|H2|H3|H4|H5|H6">
<!ENTITY % list " UL | OL | DIR | MENU " >
<!ENTITY % text "#PCDATA | A | IMG | BR">
```

These entities are used, for instance, on Lines 212, 267, 430.

```
<!ELEMENT ( %heading )  - -  (%text;)+>
```

4.5 Other DTDs

In order to get a better idea of what DTDs for more complex documents look like, we shall briefly discuss the HTML3, `DocBook` and ISO12083.

HTML3

As it name indicates, HTML3 is a successor to the present HTML Standard (also know as HTML2, and discussed in detail in the previous sections). HTML3[5] builds upon HTML2 and provides full backwards compatibility. *Tables* have been one of the most requested features; HTML3 proposes a rather simple table model that is suitable for rendering on a very wide range of output devices, including braille and speech synthesizers.

---

5. See URL `http://www.hpl.hp.co.uk/people/dsr/html/CoverPage.html`.

Inline figures are available and provide for client-side handling of hot zones whilst cleanly catering for non-graphical browsers. Text can flow around figures and full flow control for starting new elements is possible.

Mathematics support for equations and formulae in HTML3 mainly uses TEX's box paradigm. The implementation uses a simple markup scheme, that is still powerful enough to cope with over 90% of the most common cases. Filters from TEX and other word processing systems will allow one to easily convert existing sources into HTML3.

As HTML is most often used to present information on-screen, it is important to allow some positioning control for the various elements in a document. Therefore, HTML3 includes support for customized lists; fine positioning control with entities like `&emspace;`, horizontal tabs, and alignment of headers and paragraph text.

As well as this, many other often-requested features have been included, most notably a style-sheet mechanism, which counters the temptation to continually add more presentation features by giving the user almost full control over document rendering, and taking into account the user's preferences (window size, resource limitations such as availability of fonts)

The HTML3.0 Internet draft specification is being developed by the IETF (Internet Engineering Task Force) taking into account the following guidelines:

- interoperability and openness;
- simplicity and scalability;
- platform independence;
- content, not presentation markup;
- support for cascaded style sheets, non-visual media, and different ways of creating HTML.

To illustrate the use of this DTD one can look at the table and mathematics parts of the HTML3 DTD (see Appendix E) and at the markup examples and the generated output (Figures 4 and 6).

DocBook

The DocBook DTD[6] defines structural SGML markup for computer documentation and technical books. It is supported by the Davenport Group, an association of software documentation producers established to promote the interchange and delivery of computer documentation using SGML and other relevant standards.

The primary goal in developing the DTD was to filter existing software documentation into SGML. It describes the structures the collaborators of the Davenport group and other producers and consumers of software documentation have encountered in processing large bodies of documentation. The `DocBook` DTD uses a book model for the documents. A book is composed of book elements such as Prefaces, Chapters, Appendices, and Glossaries. Five section levels are available and these may contain paragraphs, lists, index entries, cross references and links.

---

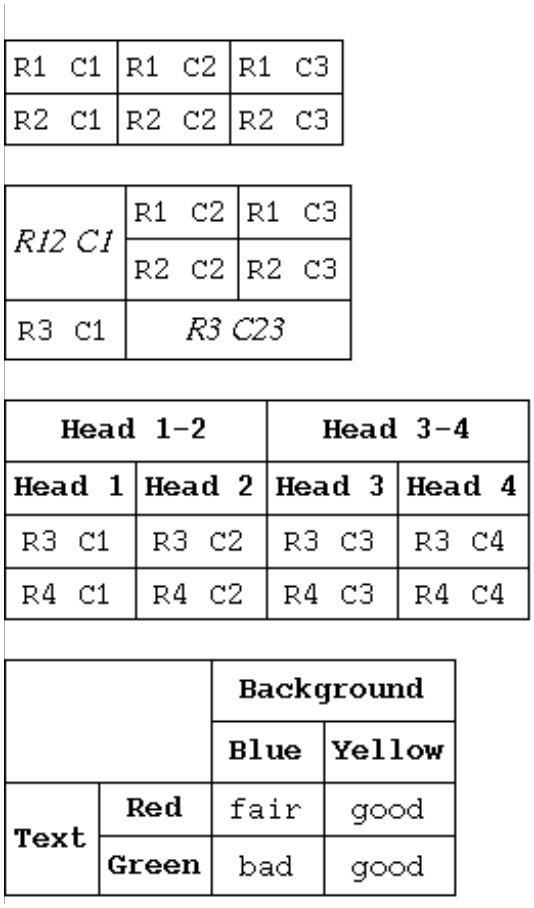
6. See URL `ftp://ftp.ora.com/pub/davenport/docbook/fullguide.sgm`.

```
<TABLE BORDER>
<TR>  <TD>R1 C1</TD><TD>R1 C2</TD><TD>R1 C3</TD>
</TR>
<TR>  <TD>R2 C1</TD><TD>R2 C2</TD><TD>R2 C3</TD>
</TR>
</TABLE>

<TABLE BORDER>
<TR>  <TD ROWSPAN=2><EM>R12 C1</EM></TD>
      <TD>R1 C2</TD><TD>R1 C3</TD>
</TR>
<TR>  <TD>R2 C2</TD><TD>R2 C3</TD>
</TR>
<TR>  <TD>R3 C1</TD><TD COLSPAN=2><EM>R3 C23</EM></TD>
</TR>
</TABLE>

<TABLE BORDER>
<TR>  <TH COLSPAN=2>Head 1-2</TH>
      <TH COLSPAN=2>Head 3-4</TH>
</TR>
<TR>  <TH>Head 1</TH><TH>Head 2</TH>
      <TH>Head 3</TH><TH>Head 4</TH>
</TR>
<TR>  <TD>R3 C1</TD><TD>R3 C2</TD>
      <TD>R3 C3</TD><TD>R3 C4</TD>
</TR>
<TR>  <TD>R4 C1</TD><TD>R4 C2</TD>
      <TD>R4 C3</TD><TD>R4 C4</TD>
</TR>
</TABLE>
<P>
<TABLE BORDER>
<TR>  <TH COLSPAN=2 ROWSPAN=2></TH>
      <TH COLSPAN=2>Background</TH>
</TR>
<TR>  <TH>Blue</TH><TH>Yellow</TH>
</TR>
<TR>  <TH ROWSPAN=2>Text</TH>
      <TH>Red</TH><TD>fair</TD><TD>good</TD>
</TR>
<TR>  <TH>Green</TH><TD>bad</TD><TD>good</TD>
</TR>
</TABLE>
```

Figure 3: HTML3 example of tables (source)



R1 C1	R1 C2	R1 C3
R2 C1	R2 C2	R2 C3

R12 C1	R1 C2	R1 C3
	R2 C2	R2 C3
R3 C1	*R3 C23*	

Head 1-2		**Head 3-4**	
Head 1	**Head 2**	**Head 3**	**Head 4**
R3 C1	R3 C2	R3 C3	R3 C4
R4 C1	R4 C2	R4 C3	R4 C4

		Background	
		Blue	**Yellow**
Text	**Red**	fair	good
	Green	bad	good

Figure 4: HTML3 example of tables (result with the `Mosaic` browser)

The DTD also leaves room for localizations. The user of the DTD is free to give own content models for appendixes, chapters, equations, indexes, etc.

The AAP effort and ISO 12083

The American Association of Publishers (AAP) has been working since the publication of the SGML Standard in 1985 on promoting SGML as an electronic standard for manuscript preparation. This document, developed over several years as the "AAP Standard," was later promoted to by the Electronic Publishing Special Interest Group (EPSIG) and the AAP as "the Electronic Manuscript Standard," and is now a NISO (National Information Standards Organization) publication. The AAP/EPSIG application is SGML-conforming, and provides a suggested tag set for authors and publishers. It defines the format syntax

```
<!DOCTYPE html PUBLIC
     "-//IETF//DTD HTML 3.0//EN//">
<HTML>
<TITLE>A Math Sampler</TITLE>
<BODY>
<H1>Formulae by examples</H1>
<MATH>x<SUP>I</SUP>y<SUP>J</SUP>
      z<sup align=center>K</sup> 
   <BOX>(<LEFT>1 + u<OVER>v<RIGHT>)</BOX>
</MATH>
<P><MATH><BOX>[<LEFT>x + y<RIGHT>]</BOX> 
         <BOX>(<LEFT>a<RIGHT>]</BOX> 
         <BOX>||<LEFT>b<RIGHT>||</BOX></MATH>
<P><MATH>int<SUB>a</SUB><SUP>b</SUP>
    <BOX>f(x)<over>1+x</BOX> 
sin ( x<SUP>2</SUP>+1) dt</MATH>
<P><MATH>
   <box>d&sigma;<over>d&epsi;</box>
  =<box>2&pi;Zr<sub>0</sub><sup>2</sup>m
        <over>&beta;<sup>2</sup>(E-m)</box>
    [<box>(&gamma;-1)<sup>2</sup>
        <over>&gamma;<sup>2</sup></box>
     +<box>1<over>&epsi;</box>]
</MATH>
</BODY>
</HTML>
```

Figure 5: HTML3 example of simple mathematics (source)

of the application of SGML publication of books and journals. The Standard achieves two goals. First, it establishes an agreed way to identify and tag parts of an electronic manuscript so that computers can distinguish between these parts. Second, it provides a logical way to represent special characters, symbols, and tabular material, using only the ASCII character set found on a standard keyboard.

For several years the AAP and the EPS (European Physical Society) have been working on a standard method for marking up scientific documents. There work has been the basis for International Standard ISO 12083, the successor to the AAP/EPSIG Standard, and four DTDs have been distributed by EPSIG as the "ISO" DTDs.[7]

7. They can be found at the URL `http://www.sil.org/sgml/gen-apps.html\#iso12083DTDs`.

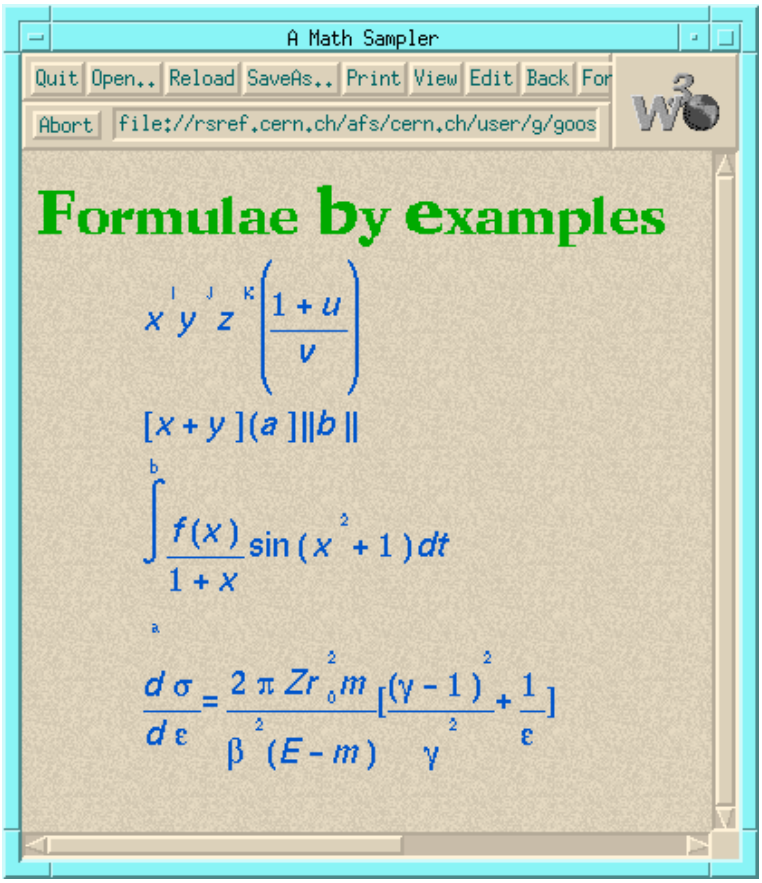


Figure 6: HTML3 example of simple mathematics (result with the `arena` browser)

This DTD has a basic book structure consisting of chapters, sections and subsections down to six levels. The mathematics part is, however, of some interest since it can be compared to HTML3.

The ISO 12083 table model

The ISO 12083 table model consists of the following elements (see Figure 7 for the relevant part of the DTD):

`<table>` the table element;
`<np>` number;
`<title>` title;
`<tbody>` table body;

```
<!-- ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ -->
<!--    Tables                                                    -->
<!-- ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ -->

<!ELEMENT table        - -  (no?, title?, tbody)      -(%i.float;) >
<!ELEMENT tbody        - O  (head*, tsubhead*, row*)               >
<!ELEMENT row          - O  (tstub?, cell*)                        >
<!ELEMENT tsubhead     - O  %m.ph;                                 >
<!ELEMENT (tstub|cell) - O  %m.pseq;                               >
```

Figure 7: Part of the ISO 12083 DTD relating to simple tables

.

`<head>`	head;
`<tsubhead>`	table subhead;
`<row>`	row;
`<tstub>`	table stub;
`<cell>`	cell.

This table model does not support spanning rows or columns. It does, however, support subhead elements that can be used to give more granularity to the table contents. An example of a marked-up table is shown below.

```
<table>
<no>1<title>Capitals in Europe
<tbody>
<row><cell>Helsinki<cell>Finland
<row><cell>Rome<cell>Italy
<row><cell>Bern<cell>Switzerland
</table>
```

Only the simple table model discussed above is part of the basic ISO 12083 DTD as distributed. There also exists a complex table model [3] that allows the user to treat more complex tabular material.

The ISO 12083 mathematics model

The mathematics model in ISO 12083 consists of the following element categories:

character transformations
`<bold>`, `<italic>`, `<sansser>`, `<typewrit>`, `<smallcap>`, `<roman>`;

fractions
`<fraction>`, `<num>`, `<den>`;

superiors, inferiors
`<sup>`, `<inf>`;

embellishments
<top>, <middle>, <bottom>;

fences, boxes, overlines and underlines
<mark>, <fence>, <post>, <box>,
<overline>, <undrline>;

roots
<radical>, <radix>, <radicand>;

arrays
<array>, <arrayrow>, <arraycol>,
<arraycel>;

spacing
<hspace>, <vspace>, <break>, <markref>;

formulas
<formula>, <dformula>, <dformgrp>.

The model has basically the same elements as the HTML3 model, but is more visual. Emphasis is on creating fences at the right places inside a formula, whereas the HTML3 model uses <left> and <right> elements. A simple example is:

```
<formula>
  S = &sum;<inf>n=1</inf><sup>10</sup>
      <fraction>
        <num>1</num>
        <den>
          <radical>3<radix>n</radical>
        </den>
      </fraction>
</formula>
```

The complete DTD is shown in Appendix F, which shows the file `math.dtd` that is part of the ISO 12083 DTD set.

5 SGML editors

Several solutions exist to enter SGML or HTML markup into a document, but an editor that is SGML-aware is probably the best solution. Several (mostly commercial) products exist (see [16], [17], and [18]), but in the remaining part of this section we shall have a look at a public domain solution based on the Emacs editor with the `psgml` application and on the Grif-based Symposia editor.

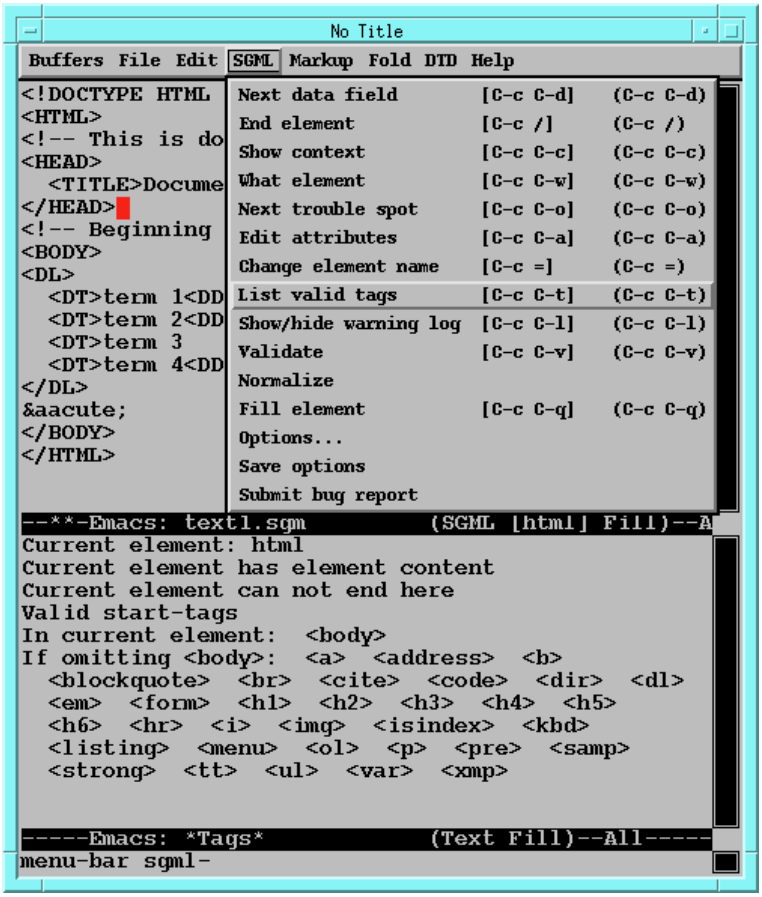


Figure 8: Emacs in `psgml` mode

5.1 Emacs and PSGML

A major mode for editing SGML documents, `psgml`[8], works with the latest versions of gnu Emacs. It includes a simple SGML parser and accepts any DTD. It offers several menus and commands for inserting tags with only the contextually valid tags, identification of structural errors, editing of attribute values in a separate window with information about types and defaults, and structure-based editing.

Figure 8 shows the first HTML test example, to be discussed later (see example `test1.html` in Section 6.2). Both the `psgml` mode and the `nsgmls` program, discussed below, use a catalog file whose structure is defined by the SGML Open consortium to

8. The `psgml` home page is at the URL `http://www.lysator.liu.se/projects/about_psgml.html`.

```
ESC C-SPC   sgml-mark-element
ESC TAB     sgml-complete
ESC C-t     sgml-transpose-element
ESC C-h     sgml-mark-current-element
ESC C-@     sgml-mark-element
ESC C-k     sgml-kill-element
ESC C-u     sgml-backward-up-element
ESC C-d     sgml-down-element
ESC C-b     sgml-backward-element
ESC C-f     sgml-forward-element
ESC C-e     sgml-end-of-element
ESC C-a     sgml-beginning-of-element
C-c C-u     Prefix Command
C-c RET     sgml-split-element
C-c C-f     Prefix Command
C-c C-w     sgml-what-element
C-c C-v     sgml-validate
C-c C-t     sgml-list-valid-tags
C-c C-s     sgml-unfold-line
C-c C-r     sgml-tag-region
C-c C-q     sgml-fill-element
C-c C-p     sgml-parse-prolog
C-c C-o     sgml-next-trouble-spot
C-c C-n     sgml-up-element
C-c C-l     sgml-show-or-clear-log
C-c C-k     sgml-kill-markup
C-c C-e     sgml-insert-element
C-c C-d     sgml-next-data-field
C-c C-c     sgml-show-context
C-c C-a     sgml-edit-attributes
C-c =       sgml-change-element-name
C-c <       sgml-insert-tag
C-c /       sgml-insert-end-tag
C-c -       sgml-untag-element
C-c #       sgml-make-character-reference
```

Figure 9: Emacs key-bindings with `psgml`

locate the SGML declarations and DTDs (see Appendix C). Thanks to the name of the DTD declared on the `<!DOCTYPE>` declaration and that catalog file, `psgml` loads the HTML2 DTD into memory and can then handle the HTML source file. In the Figure, all the elements that can occur at the position of the pointer are shown. Figures 9 shows the more important key combinations for quickly calling some functions. For instance, the sequence `C-c C-t` (`sgml-list-valid-tags`) was used to obtain the list in the lower part of Figure 8. As a last technical (but important) detail, in order to function properly, two variables should be defined in the `psgml` initialization file `psgml.el`, namely `sgml-system-path`, a list of directories used to look for system identifiers, and `sgml-public-map`, a mapping from public identifiers to file names.[9]

5.2 Symposia

At the Third International World Wide Web Conference "Technology, Tools and Applications"[10], which took place in Darmstadt, Germany, from 10 - 13 April 1995, Vincent Quint and collaborators discussed their authoring environment for SGML texts in general, and HTML on WWW in particular.[11] Their approach is based on the Grif editor, which can work with any DTD. They announced that a version with the HTML3 DTD will be made available freely under the name of Symposia. Grif (and Symposia) allow the user to enter text in a wysywig way, but entered elements are validated against the DTD. An example is given in Figure 10, which shows us to be in insert mode in the first column on the first row of the table, where we input the word "text", whilst Figure 11 shows the generated SGML(HTML) source, hidden from the user, but available for any kind of treatment that one would like to do on the document.

6 SGML utilities

As SGML is now actively used in many applications in the field of document production (see Section 1.2 and [17]) several commercial and publicly available solutions are now available to increase the productivity, user-friendliness, and ease of using SGML systems. This section reviews a few of the more interesting publicly available tools.

6.1 Validating an SGML document with NSGMLS

It is often important and useful to be able to validate an SGML (and hence HTML) document. This can, for instance, be achieved with the publicly available SGML parser

9. See the documentation coming with `psgml` for more details.
10. An overview of the papers is at the URL `http://www.igd.fhg.de/www/www95/papers/`.
11. Their paper is available at the URL `http://www.igd.fhg.de/www/www95/papers/84/EditHTML.html`.

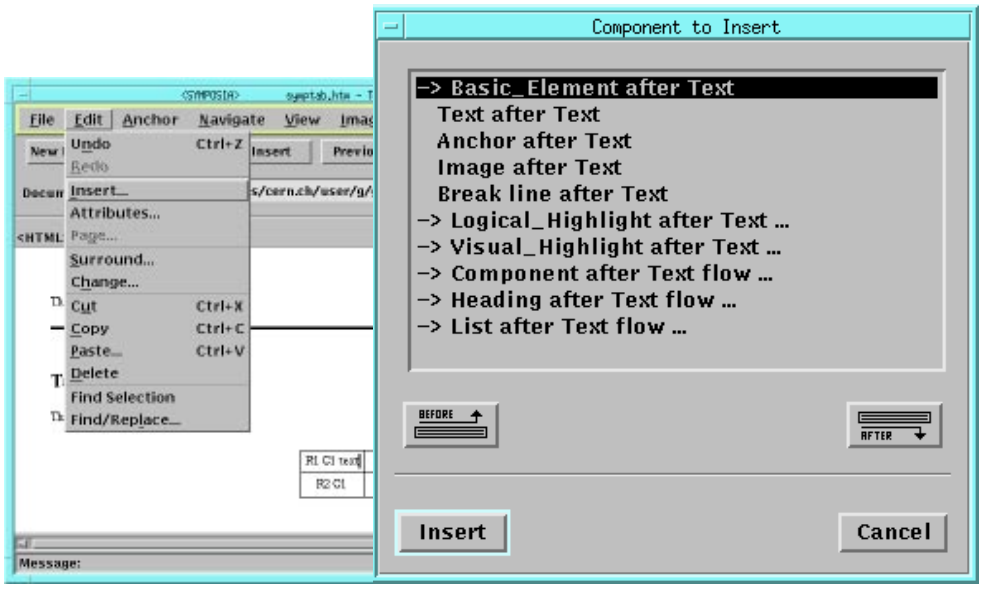


Figure 10: Inserting text in an SGML document with Symposia

Figure 11: SGML source of the document shown in Figure 10

`nsgmls`, which is part of sp[12], a system developed by James Clark (`jjc@jclark.com`), and a successor to his older `sgmls`[13] and `arcsgml`, written by Charles Goldfarb, who is considered by many as the father of SGML, and who is also the author of "The SGML Handbook" [5] describing the SGML Standard in great detail, a reference work that every serious SGML user should possess.

The `nsgmls` parser can be called with the syntax:

```
nsgmls [ -deglprsuvx ] [ -alinktype ]
       [ -ffile ] [ -iname ] [ -mfile ]
       [ -tfile ] [ -wwarning_type ]
       [ filename... ]
```

`nsgmls` needs at least four files to run:

- the catalog file, which describes how the SGML file's `<!DOCTYPE>` declaration is mapped to a filename (see below);
- the SGML declaration, defining the character set used by subsequent files, and the sizes of various internal limits, such as the permitted length of identifiers, as well as what features of SGML are used, such as tag minimization (see the start of Section 4 on page 41 and Appendix B);
- the DTD for the document type;
- an SGML or HTML document instance.

6.2 The <!DOCTYPE> declaration

The `<!DOCTYPE>` declaration has three parameters, as shown in the following example.

```
<!DOCTYPE html PUBLIC
     "-//IETF//DTD HTML//EN">
```

The first parameter specifies the name of the document class according to which the document instance (the user's source file) is marked up. The second parameter is either `SYSTEM` or `PUBLIC`. With the `SYSTEM` keyword the next parameter contains the filename of the DTD, but since actual filenames are system-dependent, this syntax should be discouraged in favour of the `PUBLIC` keyword. In this case, the whereabouts of the DTD are defined via an external entity reference. The SGML Standard does not itself define how the mapping between this entity reference and an external file is defined, but SGML

---

12. sp is available at the URL `http://www.jclark.com/sp.html`. For more information about other publicly available SGML software, have a look at the the public SGML software list at the URL `http://www.sil.org/sgml/publicSW.html`. More generally, on the SGML Web Page at `http://www.sil.org/sgml/sgml.html` one finds entry points to all the above, plus many examples of DTDs, more information about SGML, Hytime, DSSSL, etc.

13. `smgls` is written in highly portable C code, whilst `nsgmls` is C++ with extensive template use, which limits the portability and makes the installation of the latter somewhat more complicated. Also the executable module of `sgmls` is about half the size of the one of `nsgmls`. See the comments of Nelson Beebe at the URL `http://www.math.utah.edu/~beebe/sp-notes.html` for the current situation with implementing `nsgmls` on several architectures.

Open has proposed the format of a catalog file in which those mappings are specified. A few examples are shown below.

```
PUBLIC  "-//IETF//DTD HTML//EN"
    /usr/goossens/sgml/dtds/html.dtd
PUBLIC "ISO 12083:1994//DTD Math//EN"
    /usr/joe/dtds/math.dtd
PUBLIC  "-//IETF//ENTITIES Latin 1//EN"
    /use/joe/sgml/dtds/iso-lat1.sgm
```

The first string following the keyword `PUBLIC` is called a "public identifier", a name which is intended to be meaningful across systems and different user environments. Formally a public identifier is composed of several fields, separated by a double solidus, "//". The first part is an "owner identifier" (the first and third entries have a hyphen, -, meaning that these identifiers were not formally registered, and the organization who created the file was the IETF (the Internet Engineering Task Force); the second entry carries an ISO owner identifier. The second part of the public identifier (following the double solidus), is called the "text identifier". The first word indicates the "public text class" (for example, `DTD` and `ENTITIES`), and is followed by the "public text description" (`HTML`, `Latin 1`, etc.), then, optionally, after another double solidus one finds the "public text language", a code from ISO Standard 639 ([9] – `EN`, for English in our case), and this can be followed by a "display version", if needed.

The final element is the filename associated with the public identifier specified in the second field.

HTML examples

It is not our intention to describe the various options of this program in detail, but we shall limit ourselves to showing, with the help of a few simple examples, how this interesting tool can be used.

```
<!DOCTYPE html PUBLIC
     "-//IETF//DTD HTML 2.0//EN">
<HTML>
<!-- This is document test1.html -->
<HEAD>
  <TITLE>Document test1.html</TITLE>
</HEAD>
<!-- Beginning of body of document -->
<BODY>
<DL>
  <DT>term 1<DD>data 1
  <DT>term 2<DD>data 2
  <DT>term 3
  <DT>term 4<DD>data 4<DD>data 4 bis
```

```
</DL>
&aacute;
</BODY>
</HTML>
```

Presenting this document to `nsgmls` one obtains the following output in the "Element Structure Information Set" (ESIS) format.

```
> nsgmls -m catalog sgml.decl test1.html
#SDA
AVERSION CDATA -//IETF//DTD HTML 2.0//EN
ASDAFORM CDATA Book
(HTML
(HEAD
ASDAFORM CDATA Ti
(TITLE
-Document test1.html
)TITLE
)HEAD
(BODY
ACOMPACT IMPLIED
ASDAFORM CDATA List
ASDAPREF CDATA Definition List:
(DL
ASDAFORM CDATA Term
(DT
-term 1
)DT
ASDAFORM CDATA LItem
(DD
-data 1\n
)DD
ASDAFORM CDATA Term
(DT
-term 2
)DT
ASDAFORM CDATA LItem
(DD
-data 2\n
)DD
ASDAFORM CDATA Term
(DT
```

```
-term 3\n
)DT
ASDAFORM CDATA Term
(DT
-term 4
)DT
ASDAFORM CDATA LItem
(DD
-data 4
)DD
ASDAFORM CDATA LItem
(DD
-data 4 bis
)DD
)DL
-\n\|[aacute]\|
)BODY
)HTML
C
```

As it should, `nsgmls` parses this program without problems, and shows the different elements it encounters in ESIS format. The meaning of the most common output commands generated by `nsgmls` is as follows.

`\\` a \;

`\n` a *record end*;

`\|` brackets internal `SDATA` entities;

`\nnn` character whose octal code is `nnn`;

`(gi` start of element whose generic identifier is `gi`, attributes for this element are specified with `A` commands;

`)gi` end of element whose generic identifier is `gi`;

`-data` data;

`&name` reference to external data entity `name`;

`Aname val` next element has an attribute `name` with specifier and value `val` (see Tables 2 and 3)

`#text` application information (can only occur once);

`C` signals that the document was a conforming document. It will always be the last command output.

For incorrect documents `nsgmls` shows an error:

```
<!DOCTYPE html PUBLIC
     "-//IETF//DTD HTML//EN">
```

```
<HTML>
<BODY>
  <P>text inside a paragraph
</BODY>
</HTML>
```

If we present this document to `nsgmls` (placing the HTML DTD shown in the appendix at the beginning of the file) one obtains:

```
> nsgmls -m catalog sgml.decl test2.html
test2.html:4:6:E: \
           element 'BODY' not allowed here
test2.html:7:7:E: \
  end tag for 'HTML' which is not finished
#SDA
AVERSION CDATA -//IETF//DTD HTML 2.0//EN
ASDAFORM CDATA Book
(HTML
(BODY
-
ASDAFORM CDATA Para
(P
-text inside a paragraph
)P
)BODY
)HTML
```

Note that `nsgmls` indicates at the fourth line that a `<BODY>` tag cannot be used at that particular point (since no mandatory `<HEAD>` element – Line 614 of DTD – was specified). Then, after reading the last (seventh) line containing the `</HTML>` tag, `nsgmls` complains that the HTML document (enclosed inside `<HTML>` tags) is not yet finished.

```
<!DOCTYPE html PUBLIC
     "-//IETF//DTD HTML//EN">
<HTML>
<HEAD>
<TITLE>title</TITLE>
</HEAD>
<BODY>
<LI>
</BODY>
</HTML>
```

Those only interested in checking the syntax of a document can run nsgmls with the -s option, so that it will only print the error messages, as with the incorrect HTML file above.

```
> nsgmls -s -m catalog sgml.decl test3.html
test3.html:8:4:E: \
            element 'LI' not allowed here
```

nsgmls does not complain until Line 8, where an isolated list member <LI> is found. As this is not correct according to the DTD, nsgmls signals its disagreement by stating that the <LI> tag is not allowed at that point (Lines 379 and 394 of the DTD state that list member elements of type <LI> can only be used in lists of type <OL>, <UL>, <MENU>, and <DIR>).

6.3 Prettyprinting

Nelson Beebe (beebe@math.utah.edu) has developed a program htmlpty[14], written in the lex and C languages, to prettyprint HTML files. Its calling sequence is:

```
htmlpty [-options] [file(s)]
```

where the more interesting options are:

-f filename	name output file in comment banner;
-h	display usage summary;
-i nnn	set indentation to nnn spaces per level;
-n	no comment banner;
-w nnn	set output line width to nnn.

The program was run on file test1.html with the result shown below.

```
> html-pretty -i2 -n test1.html
<!DOCTYPE html PUBLIC
     "-//IETF//DTD HTML//EN">
<HTML>
  <!-- This is document doc1.sgm -->
  <HEAD>
    <TITLE>
      Document test HTML
    </TITLE>
  </HEAD>
  <!-- Beginning of body of document -->
  <BODY>
    <DL>
      <DT>
```

14. It is at URL ftp://ftp.math.utah.edu/pub/misc/htmlpty-x.yy.trz (choose the latest version x.yz offered).

```
          term 1
        </DT>
        <DD>
          data 1
        </DD>
        <DT>
          term 2
        </DT>
        <DD>
          data 2
        </DD>
        <DT>
          term 3
        </DT>
        <DT>
          term 4
        </DT>
        <DD>
          data 4
        </DD>
        <DD>
          data 4 bis
        </DD>
      </DL>
      &aacute;
    </BODY>
  </HTML>
```

The program `html-pretty` applies heuristics to detect, and often correct, common HTML errors. It can turn a pure ASCII file into a syntactically-valid HTML file that may then only require a small amount of additional markup to indicate required line breaks.

6.4 SGML document analysis tools

Earl Hook (`ehood@convex.com`) has developed a set of tools `perlSGML`[15], based on the `perl` language. They permit the analysis of SGML documents or DTDs.

`dtd2html` produces an HTML document starting from an SGML DTD that permits an easy hypertext navigation through the given DTD;

`dtddiff` compares two DTDs and shows possible differences;

`dtdtree` shows visually the hierarchical tree structure characterizing the relations between the various elements of a DTD;

15. This system can be found at the url `ftp://ftp.uci.edu/pub/dtd2html`.

`stripsgml` strips a text from its SGML markup, and attempts to translate entity references by standard ASCII characters.

Let us first look at the `dtdtree` utility. When treating the HTML2 DTD, one obtains a visual representation that is very useful for understanding the relations that exist between the various HTML elements. For each element one explicitly sees the elements it can contain. Three points "..." indicate that the contents of the element has been shown previously. Lines containing entries between brackets signal a list of elements that can be included in – `(I)` and `(Ia)` – or are excluded from – `(X)` and `(Xa)` – the content model of the element. Figure 12 shows in four columns the (condensed) output generated by the `dtdtree` program when treating the HTML2 DTD. For more clarity most of the repeated blocks have been eliminated and replaced by the string `*|**|**|` at the beginning of a line and a few lines have been cut to make them fit (marked with `***` at the end of the line).

Documenting a DTD

To document a DTD (and hence a particular SGML language instance) one can use the `dtd2html` utility, which generates, starting from the DTD in question and a file describing all document elements, a hypertext representation (in HTML) of all SGML language elements present in the DTD. This representation makes it easier for users of an SGML-based documentation system to obtain the information relating to an element they need for marking up their document. For example, in the case of HTML2, Figure 13 shows the representation as viewed by the HTML browser `mosaic`.

6.5 Searching and index entries

A search engine for regular expressions for use with the HTML2 DTD is available[16] (Figure 14), as well as an index with more than 1100 entries and phrases[17] (Figure 15).

Checking an HTML document

For those who do not have `sgmls` or `nsgmls` installed there exists a set of programs `htmlchek`[18], including heuristic checkers for common style and grammar violations. The programs are available in both `perl` and `awk` versions and syntactically check HTML2 and HTML3 files for a number of possible errors; they can perform local link cross-reference verification, and generate a rudimentary reference-dependency map.

`htmlchek` checks an HTML file for errors, and giving warnings about possible problems;

16. `http://hopf.math.nwu.edu/html2.0/dosearch.html`.
17. `http://hopf.math.nwu.edu/html2.0/docindex.html`.
18. The documentation is at the URL `http://uts.cc.utexas.edu/~churchh/htmlchek.html` and the tar file at `ftp://ftp.cs.buffalo.edu/pub/htmlchek/`.

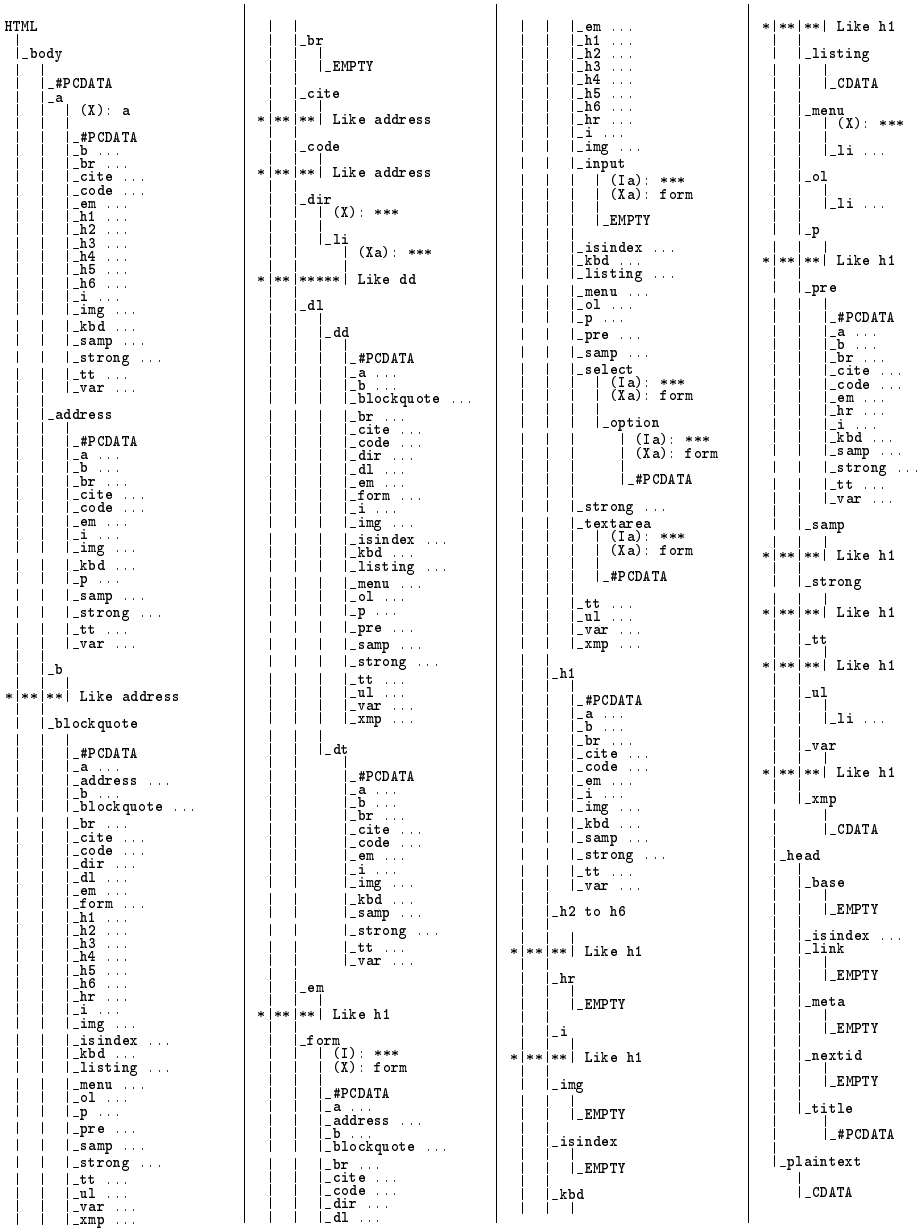


Figure 12: Output of the `dtdtree` program for the HTML2 DTD

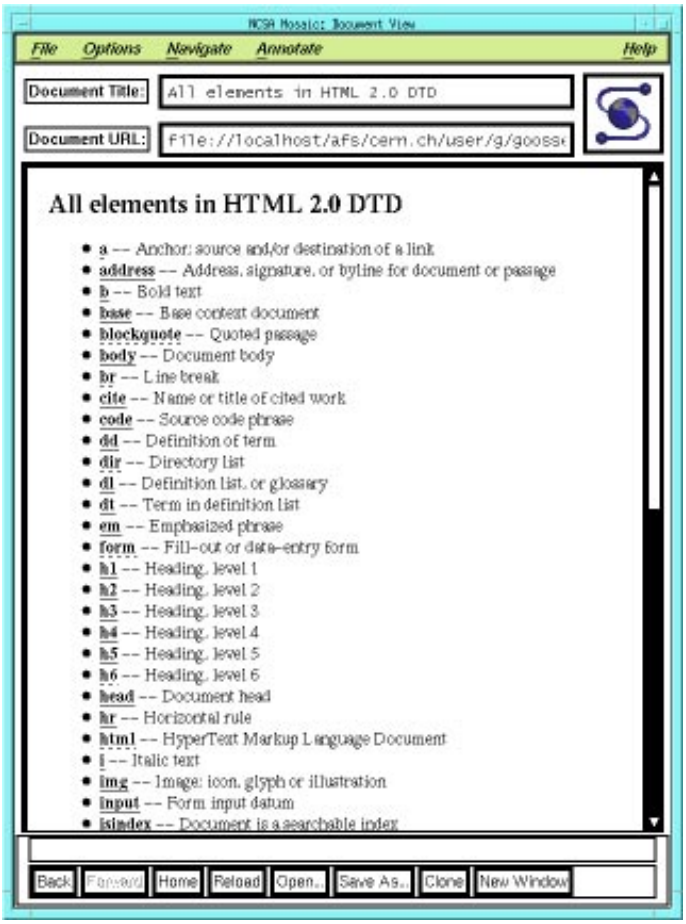


Figure 13: Hypertext description of the elements of a DTD (HTML2) as presented by the HTML browser `mosaic`

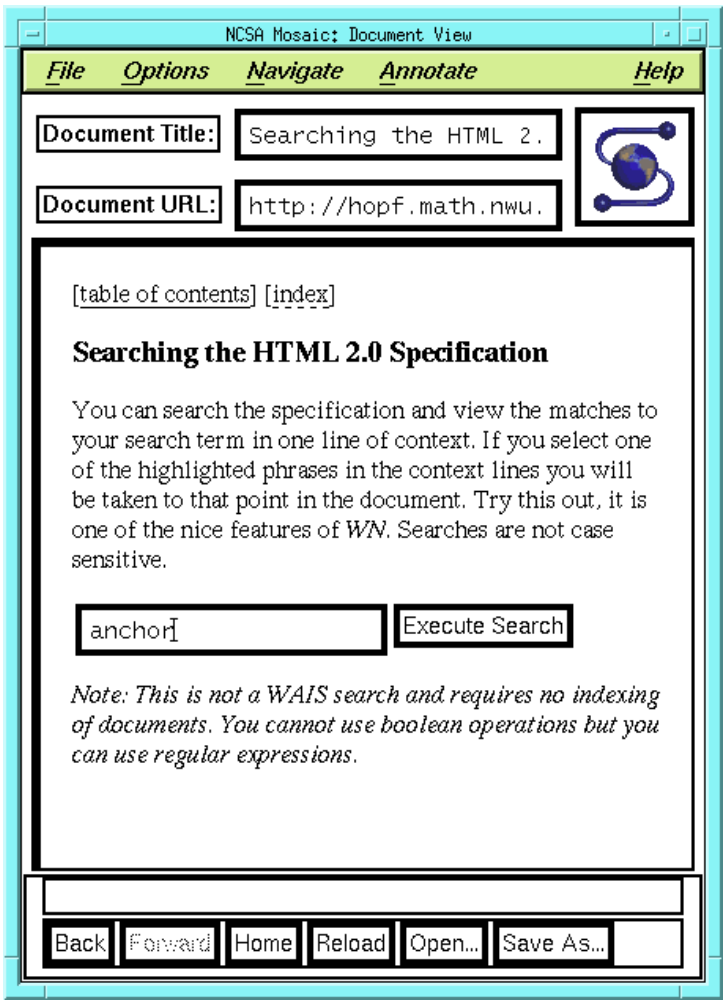


Figure 14: Searching the HTML2 DTD

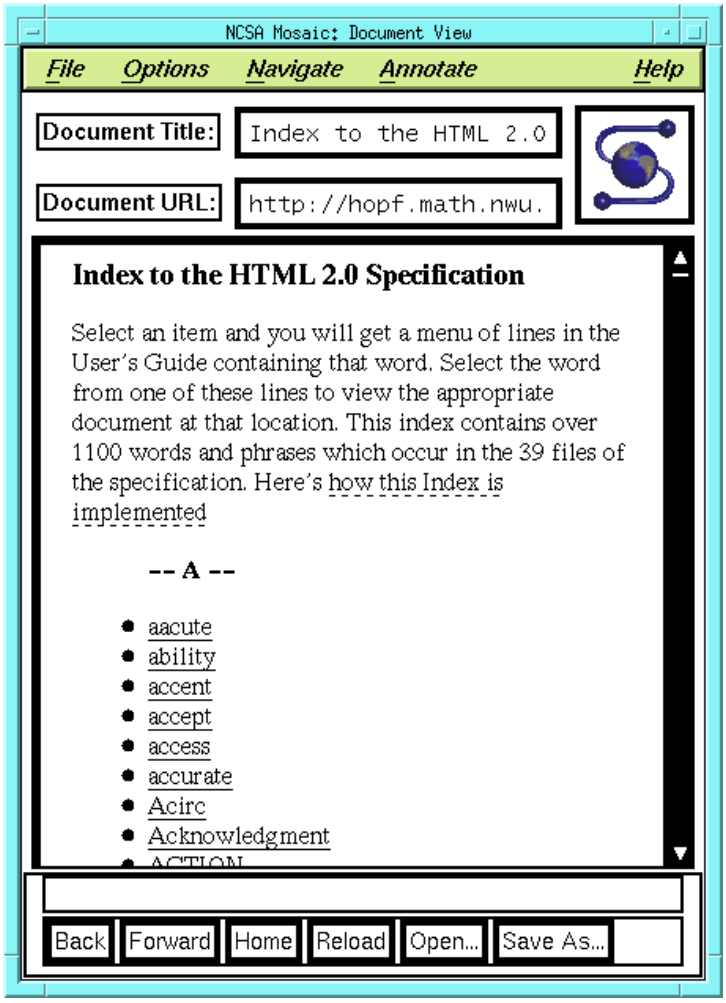


Figure 15: Index entries for the HTML2 DTD

`makemenu` makes a simple menu for HTML files, based on each file's `<TITLE>` tag; it can also make a simple table of contents based on the `<H1>`–`<H6>` heading tags;

`xtraclnk.pl` `perl` procedure to extract links and anchors from HTML files and to isolate text contained inside the `<A>` and `<TITLE>` elements;

`dehtml` removes all HTML markup from a document; is useful for spell checking;

`entify` replaces 8-bit Latin-1 input by the corresponding 7-bit-safe entity references;

The syntax to use these programs is typically:

```
awk -f htmlchek.awk [opts] infile > outfile

perl htmlchek.pl    [opts] infile > outfile
```

As an example we ran these scripts on the test files of section 6.2 with the results shown below, which are consistent with those obtained previously.

```
> perl dehtml.pl test1.html
  Document test HTML
  term 1data 1
  term 2data 2
  term 3
  term 4data 4data 4 bis


> awk -f htmlchek.awk test2.html
  Diagnostics for file "test2.html":
  <body> without preceding <head>...</head>
  Warning! at line 4 of file "test2.html"
  No <H1> in <body>...</body>
  Warning! at line 6 of file "test2.html"
  <HEAD> not used in document
  Warning! at END of file "test2.html"
  <TITLE> not used in document
  ERROR! at END of file "test2.html"
  Tag P occurred
  Tag HTML occurred
  Tag BODY occurred
  Tag !DOCTYPE occurred

> perl htmlchek.pl test3.html
  Diagnostics for file "test3.html":
  <LI> outside of list
```

```
ERROR! at line 8 of file "test3.html"
No <H1> in <body>...</body>
Warning! at line 9 of file "test3.html"
Tag !DOCTYPE occurred
Tag BODY occurred
Tag HEAD occurred
Tag HTML occurred
Tag LI occurred
Tag TITLE occurred
```

7 DTD transformations

The logical markup of SGML documents makes it possible to transform the markup associated to a DTD into that of another. When translating the markup one has to take into consideration the fact that between some elements a one-to-one mapping may not exist, but that a many-to-one, and one-to-many correspondence has to be considered. It should also be noted that the tools used for this purpose need to be sophisticated, since a normal grammar tool, such as `yacc`, is not suitable for parsing SGML documents.

7.1 SGMLS.PL

A translator skeleton, `sgmls.pl`, is included with the `nsgmls` distribution. This `perl` script reads the ESIS output of `nsgmls` and provides a set of routines that can be used for calling user-specified translation routines of each element.

7.2 SGMLS.PM and SGMLSPL

David Megginson (University of Ottawa, Canada, `dmeggins@aix1.uottawa.ca`) has developed a more object-oriented approach for the translations, also based on the ESIS output of `nsgmls` and calling event-routines for each element found in the input stream. This package includes a default configuration for translating documents marked up according to the `DocBook` DTD into HTML or LaTeX markup.

The `sp` parser provides an application level interface to SGML document handling. The core of `sp` uses C++ and provides a solid class library for parsing SGML documents. The parsing of an SGML document causes events and the user can write handlers to translate them in the appropriate way.

7.3 Conversion from DocBook to HTML3

The translation program generates events for each primitive in the source document and these events are handled by calling a corresponding routine. These routines then produce the corresponding HTML/LaTeX output. Thanks to its object-oriented flavour

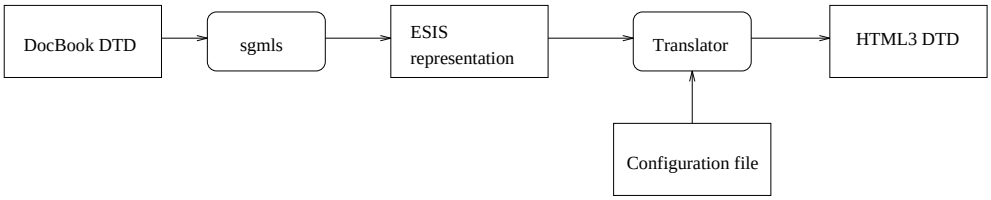


Figure 16: Schematic overview of the DocBook to HTML conversion process

the overall architecture provides solid ground for DTD translations . The following listing gives an idea of how the conversion is implemented. In the example below two elements are translated into LaTeX. When a tag is found that can be translated, the corresponding string is produced.

```
## Program listings appear in verbatim

sgml('<PROGRAMLISTING>',
          "\n\\begin{verbatim}\n");
sgml('</PROGRAMLISTING>',
           "\n\\end{verbatim}}\n");

## Class names appear in typewriter.

sgml('<CLASSNAME>', "{\\ttfamily ");
sgml('</CLASSNAME>', "}");
```

This example is extremely simple since the mappings are basically one-to-one. In the more general case, when a document element can be used inside different elements, the substitution is not just a string, but a procedure call, which allows, for instance, backtracking to cope with context-dependent conversion rules that take into account the current context. For instance, the code below shows how, when reaching the <TITLE> end tag, the title information is handled differently, according to whether it occurred inside an article header, section or table element.

```
sgml('<TITLE>',
    sub { push_output 'string'; });

sgml('</TITLE>', sub {
    my $element = shift;
    my $data = pop_output;
    if ($element->in(ARTHEADER)) {
        $title = $data;
    } elsif ($element->in(SECT1) ||
```

```
            $element->in(IMPORTANT)) {
            output "\n\n\\section{$data}\n";
            output "\\label{$id}\n" if $id;
            output "\n";
        } elsif ($element->in(TABLE)) {
            output "\\caption{$data}\n";
            output "\\label{$id}\n" if $id;
            output "\n";
        } else {
            die "No TITLE allowed in "
           . $element->parent->name . "\n";
        }
    });
```

A conversion example of an extract from the DocBook DTD manual is given in Appendix G. It shows part of the original DocBook document markup, how it is presented in the ESIS format, finally its translation in HTML3. Figure 16 shows the principle of the translation process.

7.4 Commercial solutions

Several companies provide commercial solutions for doing do the translations: Exoterica, AIS, EBT (Electronic Book Technologies) and Avalanche to mention few.

8 Other standards in the area of electronic documents

SGML is part of a vast project conceived by the International Standards Organization (ISO) to develop a model to describe the complete process of creating, exchanging, editing and viewing or printing of electronic documents. This model consists of several standards, some already adopted, others still under discussion (see [7] and [8]).

SGML (Standard Generalized Markup Language)

ISO 8879, the Standard described in this article is concerned with the creation and editing of documents. A complementary standard is ISO 9069 [10], SDIF, for "SGML Document Interchange Format". ISO/IEC 10744, the Hytime Standard, presents a formalism for the representation of hypermedia documents. The Hytime language ([6], [13]) allows the descriptions of situations that are time dependent (for example CD-I).

DSSSL (Document Style Semantics and Specification Language)

International Standard ISO 10179 [14], was adopted at the beginning of 1995. It presents a framework to express the concepts and actions necessary for transforming a structurally marked up document into its final physical form. Although this Standard is primarily

targeted at document handling, it can also define other layouts, such as those needed for use with databases.[19]

SPDL (Standard Page Description Language)

Draft International Standard ISO DIS 10180 [11] defines a formalism for the description of documents in their final, completely typeset, unrevisable form.[20] The structure of the language and its syntax strongly resemble the PostScript language, which is not surprising since PostScript has become the *de facto* standard page description language.

Fonts

To exchange documents one must also define a font standard. ISO 9541 [12] describes a method for naming and grouping glyphs or glyph collections independently of a particular font language (such as PostScript or Truetype).

Acknowledgments

We sincerely thank Nelson Beebe (Utah University, `beebe@math.utah.edu`) for several interesting e-mail discussions and for his detailed reading of the compuscript. His suggestions and hints have without doubt substantially improved the quality of the text. We also want to acknowledge the help of Steven Kennedy (CERN) who proofread the article.

References

[1] Association of American Publishers, Electronic Manuscript Series. *Author's Guide to Electronic Manuscript Preparation and Markup (Version 2)*. Association of American Publishers, EPSIG, Dublin, OH, USA, 1989.

[2] Association of American Publishers, Electronic Manuscript Series. *Markup of mathematical formulas (Version 2)*. Association of American Publishers, EPSIG, Dublin, OH, USA, 1989.

[3] Association of American Publishers, Electronic Manuscript Series. *Markup of tabular material (Version 2)*. Association of American Publishers, EPSIG, Dublin, OH, USA, 1989.

[4] Association of American Publishers, Electronic Manuscript Series. *Reference Manual on Electronic Manuscript Preparation and Markup (Version 2)*. Association of American Publishers, EPSIG, Dublin, OH, USA, 1989.

[5] C.F. Goldfarb. *The SGML Handbook*. Oxford University Press, 1990.

---

19. More on DSSSL by James Clark is available at the URL `http://www.jclark.com/dsssl/`.
20. More on SPDL can be found at the URL `http://www.st.rim.or.jp/~uda/spdl/spdl.html`.

[6] C.F. Goldfarb. Hytime: A standard for structured hypermedia interchange. *IEEE Computer*, pages 81–84, August 1991.
[7] M. Goossens and E. van Herwijnen. Introduction sgml, dsssl et spdl. *Cahiers GUTenberg*, 12:37–56, December 1991.
[8] M. Goossens and E. van Herwijnen. Scientific text processing. *Journal of Modern Physics C*, 3(3):479–546, June 1992.
[9] International Organization for Standardization. *Code for the presentation of names of languages*. ISO 639:1988 (E/F), ISO Geneva, 1988.
[10] International Organization for Standardization. *Information processing – SGML support facilities – SGML Document Interchange Format (SDIF)*. ISO 9069:1988, ISO Geneva, 1988.
[11] International Organization for Standardization. *Information Technology – Text Communication – Standard Page Description Language (SPDL)*. ISO/IEC DIS 10180, ISO Geneva, 1991.
[12] International Organization for Standardization. *Information Technology – Font information interchange (three parts)*. ISO/IEC 9541-1,2,3, ISO Geneva, 1991 and 1993.
[13] International Organization for Standardization. *Information Technology – Hypermedia/Time-based Structuring Language (Hytime)*. ISO/IEC 10744:1992, ISO Geneva, 1992.
[14] International Organization for Standardization. *Information processing – Text and office systems – Document Style Semantics and Specification Language (DSSSL)*. ISO/IEC DIS 10179.2, ISO Geneva, 1994.
[15] International Organization or Standardization. *Information processing – Text and office systems – Standard Generalized Markup Language (SGML)*. ISO 8879:1986(E), ISO Geneva, 1986.
[16] J. Karney. SGML and tag masters. *PC Magazine*, 14(3):144–162, 1995.
[17] J. Karney. SGML: It's still à la carte. *PC Magazine*, 14(3):168–171, 1995.
[18] P. Ores. Hypertext publishing – edit trial. *PC Magazine*, 14(3):132–143, 1995.
[19] Eric van Herwijnen. *Practical SGML (Second Edition)*. Wolters-Kluwer Academic Publishers, Boston, 1994.
[20] Dominique Vignaud. Éditions du Cercle de la Librairie, Paris, 1990.

Appendix A: The DTD of the HTML2 language

```
1   <!--    html.dtd
2
3           Document Type Definition for the HyperText Markup Language
4                   (HTML DTD)
5
6           $Id: html.dtd,v 1.25 1995/03/29 18:53:13 connolly Exp $
7
8           Author: Daniel W. Connolly <connolly@w3.org>
9           See Also: html.decl, html-0.dtd, html-1.dtd
```

```
10            http://info.cern.ch/hypertext/WWW/MarkUp/MarkUp.html
11    -->
12
13    <!ENTITY % HTML.Version
14            "-//IETF//DTD HTML 2.0//EN"
15
16            -- Typical usage:
17
18                <!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
19                <html>
20                ...
21                </html>
22            --
23            >
24
25
26    <!--============= Feature Test Entities ========================-->
27
28    <!ENTITY % HTML.Recommended "IGNORE"
29            -- Certain features of the language are necessary for
30               compatibility with widespread usage, but they may
31               compromise the structural integrity of a document.
32               This feature test entity enables a more prescriptive
33               document type definition that eliminates
34               those features.
35            -->
36
37    <![ %HTML.Recommended [
38            <!ENTITY % HTML.Deprecated "IGNORE">
39    ]]>
40
41    <!ENTITY % HTML.Deprecated "INCLUDE"
42            -- Certain features of the language are necessary for
43               compatibility with earlier versions of the specification,
44               but they tend to be used an implemented inconsistently,
45               and their use is deprecated. This feature test entity
46               enables a document type definition that eliminates
47               these features.
48            -->
49
50    <!ENTITY % HTML.Highlighting "INCLUDE"
51            -- Use this feature test entity to validate that a
52               document uses no highlighting tags, which may be
53               ignored on minimal implementations.
54            -->
55
56    <!ENTITY % HTML.Forms "INCLUDE"
57            -- Use this feature test entity to validate that a document
58               contains no forms, which may not be supported in minimal
59               implementations
60            -->
61
62    <!--============== Imported Names ==============================-->
63
64    <!ENTITY % Content-Type "CDATA"
65            -- meaning an internet media type
66               (aka MIME content type, as per RFC1521)
67            -->
68
69    <!ENTITY % HTTP-Method "GET | POST"
70            -- as per HTTP specification, in progress
71            -->
72
73    <!ENTITY % URI "CDATA"
```

```
74          -- The term URI means a CDATA attribute
75             whose value is a Uniform Resource Identifier,
76             as defined by
77          "Universal Resource Identifiers" by Tim Berners-Lee
78          aka RFC 1630
79
80          Note that CDATA attributes are limited by the LITLEN
81          capacity (1024 in the current version of html.decl),
82          so that URIs in HTML have a bounded length.
83
84          -->
85
86
87  <!--========== DTD "Macros" =====================-->
88
89  <!ENTITY % heading "H1|H2|H3|H4|H5|H6">
90
91  <!ENTITY % list " UL | OL | DIR | MENU " >
92
93
94  <!--======= Character mnemonic entities ==================-->
95
96
97  <!ENTITY % ISOlat1 PUBLIC
98    "-//IETF//ENTITIES Added Latin 1 for HTML//EN" "iso-lat1.gml">
99
100 %ISOlat1;
101
102 <!ENTITY amp CDATA "&#38;"     -- ampersand          -->
103 <!ENTITY gt CDATA "&#62;"      -- greater than       -->
104 <!ENTITY lt CDATA "&#60;"      -- less than          -->
105 <!ENTITY quot CDATA "&#34;"    -- double quote       -->
106
107
108 <!--========== SGML Document Access (SDA) Parameter Entities =====-->
109
110 <!-- HTML 2.0 contains SGML Document Access (SDA) fixed attributes
111 in support of easy transformation to the International Committee
112 for Accessible Document Design (ICADD) DTD
113          "-//EC-USA-CDA/ICADD//DTD ICADD22//EN".
114 ICADD applications are designed to support usable access to
115 structured information by print-impaired individuals through
116 Braille, large print and voice synthesis.  For more information on
117 SDA & ICADD:
118         - ISO 12083:1993, Annex A.8, Facilities for Braille,
119           large print and computer voice
120         - ICADD ListServ
121           <ICADD%ASUACAD.BITNET@ARIZVM1.ccit.arizona.edu>
122         - Usenet news group bit.listserv.easi
123         - Recording for the Blind, +1 800 221 4792
124 -->
125
126 <!ENTITY % SDAFORM  "SDAFORM  CDATA  #FIXED"
127           -- one to one mapping        -->
128 <!ENTITY % SDARULE  "SDARULE  CDATA  #FIXED"
129           -- context-sensitive mapping -->
130 <!ENTITY % SDAPREF  "SDAPREF  CDATA  #FIXED"
131           -- generated text prefix     -->
132 <!ENTITY % SDASUFF  "SDASUFF  CDATA  #FIXED"
133           -- generated text suffix     -->
134 <!ENTITY % SDASUSP  "SDASUSP  NAME   #FIXED"
135           -- suspend transform process -->
136
137
```

```
<!--========== Text Markup ====================-->

<![ %HTML.Highlighting [

<!ENTITY % font " TT | B | I ">

<!ENTITY % phrase "EM | STRONG | CODE | SAMP | KBD | VAR | CITE ">

<!ENTITY % text "#PCDATA | A | IMG | BR | %phrase | %font">

<!ELEMENT (%font;|%phrase) - - (%text)*>
<!ATTLIST ( TT | CODE | SAMP | KBD | VAR )
        %SDAFORM; "Lit"
        >
<!ATTLIST ( B | STRONG )
        %SDAFORM; "B"
        >
<!ATTLIST ( I | EM | CITE )
        %SDAFORM; "It"
        >

<!-- <TT>       Typewriter text                      -->
<!-- <B>        Bold text                            -->
<!-- <I>        Italic text                          -->

<!-- <EM>       Emphasized phrase                    -->
<!-- <STRONG>   Strong emphais                       -->
<!-- <CODE>     Source code phrase                   -->
<!-- <SAMP>     Sample text or characters            -->
<!-- <KBD>      Keyboard phrase, e.g. user input     -->
<!-- <VAR>      Variable phrase or substituable      -->
<!-- <CITE>     Name or title of cited work          -->

<!ENTITY % pre.content "#PCDATA | A | HR | BR | %font | %phrase">

]]>

<!ENTITY % text "#PCDATA | A | IMG | BR">

<!ELEMENT BR    - O EMPTY>
<!ATTLIST BR
        %SDAPREF; "&#RE;"
        >

<!-- <BR>       Line break      -->


<!--========== Link Markup ======================-->

<![ %HTML.Recommended [
        <!ENTITY % linkName "ID">
]]>

<!ENTITY % linkName "CDATA">

<!ENTITY % linkType "NAME"
        -- a list of these will be specified at a later date -->

<!ENTITY % linkExtraAttributes
        "REL %linkType #IMPLIED
        REV %linkType #IMPLIED
        URN CDATA #IMPLIED
        TITLE CDATA #IMPLIED
        METHODS NAMES #IMPLIED
```

```
202         ">
203
204 <![ %HTML.Recommended [
205         <!ENTITY % A.content   "(%text)*"
206         -- <H1><a name="xxx">Heading</a></H1>
207                 is preferred to
208            <a name="xxx"><H1>Heading</H1></a>
209         -->
210 ]]>
211
212 <!ENTITY % A.content   "(%heading|%text)*">
213
214 <!ELEMENT A     - - %A.content -(A)>
215 <!ATTLIST A
216         HREF %URI #IMPLIED
217         NAME %linkName #IMPLIED
218         %linkExtraAttributes;
219         %SDAPREF; "<Anchor: #AttList>"
220         >
221 <!-- <A               Anchor; source/destination of link      -->
222 <!-- <A NAME="...">   Name of this anchor                     -->
223 <!-- <A HREF="...">   Address of link destination            -->
224 <!-- <A URN="...">    Permanent address of destination        -->
225 <!-- <A REL=...>      Relationship to destination             -->
226 <!-- <A REV=...>      Relationship of destination to this     -->
227 <!-- <A TITLE="...">  Title of destination (advisory)        -->
228 <!-- <A METHODS="..."> Operations on destination (advisory)  -->
229
230
231 <!--========== Images ==========================-->
232
233 <!ELEMENT IMG    - O EMPTY>
234 <!ATTLIST IMG
235         SRC %URI;  #REQUIRED
236         ALT CDATA #IMPLIED
237         ALIGN (top|middle|bottom) #IMPLIED
238         ISMAP (ISMAP) #IMPLIED
239         %SDAPREF; "<Fig><?SDATrans Img: #AttList>#AttVal(Alt)</Fig>"
240         >
241
242 <!-- <IMG>            Image; icon, glyph or illustration      -->
243 <!-- <IMG SRC="...">  Address of image object                -->
244 <!-- <IMG ALT="...">  Textual alternative                     -->
245 <!-- <IMG ALIGN=...>  Position relative to text               -->
246 <!-- <IMG ISMAP>      Each pixel can be a link                -->
247
248 <!--========== Paragraphs=======================-->
249
250 <!ELEMENT P     - O (%text)*>
251 <!ATTLIST P
252         %SDAFORM; "Para"
253         >
254
255 <!-- <P>        Paragraph       -->
256
257
258 <!--========== Headings, Titles, Sections ===============-->
259
260 <!ELEMENT HR    - O EMPTY>
261 <!ATTLIST HR
262         %SDAPREF; "&#RE;&#RE;"
263         >
264
265 <!-- <HR>       Horizontal rule -->
```

```
266
267 <!ELEMENT ( %heading )  - -  (%text;)*>
268 <!ATTLIST H1
269         %SDAFORM; "H1"
270         >
271 <!ATTLIST H2
272         %SDAFORM; "H2"
273         >
274 <!ATTLIST H3
275         %SDAFORM; "H3"
276         >
277 <!ATTLIST H4
278         %SDAFORM; "H4"
279         >
280 <!ATTLIST H5
281         %SDAFORM; "H5"
282         >
283 <!ATTLIST H6
284         %SDAFORM; "H6"
285         >
286
287 <!-- <H1>       Heading, level 1 -->
288 <!-- <H2>       Heading, level 2 -->
289 <!-- <H3>       Heading, level 3 -->
290 <!-- <H4>       Heading, level 4 -->
291 <!-- <H5>       Heading, level 5 -->
292 <!-- <H6>       Heading, level 6 -->
293
294
295 <!--========== Text Flows ======================-->
296
297 <![ %HTML.Forms [
298         <!ENTITY % block.forms "BLOCKQUOTE | FORM | ISINDEX">
299 ]]>
300
301 <!ENTITY % block.forms "BLOCKQUOTE">
302
303 <![ %HTML.Deprecated [
304         <!ENTITY % preformatted "PRE | XMP | LISTING">
305 ]]>
306
307 <!ENTITY % preformatted "PRE">
308
309 <!ENTITY % block "P | %list | DL
310         | %preformatted
311         | %block.forms">
312
313 <!ENTITY % flow "(%text|%block)*">
314
315 <!ENTITY % pre.content "#PCDATA | A | HR | BR">
316 <!ELEMENT PRE - - (%pre.content)*>
317 <!ATTLIST PRE
318         WIDTH NUMBER #implied
319         %SDAFORM; "Lit"
320         >
321
322 <!-- <PRE>              Preformatted text              -->
323 <!-- <PRE WIDTH=...>    Maximum characters per line    -->
324
325 <![ %HTML.Deprecated [
326
327 <!ENTITY % literal "CDATA"
328         -- historical, non-conforming parsing mode where
329            the only markup signal is the end tag
```

```
330             in full
331         -->
332
333 <!ELEMENT (XMP|LISTING) - -  %literal>
334 <!ATTLIST XMP
335         %SDAFORM; "Lit"
336         %SDAPREF; "Example:&#RE;"
337         >
338 <!ATTLIST LISTING
339         %SDAFORM; "Lit"
340         %SDAPREF; "Listing:&#RE;"
341         >
342
343 <!-- <XMP>              Example section         -->
344 <!-- <LISTING>          Computer listing        -->
345
346 <!ELEMENT PLAINTEXT - O %literal>
347 <!-- <PLAINTEXT>        Plain text passage      -->
348
349 <!ATTLIST PLAINTEXT
350         %SDAFORM; "Lit"
351         >
352 ]]>
353
354
355 <!--========== Lists ==================-->
356
357 <!ELEMENT DL    - -  (DT | DD)+>
358 <!ATTLIST DL
359         COMPACT (COMPACT) #IMPLIED
360         %SDAFORM; "List"
361         %SDAPREF; "Definition List:"
362         >
363
364 <!ELEMENT DT    - O (%text)*>
365 <!ATTLIST DT
366         %SDAFORM; "Term"
367         >
368
369 <!ELEMENT DD    - O %flow>
370 <!ATTLIST DD
371         %SDAFORM; "LItem"
372         >
373
374 <!-- <DL>               Definition list, or glossary    -->
375 <!-- <DL COMPACT>       Compact style list              -->
376 <!-- <DT>               Term in definition list         -->
377 <!-- <DD>               Definition of term              -->
378
379 <!ELEMENT (OL|UL) - -  (LI)+>
380 <!ATTLIST OL
381         COMPACT (COMPACT) #IMPLIED
382         %SDAFORM; "List"
383         >
384 <!ATTLIST UL
385         COMPACT (COMPACT) #IMPLIED
386         %SDAFORM; "List"
387         >
388 <!-- <UL>               Unordered list                  -->
389 <!-- <UL COMPACT>       Compact list style              -->
390 <!-- <OL>               Ordered, or numbered list       -->
391 <!-- <OL COMPACT>       Compact list style              -->
392
393
```

```
<!ELEMENT (DIR|MENU) - -  (LI)+ -(%block)>
<!ATTLIST DIR
        COMPACT (COMPACT) #IMPLIED
        %SDAFORM; "List"
        %SDAPREF; "<LHead>Directory</LHead>"
        >
<!ATTLIST MENU
        COMPACT (COMPACT) #IMPLIED
        %SDAFORM; "List"
        %SDAPREF; "<LHead>Menu</LHead>"
        >

<!-- <DIR>              Directory list                 -->
<!-- <DIR COMPACT>      Compact list style             -->
<!-- <MENU>             Menu list                      -->
<!-- <MENU COMPACT>     Compact list style             -->

<!ELEMENT LI    - O %flow>
<!ATTLIST LI
        %SDAFORM; "LItem"
        >

<!-- <LI>               List item                      -->

<!--========== Document Body ==================-->

<![ %HTML.Recommended [
        <!ENTITY % body.content "(%heading|%block|HR|ADDRESS|IMG)*"
        -- <h1>Heading</h1>
           <p>Text ...
                is preferred to
           <h1>Heading</h1>
           Text ...
        -->
]]>

<!ENTITY % body.content "(%heading | %text | %block |
                                HR | ADDRESS)*">

<!ELEMENT BODY O O  %body.content>

<!-- <BODY>     Document body   -->

<!ELEMENT BLOCKQUOTE - - %body.content>
<!ATTLIST BLOCKQUOTE
        %SDAFORM; "BQ"
        >

<!-- <BLOCKQUOTE>       Quoted passage  -->

<!ELEMENT ADDRESS - - (%text|P)*>
<!ATTLIST  ADDRESS
        %SDAFORM; "Lit"
        %SDAPREF; "Address:&#RE;"
        >

<!-- <ADDRESS>  Address, signature, or byline   -->


<!--======= Forms ====================-->

<![ %HTML.Forms [

<!ELEMENT FORM - - %body.content -(FORM) +(INPUT|SELECT|TEXTAREA)>
```

```
<!ATTLIST FORM
        ACTION %URI #IMPLIED
        METHOD (%HTTP-Method) GET
        ENCTYPE %Content-Type; "application/x-www-form-urlencoded"
        %SDAPREF; "<Para>Form:</Para>"
        %SDASUFF; "<Para>Form End.</Para>"
        >

<!-- <FORM>                    Fill-out or data-entry form     -->
<!-- <FORM ACTION="...">       Address for completed form      -->
<!-- <FORM METHOD=...>         Method of submitting form       -->
<!-- <FORM ENCTYPE="...">      Representation of form data     -->

<!ENTITY % InputType "(TEXT | PASSWORD | CHECKBOX |
                       RADIO | SUBMIT | RESET |
                       IMAGE | HIDDEN )">
<!ELEMENT INPUT - O EMPTY>
<!ATTLIST INPUT
        TYPE %InputType TEXT
        NAME CDATA #IMPLIED
        VALUE CDATA #IMPLIED
        SRC %URI #IMPLIED
        CHECKED (CHECKED) #IMPLIED
        SIZE CDATA #IMPLIED
        MAXLENGTH NUMBER #IMPLIED
        ALIGN (top|middle|bottom) #IMPLIED
        %SDAPREF; "Input: "
        >

<!-- <INPUT>                   Form input datum                -->
<!-- <INPUT TYPE=...>          Type of input interaction       -->
<!-- <INPUT NAME=...>          Name of form datum              -->
<!-- <INPUT VALUE="...">       Default/initial/selected value  -->
<!-- <INPUT SRC="...">         Address of image                -->
<!-- <INPUT CHECKED>           Initial state is "on"           -->
<!-- <INPUT SIZE=...>          Field size hint                 -->
<!-- <INPUT MAXLENGTH=...>     Data length maximum             -->
<!-- <INPUT ALIGN=...>         Image alignment                 -->

<!ELEMENT SELECT - - (OPTION+) -(INPUT|SELECT|TEXTAREA)>
<!ATTLIST SELECT
        NAME CDATA #REQUIRED
        SIZE NUMBER #IMPLIED
        MULTIPLE (MULTIPLE) #IMPLIED
        %SDAFORM; "List"
        %SDAPREF;
        "<LHead>Select #AttVal(Multiple)</LHead>"
        >

<!-- <SELECT>                  Selection of option(s)          -->
<!-- <SELECT NAME=...>         Name of form datum              -->
<!-- <SELECT SIZE=...>         Options displayed at a time     -->
<!-- <SELECT MULTIPLE>         Multiple selections allowed     -->

<!ELEMENT OPTION - O (#PCDATA)*>
<!ATTLIST OPTION
        SELECTED (SELECTED) #IMPLIED
        VALUE CDATA #IMPLIED
        %SDAFORM; "LItem"
        %SDAPREF;
        "Option: #AttVal(Value) #AttVal(Selected)"
        >

<!-- <OPTION>                  A selection option              -->
```

```
<!-- <OPTION SELECTED>          Initial state                    -->
<!-- <OPTION VALUE="...">       Form datum value for this option-->

<!ELEMENT TEXTAREA - - (#PCDATA)* -(INPUT|SELECT|TEXTAREA)>
<!ATTLIST TEXTAREA
        NAME CDATA #REQUIRED
        ROWS NUMBER #REQUIRED
        COLS NUMBER #REQUIRED
        %SDAFORM; "Para"
        %SDAPREF; "Input Text -- #AttVal(Name): "
        >

<!-- <TEXTAREA>                 An area for text input           -->
<!-- <TEXTAREA NAME=...>        Name of form datum               -->
<!-- <TEXTAREA ROWS=...>        Height of area                   -->
<!-- <TEXTAREA COLS=...>        Width of area                    -->

]]>


<!--======= Document Head ======================-->

<![ %HTML.Recommended [
        <!ENTITY % head.extra "META* & LINK*">
]]>

<!ENTITY % head.extra "NEXTID? & META* & LINK*">

<!ENTITY % head.content "TITLE & ISINDEX? & BASE? &
                         (%head.extra)">

<!ELEMENT HEAD O O  (%head.content)>

<!-- <HEAD>     Document head   -->

<!ELEMENT TITLE - -  (#PCDATA)*>
<!ATTLIST TITLE
        %SDAFORM; "Ti"    >

<!-- <TITLE>    Title of document -->

<!ELEMENT LINK - O EMPTY>
<!ATTLIST LINK
        HREF %URI #REQUIRED
        %linkExtraAttributes;
        %SDAPREF; "Linked to : #AttVal (TITLE) (URN) (HREF)>"    >

<!-- <LINK>             Link from this document                  -->
<!-- <LINK HREF="...">  Address of link destination             -->
<!-- <LINK URN="...">   Lasting name of destination             -->
<!-- <LINK REL=...>     Relationship to destination              -->
<!-- <LINK REV=...>     Relationship of destination to this      -->
<!-- <LINK TITLE="..."> Title of destination (advisory)         -->
<!-- <LINK METHODS="..."> Operations allowed (advisory)         -->

<!ELEMENT ISINDEX - O EMPTY>
<!ATTLIST ISINDEX
        %SDAPREF;
   "<Para>[Document is indexed/searchable.]</Para>">

<!-- <ISINDEX>          Document is a searchable index           -->

<!ELEMENT BASE - O EMPTY>
<!ATTLIST BASE
```

```
586         HREF %URI; #REQUIRED     >
587
588 <!-- <BASE>             Base context document                  -->
589 <!-- <BASE HREF="...">  Address for this document             -->
590
591 <!ELEMENT NEXTID - O EMPTY>
592 <!ATTLIST NEXTID
593         N %linkName #REQUIRED     >
594
595 <!-- <NEXTID>           Next ID to use for link name           -->
596 <!-- <NEXTID N=...>     Next ID to use for link name           -->
597
598 <!ELEMENT META - O EMPTY>
599 <!ATTLIST META
600         HTTP-EQUIV  NAME    #IMPLIED
601         NAME        NAME    #IMPLIED
602         CONTENT     CDATA   #REQUIRED    >
603
604 <!-- <META>                     Generic Metainformation        -->
605 <!-- <META HTTP-EQUIV=...>      HTTP response header name      -->
606 <!-- <META NAME=...>            Metainformation name           -->
607 <!-- <META CONTENT="...">       Associated information         -->
608
609 <!--======= Document Structure ================-->
610
611 <![ %HTML.Deprecated [
612         <!ENTITY % html.content "HEAD, BODY, PLAINTEXT?">
613 ]]>
614 <!ENTITY % html.content "HEAD, BODY">
615
616 <!ELEMENT HTML O O  (%html.content)>
617 <!ENTITY % version.attr "VERSION CDATA #FIXED '%HTML.Version;'">
618
619 <!ATTLIST HTML
620         %version.attr;
621         %SDAFORM; "Book"
622         >
623
624 <!-- <HTML>                     HTML Document   -->
```

Appendix B: The HTML2 SGML declaration

```
1   <!SGML  "ISO 8879:1986"
2   --
3           SGML Declaration for HyperText Markup Language (HTML).
4
5   --
6
7   CHARSET
8            BASESET  "ISO 646:1983//CHARSET
9                      International Reference Version
10                     (IRV)//ESC 2/5 4/0"
11           DESCSET  0   9   UNUSED
12                    9   2   9
13                    11  2   UNUSED
14                    13  1   13
15                    14  18  UNUSED
16                    32  95  32
17                    127 1   UNUSED
18       BASESET   "ISO Registration Number 100//CHARSET
19                  ECMA-94 Right Part of
20                  Latin Alphabet Nr. 1//ESC 2/13 4/1"
```

```
21
22          DESCSET  128  32   UNUSED
23                   160  96    32
24
25  CAPACITY        SGMLREF
26                  TOTALCAP        150000
27                  GRPCAP          150000
28
29  SCOPE    DOCUMENT
30  SYNTAX
31          SHUNCHAR CONTROLS 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
32                   17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 127
33          BASESET  "ISO 646:1983//CHARSET
34                    International Reference Version
35                    (IRV)//ESC 2/5 4/0"
36          DESCSET  0 128 0
37          FUNCTION
38                   RE          13
39                   RS          10
40                   SPACE       32
41                   TAB SEPCHAR  9
42
43
44          NAMING   LCNMSTRT ""
45                   UCNMSTRT ""
46                   LCNMCHAR ".-"
47                   UCNMCHAR ".-"
48                   NAMECASE GENERAL YES
49                            ENTITY  NO
50          DELIM    GENERAL  SGMLREF
51                   SHORTREF SGMLREF
52          NAMES    SGMLREF
53          QUANTITY SGMLREF
54                   ATTSPLEN 2100
55                   LITLEN   1024
56                   NAMELEN  72    -- somewhat arbitrary; taken from
57                                  internet line length conventions --
58                   PILEN    1024
59                   TAGLEN   2100
60                   GRPGTCNT 150
61                   GRPCNT   64
62
63  FEATURES
64    MINIMIZE
65      DATATAG  NO
66      OMITTAG  YES
67      RANK     NO
68      SHORTTAG YES
69    LINK
70      SIMPLE   NO
71      IMPLICIT NO
72      EXPLICIT NO
73    OTHER
74      CONCUR   NO
75      SUBDOC   NO
76      FORMAL   YES
77    APPINFO    "SDA"  -- conforming SGML Document Access application
78                      --
79  >
80  <!--
81          $Id: html.decl,v 1.14 1995/02/10 22:20:05 connolly Exp $
82
83          Author: Daniel W. Connolly <connolly@hal.com>
84
```

```
85          See also: http://www.hal.com/%7Econnolly/html-spec
86            http://info.cern.ch/hypertext/WWW/MarkUp/MarkUp.html
87   -->
```

Appendix C: The SGML open HTML catalog file

SGML Open is an industry consortium dedicated to encouraging the adoption of SGML as a standard for document and data interchange. It proposes a standard way for mapping entity and other external references in a DTD to file names via a "catalog" file. Below is an example of such a catalog file for HTML.

Appendix D: The ISO-Latin1 entity set

To have an idea of how character entity sets are defined in practice, below is shown the file corresponding to Latin1 (standard ISO/IEC 8859-1), available as SGML public entity set `ISOlat1` with ISO 8879.

```
1   <!-- (C) International Organization for Standardization 1986
2        Permission to copy in any form is granted for use with
3        conforming SGML systems and applications as defined in
4        ISO 8879, provided this notice is included in all copies.
5   -->
6   <!-- Character entity set. Typical invocation:
7        <!ENTITY % ISOlat1 PUBLIC
8          "ISO 8879-1986//ENTITIES Added Latin 1//EN">
9        %ISOlat1;
10  -->
11  <!ENTITY aacute SDATA "[aacute]"--=small a, acute accent-->
12  <!ENTITY Aacute SDATA "[Aacute]"--=capital A, acute accent-->
13  <!ENTITY acirc  SDATA "[acirc ]"--=small a, circumflex accent-->
14  <!ENTITY Acirc  SDATA "[Acirc ]"--=capital A, circumflex accent-->
15  <!ENTITY agrave SDATA "[agrave]"--=small a, grave accent-->
16  <!ENTITY Agrave SDATA "[Agrave]"--=capital A, grave accent-->
17  <!ENTITY aring  SDATA "[aring ]"--=small a, ring-->
18  <!ENTITY Aring  SDATA "[Aring ]"--=capital A, ring-->
19  <!ENTITY atilde SDATA "[atilde]"--=small a, tilde-->
20  <!ENTITY Atilde SDATA "[Atilde]"--=capital A, tilde-->
21  <!ENTITY auml   SDATA "[auml  ]"--=small a, dieresis or umlaut mark-->
22  <!ENTITY Auml   SDATA "[Auml  ]"--=capital A, dieresis or umlaut mark-->
23  <!ENTITY aelig  SDATA "[aelig ]"--=small ae diphthong (ligature)-->
24  <!ENTITY AElig  SDATA "[AElig ]"--=capital AE diphthong (ligature)-->
25  <!ENTITY ccedil SDATA "[ccedil]"--=small c, cedilla-->
26  <!ENTITY Ccedil SDATA "[Ccedil]"--=capital C, cedilla-->
27  <!ENTITY eth    SDATA "[eth   ]"--=small eth, Icelandic-->
28  <!ENTITY ETH    SDATA "[ETH   ]"--=capital Eth, Icelandic-->
29  <!ENTITY eacute SDATA "[eacute]"--=small e, acute accent-->
30  <!ENTITY Eacute SDATA "[Eacute]"--=capital E, acute accent-->
31  <!ENTITY ecirc  SDATA "[ecirc ]"--=small e, circumflex accent-->
32  <!ENTITY Ecirc  SDATA "[Ecirc ]"--=capital E, circumflex accent-->
33  <!ENTITY egrave SDATA "[egrave]"--=small e, grave accent-->
34  <!ENTITY Egrave SDATA "[Egrave]"--=capital E, grave accent-->
35  <!ENTITY euml   SDATA "[euml  ]"--=small e, dieresis or umlaut mark-->
36  <!ENTITY Euml   SDATA "[Euml  ]"--=capital E, dieresis or umlaut mark-->
37  <!ENTITY iacute SDATA "[iacute]"--=small i, acute accent-->
```

```
<!ENTITY Iacute SDATA "[Iacute]"--=capital I, acute accent-->
<!ENTITY icirc  SDATA "[icirc ]"--=small i, circumflex accent-->
<!ENTITY Icirc  SDATA "[Icirc ]"--=capital I, circumflex accent-->
<!ENTITY igrave SDATA "[igrave]"--=small i, grave accent-->
<!ENTITY Igrave SDATA "[Igrave]"--=capital I, grave accent-->
<!ENTITY iuml   SDATA "[iuml  ]"--=small i, dieresis or umlaut mark-->
<!ENTITY Iuml   SDATA "[Iuml  ]"--=capital I, dieresis or umlaut mark-->
<!ENTITY ntilde SDATA "[ntilde]"--=small n, tilde-->
<!ENTITY Ntilde SDATA "[Ntilde]"--=capital N, tilde-->
<!ENTITY oacute SDATA "[oacute]"--=small o, acute accent-->
<!ENTITY Oacute SDATA "[Oacute]"--=capital O, acute accent-->
<!ENTITY ocirc  SDATA "[ocirc ]"--=small o, circumflex accent-->
<!ENTITY Ocirc  SDATA "[Ocirc ]"--=capital O, circumflex accent-->
<!ENTITY ograve SDATA "[ograve]"--=small o, grave accent-->
<!ENTITY Ograve SDATA "[Ograve]"--=capital O, grave accent-->
<!ENTITY oslash SDATA "[oslash]"--=small o, slash-->
<!ENTITY Oslash SDATA "[Oslash]"--=capital O, slash-->
<!ENTITY otilde SDATA "[otilde]"--=small o, tilde-->
<!ENTITY Otilde SDATA "[Otilde]"--=capital O, tilde-->
<!ENTITY ouml   SDATA "[ouml  ]"--=small o, dieresis or umlaut mark-->
<!ENTITY Ouml   SDATA "[Ouml  ]"--=capital O, dieresis or umlaut mark-->
<!ENTITY szlig  SDATA "[szlig ]"--=small sharp s, German (sz ligature)-->
<!ENTITY thorn  SDATA "[thorn ]"--=small thorn, Icelandic-->
<!ENTITY THORN  SDATA "[THORN ]"--=capital THORN, Icelandic-->
<!ENTITY uacute SDATA "[uacute]"--=small u, acute accent-->
<!ENTITY Uacute SDATA "[Uacute]"--=capital U, acute accent-->
<!ENTITY ucirc  SDATA "[ucirc ]"--=small u, circumflex accent-->
<!ENTITY Ucirc  SDATA "[Ucirc ]"--=capital U, circumflex accent-->
<!ENTITY ugrave SDATA "[ugrave]"--=small u, grave accent-->
<!ENTITY Ugrave SDATA "[Ugrave]"--=capital U, grave accent-->
<!ENTITY uuml   SDATA "[uuml  ]"--=small u, dieresis or umlaut mark-->
<!ENTITY Uuml   SDATA "[Uuml  ]"--=capital U, dieresis or umlaut mark-->
<!ENTITY yacute SDATA "[yacute]"--=small y, acute accent-->
<!ENTITY Yacute SDATA "[Yacute]"--=capital Y, acute accent-->
<!ENTITY yuml   SDATA "[yuml  ]"--=small y, dieresis or umlaut mark-->

```

Appendix E: The HTML3 DTD – Tables and mathematics parts

This appendix shows those parts of the HTML3 DTD that relate to tables and mathematics.

```
<!--======================== Captions ========================================-->

<!ELEMENT CAPTION - - (%text;)+ -- table or figure caption -->
<!ATTLIST CAPTION
        %attrs;
        align (top|bottom|left|right) #IMPLIED
        >
<!--======================== Tables ==========================================-->

<!--
    Tables and figures can be aligned in several ways:

    bleedleft   flush left with the left (window) border
    left        flush left with the left text margin
    center      centered (text flow is disabled for this mode)
    right       flush right with the right text margin
    bleedright  flush right with the right (window) border
    justify     when applicable the table/figure should stretch
                to fill space between the text margins
```

```
20
21      Note: text will flow around the table or figure if the browser
22      judges there is enough room and the alignment is not centered
23      or justified. The table or figure may itself be part of the
24      text flow around some earlier figure. You can in this case use
25      the clear or needs attributes to move the new table or figure
26      down the page beyond the obstructing earlier figure. Similarly,
27      you can use the clear or needs attributes with other elements
28      such as headers and lists to move them further down the page.
29  -->
30
31  <!ENTITY % block.align
32          "align  (bleedleft|left|center|right|bleedright|justify) center">
33
34  <!--
35      The HTML 3.0 table model has been chosen for its simplicity
36      and the ease in writing filters from common DTP packages.
37
38      By default the table is automatically sized according to the
39      cell contents and the current window size. Specifying the columns
40      widths using the colspec attribute allows browsers to start
41      displaying the table without having to wait for last row.
42
43      The colspec attribute is a list of column widths and alignment
44      specifications. The columns are listed from left to right with
45      a capital letter followed by a number, e.g. COLSPEC="L20 C8 L40".
46      The letter is L for left, C for center, R for right alignment of
47      cell contents. J is for justification, when feasible, otherwise
48      this is treated in the same way as L for left alignment.
49      Column entries are delimited by one or more space characters.
50
51      The number specifies the width in em's, pixels or as a
52      fractional value of the table width, as according to the
53      associated units attribute. This approach is more compact
54      than used with most SGML table models and chosen to simplify
55      hand entry. The width attribute allows you to specify the
56      width of the table in pixels, em units or as a percentage
57      of the space between the current left and right margins.
58
59      To assist with rendering to speech, row and column headers
60      can be given short names using the AXIS attribute. The AXES
61      attribute is used to explicitly specify the row and column
62      names for use with each cell. Otherwise browsers can follow
63      up columns and left along rows (right for some languages)
64      to find the corresponding header cells.
65
66      Table content model: Braille limits the width of tables,
67      placing severe limits on column widths. User agents need
68      to render big cells by moving the content to a note placed
69      before the table. The cell is then rendered as a link to
70      the corresponding note.
71
72      To assist with formatting tables to paged media, authors
73      can differentiate leading and trailing rows that are to
74      be duplicated when splitting tables across page boundaries.
75      The recommended way is to subclass rows with the CLASS attribute
76      For example: <TR CLASS=Header>, <TR CLASS=Footer> are used for
77      header and footer rows. Paged browsers insert footer rows at
78      the bottom of the current page and header rows at the top of
79      the new page, followed by the remaining body rows.
80  -->
81
82  <!ELEMENT TABLE - - (CAPTION?, TR*) -- mixed headers and data -->
83  <!ATTLIST TABLE
```

```
84          %attrs;
85          %needs; -- for control of text flow --
86          border (border) #IMPLIED -- draw borders --
87          colspec CDATA   #IMPLIED -- column widths and alignment --
88          units  (em|pixels|relative) em -- units for column widths --
89          width NUMBER    #IMPLIED -- absolute or percentage width --
90          %block.align;  -- horizontal alignment --
91          nowrap (nowrap) #IMPLIED -- don't wrap words --
92          >
93
94  <!ENTITY % cell "TH | TD">
95  <!ENTITY % vertical.align "top|middle|bottom|baseline">
96
97  <!--
98      Browsers should tolerate an omission of the first <TR>
99      tag as it is implied by the context. Missing trailing
100     <TR>s implied by rowspans should be ignored.
101
102     The alignment attributes act as defaults for rows
103     overriding the colspec attribute and being in turn
104     overridden by alignment attributes on cell elements.
105     Use valign=baseline when you want to ensure that text
106     in different cells on the same row is aligned on the
107     same baseline regardless of fonts. It only applies
108     when the cells contain a single line of text.
109 -->
110
111 <!ELEMENT TR - O (%cell)* -- row container -->
112 <!ATTLIST TR
113         %attrs;
114         align  (left|center|right|justify) #IMPLIED
115         valign (%vertical.align) top -- vertical alignment --
116         nowrap (nowrap) #IMPLIED -- don't wrap words --
117         >
118
119 <!--
120     Note that table cells can include nested tables.
121     Missing cells are considered to be empty, while
122     missing rows should be ignored, i.e. if a cell
123     spans a row and there are no further TR elements
124     then the implied row should be ignored.
125 -->
126
127 <!ELEMENT (%cell) - O %body.content>
128 <!ATTLIST (%cell)
129         %attrs;
130         colspan NUMBER    1      -- columns spanned --
131         rowspan NUMBER    1      -- rows spanned --
132         align  (left|center|right|justify) #IMPLIED
133         valign (%vertical.align) top -- vertical alignment --
134         nowrap (nowrap) #IMPLIED -- don't wrap words --
135         axis CDATA #IMPLIED -- axis name, defaults to element content --
136         axes CDATA #IMPLIED -- comma separated list of axis names --
137         >
138
139 <!--================ Entities for math symbols ============================-->
140
141 <!-- ISO subset chosen for use with the widely available Adobe math font -->
142
143 <!ENTITY % HTMLmath PUBLIC
144   "-//IETF//ENTITIES Math and Greek for HTML//EN">
145 %HTMLmath;
146
147 <!--======================== Math  =======================================-->
```

```
148
149 <!-- Use     etc for greater control of spacing. -->
150
151 <!-- Subscripts and Superscripts
152
153   <SUB> and <SUP> are used for subscripts and superscripts.
154
155                                          i j
156       X <SUP>i</SUP>Y<SUP>j</SUP>  is   X  Y
157
158   i.e. the space following the X disambiguates the binding.
159   The align attribute can be used for horizontal alignment,
160   e.g. to explicitly place an index above an element:
161                                              i
162         X<sup align=center>i</sup>  produces  X
163
164 Short references are defined for superscripts, subscripts and boxes
165 to save typing when manually editing HTML math, e.g.
166
167       x^2^    is mapped to   x<sup>2</sup>
168       y_z_    is mapped to   y<sub>z</sub>
169       {a+b}   is mapped to   <box>a + b</box>
170
171 Note that these only apply within the MATH element and can't be
172 used in normal text!
173 -->
174 <!ENTITY REF1   STARTTAG   "SUP">
175 <!ENTITY REF2   ENDTAG     "SUP">
176 <!ENTITY REF3   STARTTAG   "SUB">
177 <!ENTITY REF4   ENDTAG     "SUB">
178 <!ENTITY REF5   STARTTAG   "BOX">
179 <!ENTITY REF6   ENDTAG     "BOX">
180
181 <!USEMAP MAP1   MATH>
182 <!USEMAP MAP2   SUP>
183 <!USEMAP MAP3   SUB>
184 <!USEMAP MAP4   BOX>
185
186 <!SHORTREF MAP1 "^" REF1
187                 "_" REF3
188                 "{" REF5 >
189
190 <!SHORTREF MAP2 "^" REF2
191                 "_" REF3
192                 "{" REF5 >
193
194 <!SHORTREF MAP3 "_" REF4
195                 "^" REF1
196                 "{" REF5 >
197
198 <!SHORTREF MAP4 "}" REF6
199                 "^" REF1
200                 "_" REF3
201                 "{" REF5 >
202
203 <!--
204  The inclusion of %math and exclusion of %notmath is used here
205  to alter the content model for the B, SUB and SUP elements,
206  to limit them to formulae rather than general text elements.
207 -->
208
209 <!ENTITY % mathvec "VEC|BAR|DOT|DDOT|HAT|TILDE" -- common accents -->
210 <!ENTITY % mathface "B|T|BT" -- control of font face -->
211 <!ENTITY % math "BOX|ABOVE|BELOW|%mathvec|ROOT|SQRT|ARRAY|SUB|SUP|%mathface">
```

```
<!ENTITY % formula "#PCDATA|%math">

<!ELEMENT MATH - - (#PCDATA)* -(%notmath) +(%math)>
<!ATTLIST MATH
        id      ID      #IMPLIED
        model   CDATA   #IMPLIED>

<!-- The BOX element acts as brackets. Delimiters are optional and
     stretch to match the height of the box. The OVER element is used
     when you want a line between numerator and denominator. This line
     is suppressed with the alternative ATOP element. CHOOSE acts like
     ATOP but adds enclosing round brackets as a convenience for binomial
     coefficients. Note the use of { and } as shorthand for <BOX> and
     </BOX> respectively:

                            1 + X
     {1 + X<OVER>Y}  is   -------
                            Y

                                  a + b
     {a + b<ATOP>c - d} is
                                  c - d

     The delimiters are represented using the LEFT and RIGHT
     elements as in:

     {[<LEFT>x + y<RIGHT>]}   is   [ x + y ]
     {(<LEFT>a<RIGHT>]}       is   (a]
     {||<LEFT>a<RIGHT>||}     is   || a ||

     Use &lbrace; and &rbrace; for "{" and "}" respectively as
     these symbols are used as shorthand for BOX, e.g.

     {&lbrace;<LEFT>a+b<RIGHT>&rbrace;}  is  {a+b}

     You can stretch definite integrals to match the integrand, e.g.

     {&int;<SUB>a</SUB><SUP>b</SUP><LEFT>{f(x)<over>1+x} dx}

            b
            /  f(x)
            | ----- dx
            / 1 + x
            a

     Note the complex content model for BOX is a work around
     for the absence of support for infix operators in SGML.

     You can get oversize delimiters with the SIZE attribute,
     for example <BOX SIZE=large>(<LEFT>...<RIGHT>)</BOX>

     Note that the names of common functions are recognized
     by the parser without the need to use "&" and ";" around
     them, e.g. int, sum, sin, cos, tan, ...
-->

<!ELEMENT BOX - - ((%formula)*, (LEFT, (%formula)*)?,
                   ((OVER|ATOP|CHOOSE), (%formula)*)?,
                   (RIGHT, (%formula)*)?)>
<!ATTLIST BOX
        size  (normal|medium|large|huge) normal -- oversize delims -->

<!ELEMENT (OVER|ATOP|CHOOSE|LEFT|RIGHT) - O EMPTY>

```

```
<!-- Horizontal line drawn ABOVE contents
     The symbol attribute allows authors to supply
     an entity name for an accent, arrow symbol etc.
     Generalisation of LaTeX's overline command.
 -->

<!ELEMENT ABOVE - - (%formula)+>
<!ATTLIST ABOVE symbol ENTITY #IMPLIED>

<!-- Horizontal line drawn BELOW contents
     The symbol attribute allows authors to
     supply an entity name for an arrow symbol etc.
     Generalisation of LaTeX's underline command.
 -->

<!ELEMENT BELOW - - (%formula)+>
<!ATTLIST BELOW symbol ENTITY #IMPLIED>

<!-- Convenience tags for common accents:
     vec, bar, dot, ddot, hat and tilde
-->

<!ELEMENT (%mathvec) - - (%formula)+>

<!--
  T and BT are used to designate terms which should
  be rendered in an upright font (& bold face for BT)
-->

<!ELEMENT (T|BT) - - (%formula)+>
<!ATTLIST (T|BT) class NAMES #IMPLIED>

<!-- Roots  e.g. <ROOT>3<OF>1+x</ROOT> -->

<!ELEMENT ROOT - - ((%formula)+, OF, (%formula)+)>
<!ELEMENT OF - O (%formula)* -- what the root applies to -->

<!ELEMENT SQRT - - (%formula)* -- square root convenience tag -->

<!-- LaTeX like arrays. The COLDEF attribute specifies
     a single capital letter for each column determining
     how the column should be aligned, e.g. coldef="CCC"

        "L"     left
        "C"     center
        "R"     right

     An optional separator letter can occur between columns
     and should be one of + - or =, e.g. "C+C+C+C=C".
     Whitespace within coldef is ignored. By default, the
     columns are all centered.

     The ALIGN attribute alters the vertical position of the
     array as compared with preceding and following expressions.

     Use LDELIM and RDELIM attributes for delimiter entities.
     When the LABELS attribute is present, the array is
     displayed with the first row and the first column as
     labels displaced from the other elements. In this case,
     the first element of the first row should normally be
     left blank.

     Use &vdots; &cdots; and &ddots; for vertical, horizontal
     and diagonal ellipsis dots. Use &dotfill; to fill an array
```

```
340       cell with horizontal dots (e.g. for a full row).
341       Note &ldots; places the dots on the baseline, while &cdots;
342       places them higher up.
343   -->
344
345   <!ELEMENT ARRAY - - (ROW)+>
346   <!ATTLIST ARRAY
347           align (top|middle|bottom) middle -- vertical alignment --
348           coldef  CDATA   #IMPLIED  -- column alignment and separator --
349           ldelim  NAMES   #IMPLIED  -- stretchy left delimiter --
350           rdelim  NAMES   #IMPLIED  -- stretchy right delimiter --
351           labels (labels) #IMPLIED  -- TeX's \bordermatrix style -->
352
353   <!ELEMENT ROW - O (ITEM)*>
354   <!ELEMENT ITEM - O (%formula)*>
355   <!ATTLIST ITEM
356           align   CDATA  #IMPLIED  -- override coldef alignment --
357           colspan NUMBER 1         -- merge columns as per TABLE --
358           rowspan NUMBER 1         -- merge rows as per TABLE -->
```

Appendix F: The ISO-12083 mathematics DTD

This appendix shows the mathematics DTD `math.dtd` of the ISO 12083 DTD.

```
1    <!-- This is the ISO12083:1994 document type definition for Mathematics    -->
2
3    <!-- Copyright: (C) International Organization for Standardization 1994.
4    Permission to copy in any form is granted for use with conforming SGML
5    systems and applications as defined in ISO 8879:1986, provided this notice
6    is included in all copies.                                                -->
7
8    <!-- ================================================================== -->
9    <!--            PUBLIC DOCUMENT TYPE DEFINITION SUBSET                -->
10   <!-- ================================================================== -->
11
12   <!--
13   This DTD is included by the Book and Article DTDs of ISO12083:1994.
14   As it is a separate entity it may also be included by other DTDs.
15
16   Since there is no consensus on how to describe the semantics of formulas,
17   it only describes their presentational or visual structure. Since, however,
18   there is a strong need for such description (especially within the
19   print-disabled community), it is recommended that the following
20   declaration be added where there is a requirement for a consistent,
21   standardized mechanism to carry semantic meanings for the SGML
22   elements declared throughout this part of this International Standard:
23
24   <!ENTITY % SDAMAP       "SDAMAP   NAME    #IMPLIED"          >
25
26   and that the attribute represented by %SDAMAP; be made available for
27   all elements which may require a semantic association, or, in the simpler
28   case, be added to all elements in this DTD.                    -->
29
30
31
32   <!-- ================================================================== -->
33   <!--  Parameter entities describing the possible contents of formulas.    -->
34   <!-- ================================================================== -->
35
36   <!ENTITY % p.trans "bold|italic|sansser|typewrit|smallcap|roman"
37                      -- character transformations                          -->
38   <!ENTITY % m.math "fraction|subform|sup|inf|top|bottom|middle|fence|mark|
```

```
39      post|box|overline|undrline|radical|array|hspace|vspace|break|markref|
40      #PCDATA" -- mathematical formula elements                              -->
41
42
43
44    <!-- ================================================================== -->
45    <!-- Accessible Document and other Parameter Entities
46         If this DTD is not imbedded by a ISO12083:1994 Book or Article,
47         the comment delimiters should be removed.                          -->
48    <!-- ================================================================== -->
49
50    <!--ENTITY % SDAFORM      "SDAFORM   CDATA    #FIXED"                   -->
51    <!--ENTITY % SDARULE      "SDARULE   CDATA    #FIXED"                   -->
52    <!--ENTITY % SDAPREF      "SDAPREF   CDATA    #FIXED"                   -->
53    <!--ENTITY % SDASUFF      "SDASUFF   CDATA    #FIXED"                   -->
54    <!--ENTITY % SDASUSP      "SDASUSP   NAME     #FIXED"                   -->
55
56
57
58    <!-- ================================================================== -->
59    <!-- This entity is for an attribute to indicate which alphabet is
60         used in the element (formula, dformula). You may change this to
61         a notation attribute, where the notation could describe a
62         keyboard mapping. Please modify the set as necessary.
63         If this DTD is not imbedded by a ISO12083:1994 Book or Article,
64         the comment delimiters should be removed.                          -->
65    <!-- ================================================================== -->
66
67    <!-- ENTITY % a.types "(latin|greek|cyrillic|hebrew|kanji) latin"        -->
68
69
70    <!-- ================================================================== -->
71    <!-- character transformations                                          -->
72    <!-- ================================================================== -->
73
74    <!--      ELEMENT             MIN  CONTENT             EXPLANATIONS     -->
75    <!ELEMENT bold                - -  (%p.trans;|#PCDATA)* -- bold          -->
76    <!ELEMENT italic              - -  (%p.trans;|#PCDATA)* -- italic        -->
77    <!ELEMENT sansser             - -  (%p.trans;|#PCDATA)* -- sans serif    -->
78    <!ELEMENT typewrit            - -  (%p.trans;|#PCDATA)* -- typewriter    -->
79    <!ELEMENT smallcap            - -  (%p.trans;|#PCDATA)* -- small caps    -->
80    <!ELEMENT roman               - -  (%p.trans;|#PCDATA)* -- roman         -->
81
82
83    <!-- ================================================================== -->
84    <!-- Fractions                                                          -->
85    <!-- ================================================================== -->
86
87    <!--      ELEMENT             MIN  CONTENT              EXPLANATIONS    -->
88    <!ELEMENT fraction            - -  (num, den)      -- fraction          -->
89    <!ELEMENT num                 - -  (%p.trans;|%m.math;)* -- numerator    -->
90    <!ELEMENT den                 - -  (%p.trans;|%m.math;)* -- denominator  -->
91    <!--      ELEMENT   NAME      VALUE            DEFAULT                   -->
92    <!ATTLIST fraction  shape     (built|case) #IMPLIED
93                        align     (left|center|right)
94                                              center
95                        style     (single|double|triple|dash|dot|bold|blank|none)
96                                              single                      >
97
98
99
100   <!-- ================================================================== -->
101   <!-- Superiors, inferiors, accents, over and under                      -->
102   <!-- ================================================================== -->
```

```
103
104 <!--      ELEMENT              MIN  CONTENT              EXPLANATIONS     -->
105 <!ELEMENT sup                  - -  (%p.trans;|%m.math;)* -- superior      -->
106 <!ELEMENT inf                  - -  (%p.trans;|%m.math;)* -- inferior      -->
107 <!--      ELEMENT   NAME       VALUE            DEFAULT                   -->
108 <!ATTLIST sup       location   (pre|post)       post
109                     arrange    (compact|stagger)
110                                                 compact                    >
111 <!ATTLIST inf       location   (pre|post) post
112                     arrange    (compact|stagger) compact                   >
113
114
115 <!-- ====================================================================== -->
116 <!-- Embellishments                                                       -->
117 <!-- ====================================================================== -->
118
119 <!--      ELEMENT              MIN  CONTENT              EXPLANATIONS     -->
120 <!ELEMENT top                  - -  (%p.trans;|%m.math;)*
121                                                        -- top embellishment   -->
122 <!ELEMENT middle               - -  (%p.trans;|%m.math;)*
123                                                        -- middle, or "through" -->
124 <!ELEMENT bottom               - -  (%p.trans;|%m.math;)*
125                                                        -- bottom embellishment -->
126 <!--      ELEMENT   NAME       VALUE            DEFAULT                   -->
127 <!ATTLIST top       align      (left|center|right)
128                                                 center
129                     sizeid     ID               #IMPLIED
130                                                 -- to pass on the height   -->
131 <!ATTLIST middle    align      (left|center|right)
132                                                 center
133                     sizeid     ID               #IMPLIED
134                                                 -- to pass on the height   -->
135 <!ATTLIST bottom    align      (left|center|right)
136                                                 center
137                     sizeid    ID               #IMPLIED
138                                                 -- to pass on the height   -->
139
140
141 <!-- The subform element is defined later                                 -->
142
143
144
145 <!-- ====================================================================== -->
146 <!-- Fences, boxes, overlines and underlines                              -->
147 <!-- ====================================================================== -->
148
149 <!--      ELEMENT              MIN  CONTENT              EXPLANATIONS     -->
150 <!ELEMENT mark                 - O  EMPTY                                  >
151 <!ELEMENT fence                - -  (%p.trans;|%m.math;)* -- fence         -->
152 <!ELEMENT post                 - O  EMPTY            -- post               -->
153 <!ELEMENT box                  - -  (%p.trans;|%m.math;)* -- box           -->
154 <!ELEMENT overline             - -  (%p.trans;|%m.math;)* -- overline      -->
155 <!ELEMENT undrline             - -  (%p.trans;|%m.math;)* -- underline     -->
156 <!--      ELEMENT   NAME       VALUE            DEFAULT                   -->
157 <!ATTLIST mark      id         ID               #REQUIRED                   >
158 <!ATTLIST fence     lpost      CDATA            "|" -- left post           --
159                     rpost      CDATA            "|" -- right post          --
160                     style      (single|double|triple|dash|dot|bold|blank|none)
161                                                 single
162                     sizeid     ID               #IMPLIED
163                                                 -- to pass on the height   --
164                     sizeref    IDREF            #IMPLIED
165                                                 -- to pick up a height     -->
166 <!ATTLIST post      post       CDATA            "|"
```

```
                    style      (single|double|triple|dash|dot|bold|blank|none)
                                                 single
                    sizeid     ID                #IMPLIED
                                                 -- to pass on the height   --
                    sizeref    IDREF             #IMPLIED
                                                 -- to pick up a height     -->
<!ATTLIST box       style      (single|double|triple|dash|dot|bold|blank|none)
                                                 single                      >
<!ATTLIST overline  type       CDATA             "-" -- embellishment type --
                    style      (single|double|triple|dash|dot|bold|blank|none)
                                                 single
                    start      IDREF             #IMPLIED
                    end        IDREF             #IMPLIED                     >

<!ATTLIST undrline  type       CDATA             "_" -- embellishment
                                                                       type --
                    style      (single|double|triple|dash|dot|bold|blank|none)
                                                 single
                    start      IDREF             #IMPLIED
                    end        IDREF             #IMPLIED                     >


<!-- ================================================================== -->
<!-- Labelled arrows                                                    -->
<!-- ================================================================== -->

<!--      ELEMENT              MIN  CONTENT              EXPLANATIONS     -->
<!ELEMENT subform              - -  (%p.trans;|%m.math;)* -- base element  -->
<!--      ELEMENT   NAME       VALUE             DEFAULT                  -->
<!ATTLIST subform   sizeid     ID                #IMPLIED
                                                 -- to pass on a width, or
                                                 a height                 --
                    sizeref    IDREF             #IMPLIED
                                                 -- to pick up a width    -->


<!-- ================================================================== -->
<!-- Roots                                                              -->
<!-- ================================================================== -->

<!--      ELEMENT              MIN  CONTENT              EXPLANATIONS     -->
<!ELEMENT radical              - -  (radix?, radicand) -- root or radical  -->
<!ELEMENT radix                - -  (%p.trans;|%m.math;)* -- radix         -->
<!ELEMENT radicand             O O  (%p.trans;|%m.math;)* -- radicand      -->


<!-- ================================================================== -->
<!-- Arrays                                                             -->
<!-- ================================================================== -->

<!--      ELEMENT              MIN  CONTENT              EXPLANATIONS     -->
<!ELEMENT array                - -  (arrayrow+|arraycol+) -- array         -->
<!ELEMENT arrayrow             - O  (arraycel+)      -- array row          -->
<!ELEMENT arraycol             - O  (arraycel+)      -- array column       -->
<!ELEMENT arraycel             - O  (%p.trans;|%m.math;)* -- array cell    -->

<!--      ELEMENT   NAME       VALUE             DEFAULT                  -->
<!ATTLIST array     rowalign   NMTOKENS          #IMPLIED -- row alignment  --
                    colalign   NMTOKENS          #IMPLIED -- column
                                                                 alignment --
                    rowsep     NMTOKENS          #IMPLIED -- row separators --
                    colsep     NMTOKENS          #IMPLIED -- column
                                                                 separators -->

```

```
231
232 <!-- ================================================================== -->
233 <!-- Spacing                                                            -->
234 <!-- ================================================================== -->
235
236 <!--      ELEMENT              MIN  CONTENT              EXPLANATIONS     -->
237 <!ELEMENT hspace               - O EMPTY            -- horizontal spacing  -->
238 <!ELEMENT vspace               - O EMPTY            -- vertical   spacing  -->
239 <!ELEMENT break                - O EMPTY            -- turn line, break    -->
240 <!ELEMENT markref               - O EMPTY            -- hmark reference     -->
241
242 <!--      ELEMENT   NAME       VALUE            DEFAULT                    -->
243 <!ATTLIST hspace    space      CDATA            "1 mm"
244                                                    -- units as required   -->
245 <!ATTLIST vspace    space      CDATA            "1 mm"
246                                                    -- units as required   -->
247 <!ATTLIST markref   refid      IDREF            #REQUIRED
248                     direct     (hor|ver)        hor
249                                                  -- horizontal or vertical -->
250
251
252 <!-- ================================================================== -->
253 <!-- the formula elements                                               -->
254 <!-- ================================================================== -->
255
256 <!--      ELEMENT              MIN  CONTENT              EXPLANATIONS     -->
257 <!ELEMENT formula              - -  (%p.trans;|%m.math;)*
258                                                       -- in-line formula -->
259 <!ELEMENT dformula             - -  (%p.trans;|%m.math;)*
260                                                       -- display formula -->
261 <!ELEMENT dformgrp             - -  (formula|dformula)+
262                                                         -- display-formula group -->
263
264 <!--      ELEMENT   NAME       VALUE            DEFAULT                    -->
265 <!ATTLIST formula   id         ID               #IMPLIED
266                     alphabet   %a.types;
267 --                  %SDAPREF;      "<?SDATRANS>Inline formula"    --
268 --                  %SDASUSP;       "SUSPEND"        --
269 >
270 <!ATTLIST dformula  id         ID               #IMPLIED
271                     num        CDATA            #IMPLIED
272                     align      (left|center|right)
273                                                 center
274                     alphabet   %a.types;
275 --                  %SDAPREF;      "<?SDATRANS>Display formula" --
276 --                  %SDASUSP;       "SUSPEND"        --
277 >
278 <!ATTLIST dformgrp  id         ID               #IMPLIED
279                     num        CDATA            #IMPLIED
280                     align      (left|center|right)
281                                                 center
282 --                  %SDAPREF;      "<?SDATRANS>Display formula group" --
283
284 >
```

Appendix G: Example of a conversion of the DocBook DTD to HTML3

G.1 The original document marked up in the DocBook DTD

The listing below is part of the manual describing the DocBook DTD and is tagged according to that same DocBook DTD (V2.2.1).

```
<sect1><title>How to Get the DocBook DTD Online</title>

<para>
You can find the DocBook DTD and its documentation online in
the Davenport archive (<filename>/pub/davenport/docbook</filename>)
at <filename>ftp.ora.com</filename> (198.112.208.13).
</para>

<para>
This sample session shows how to retrieve the DTD and its documentation:

<screen>
<!-- could mark up the prompt in next line with computeroutput -->
<systemitem class="prompt">%</><userinput>ftp ftp.ora.com</>
<computeroutput>Connected to amber.ora.com.</>
<computeroutput>220 amber FTP server (Version wu-2.4(1) Fri Apr 15 14:14:30 EDT 1994) ready.</>
<computeroutput>Name (ftp.ora.com:terry): </><userinput>anonymous</>
<computeroutput>331 Guest login ok, send your complete e-mail address as password.</>
<computeroutput>Password: </><lineannotation>&larr; type e-mail address</>
<systemitem class="prompt">ftp&gt;</><userinput>cd pub/davenport/docbook</>
</screen>

The DocBook DTD and related ASCII files are in a file named
<filename>docbook.N.shar</>, where <emphasis>N</>
is the current revision number:

<screen>
<systemitem class="prompt">ftp&gt;</><userinput>get docbook.2.2.1.shar</>
</screen>

Most of these files also exist separately and may be ftp'd individually.
</para>

<para>
The <command>get</> command will put this ASCII shar file
on your system.  You must later unpack it on your system:
<screen>
<userinput>sh docbook.2.2.1.shar</>
</screen>
</para>
```

G.2 ESIS representation of the source document

The following is the ESIS representation of the same document produced by `nsgmls`.

```
AID IMPLIED
ALANG IMPLIED
AREMAP IMPLIED
AROLE IMPLIED
AXREFLABEL IMPLIED
ALABEL IMPLIED
ARENDERAS IMPLIED

(SECT1

AID IMPLIED
ALANG IMPLIED
AREMAP IMPLIED
AROLE IMPLIED
AXREFLABEL IMPLIED

APAGENUM IMPLIED

(TITLE

-How to Get the DocBook DTD
Online

)TITLE

AID IMPLIED
ALANG IMPLIED
AREMAP IMPLIED
AROLE IMPLIED
AXREFLABEL IMPLIED

(PARA

-You can find the DocBook DTD
and its documentation \nonline
in the Davenport archive \n(

AID IMPLIED
ALANG IMPLIED
AREMAP IMPLIED
AROLE IMPLIED
AXREFLABEL IMPLIED
AMOREINFO TOKEN NONE

(FILENAME
```

```
-/pub/davenport/docbook

)FILENAME

-) at \n

AID IMPLIED
ALANG IMPLIED
AREMAP IMPLIED
AROLE IMPLIED
AXREFLABEL IMPLIED
AMOREINFO TOKEN NONE

(FILENAME

-ftp.ora.com

)FILENAME

- (198.112.208.13).

)PARA

AID IMPLIED
ALANG IMPLIED
AREMAP IMPLIED
AROLE IMPLIED
AXREFLABEL IMPLIED

(PARA

-This sample session shows how
to retrieve the DTD\nand its
documentation:\n

AID IMPLIED
ALANG IMPLIED
AREMAP IMPLIED
AROLE IMPLIED
AXREFLABEL IMPLIED

sline ends and leading white
space must be preserved in
output

NLINESPECIFIC

AFORMAT NOTATION LINESPECIFIC
AWIDTH IMPLIED

(SCREEN

AID IMPLIED
ALANG IMPLIED
AREMAP IMPLIED
AROLE IMPLIED
AXREFLABEL IMPLIED
ACLASS TOKEN PROMPT
AMOREINFO TOKEN NONE

(SYSTEMITEM

-%

)SYSTEMITEM

AID IMPLIED
ALANG IMPLIED
AREMAP IMPLIED
AROLE IMPLIED
AXREFLABEL IMPLIED
AMOREINFO TOKEN NONE

(USERINPUT

-ftp ftp.ora.com

)USERINPUT

-\n

AID IMPLIED
ALANG IMPLIED
AREMAP IMPLIED
AROLE IMPLIED
AXREFLABEL IMPLIED
AMOREINFO TOKEN NONE

(COMPUTEROUTPUT

-Connected to amber.ora.com.

)COMPUTEROUTPUT

-\n

AID IMPLIED
ALANG IMPLIED
AREMAP IMPLIED
AROLE IMPLIED
AXREFLABEL IMPLIED
AMOREINFO TOKEN NONE

(COMPUTEROUTPUT

-220 amber FTP server (Version
wu-2.4(1) Fri Apr 15 14:14:30
EDT 1994) ready.

)COMPUTEROUTPUT

-\n

AID IMPLIED
ALANG IMPLIED
AREMAP IMPLIED
AROLE IMPLIED
AXREFLABEL IMPLIED
AMOREINFO TOKEN NONE

(COMPUTEROUTPUT

-Name (ftp.ora.com:terry):

)COMPUTEROUTPUT

AID IMPLIED
ALANG IMPLIED
AREMAP IMPLIED
AROLE IMPLIED
AXREFLABEL IMPLIED
AMOREINFO TOKEN NONE

(USERINPUT

-anonymous

)USERINPUT

-\n

AID IMPLIED
ALANG IMPLIED
AREMAP IMPLIED
AROLE IMPLIED
AXREFLABEL IMPLIED
AMOREINFO TOKEN NONE

(COMPUTEROUTPUT

-331 Guest login ok, send your
complete e-mail address as
password.

)COMPUTEROUTPUT

-\n

AID IMPLIED
ALANG IMPLIED
AREMAP IMPLIED
AROLE IMPLIED
AXREFLABEL IMPLIED
AMOREINFO TOKEN NONE

(COMPUTEROUTPUT

-Password:

)COMPUTEROUTPUT

AID IMPLIED
ALANG IMPLIED
AREMAP IMPLIED
AROLE IMPLIED
AXREFLABEL IMPLIED

(LINEANNOTATION

-\|[larr  ]\| type e-mail
address

)LINEANNOTATION

-\n

AID IMPLIED
ALANG IMPLIED
AREMAP IMPLIED
AROLE IMPLIED
AXREFLABEL IMPLIED
ACLASS TOKEN PROMPT
AMOREINFO TOKEN NONE

(SYSTEMITEM

-ftp\|[gt    ]\|

)SYSTEMITEM

AID IMPLIED
ALANG IMPLIED
AREMAP IMPLIED
AROLE IMPLIED
AXREFLABEL IMPLIED
AMOREINFO TOKEN NONE

(USERINPUT

-cd pub/davenport/docbook

)USERINPUT

)SCREEN

-\nThe DocBook DTD and related
ASCII files are in\na file
named
```

```
AID IMPLIED
ALANG IMPLIED
AREMAP IMPLIED
AROLE IMPLIED
AXREFLABEL IMPLIED
AMOREINFO TOKEN NONE

(FILENAME

-docbook.N.shar

)FILENAME

-, where

AID IMPLIED
ALANG IMPLIED
AREMAP IMPLIED
AROLE IMPLIED
AXREFLABEL IMPLIED

(EMPHASIS

-N

)EMPHASIS

-\nis the current revision
number:\n

AID IMPLIED
ALANG IMPLIED
AREMAP IMPLIED
AROLE IMPLIED
AXREFLABEL IMPLIED
AFORMAT NOTATION LINESPECIFIC
AWIDTH IMPLIED

(SCREEN

AID IMPLIED
ALANG IMPLIED
AREMAP IMPLIED
AROLE IMPLIED
AXREFLABEL IMPLIED
ACLASS TOKEN PROMPT
AMOREINFO TOKEN NONE

(SYSTEMITEM

-ftp\|[gt    ]\|

)SYSTEMITEM

AID IMPLIED
ALANG IMPLIED
AREMAP IMPLIED
AROLE IMPLIED
AXREFLABEL IMPLIED
AMOREINFO TOKEN NONE

(USERINPUT

-get docbook.2.2.1.shar

)USERINPUT

)SCREEN

-\nMost of these files\nalso
exist separately and may be
ftp'd individually.

)PARA

AID IMPLIED
ALANG IMPLIED
AREMAP IMPLIED
AROLE IMPLIED
AXREFLABEL IMPLIED

(PARA

-The

AID IMPLIED
ALANG IMPLIED
AREMAP IMPLIED
AROLE IMPLIED
AXREFLABEL IMPLIED
AMOREINFO TOKEN NONE

(COMMAND

-get

)COMMAND

- command will put this ASCII
shar \nfile on your system.
You must later unpack \nit on

your system:\n

AID IMPLIED
ALANG IMPLIED
AREMAP IMPLIED
AROLE IMPLIED
AXREFLABEL IMPLIED
AFORMAT NOTATION LINESPECIFIC
AWIDTH IMPLIED

(SCREEN

AID IMPLIED
ALANG IMPLIED
AREMAP IMPLIED
AROLE IMPLIED
AXREFLABEL IMPLIED
AMOREINFO TOKEN NONE

(USERINPUT

-sh docbook.2.2.1.shar

)USERINPUT

)SCREEN

)PARA
```

G.3 HTML3 output

The following presents the final HTML3 output resulting from the translation process.

```
<HTML>
<HEAD>
<TITLE>How to Get the DocBook DTD Online</TITLE>
</HEAD>
<BODY>
<H1>How to Get the DocBook DTD Online</H1>
You can find the DocBook DTD and its documentation online in the
Davenport archive (/pub/davenport/docbook) at ftp.ora.com
(198.112.208.13).<P>This sample session shows how to retrieve
the DTD and its documentation:
<pre>
%<i>ftp ftp.ora.com</i>
Connected to amber.ora.com.
220 amber FTP server (Version wu-2.4(1) Fri Apr 15 14:14:30 EDT 1994) ready.
Name (ftp.ora.com:terry): <i>anonymous</i>
331 Guest login ok, send your complete e-mail address as password.
Password:  type e-mail address
ftp&gt;<i>cd pub/davenport/docbook</i>
</pre>
The DocBook DTD and related ASCII files are in a file named docbook.N.shar,
where <STRONG>N</STRONG> is the current revision number:
<pre>
ftp&gt;<i>get docbook.2.2.1.shar</i>
```

```
</pre>
Most of these files also exist separately and may be ftp'd individually.
<P>
The get command will put this ASCII shar file on your system.
You must later unpack it on your system:
<pre>
<i>sh docbook.2.2.1.shar</i>
</pre>
</BODY>
</HTML>
```