

Techniques for using standard generalized markup language (SGML) —

(ISO/IEC/TR 9573:1988)

Committees responsible for this Draft for Development

The preparation of this Draft for Development was entrusted by the Information Systems Technology Standards Policy Committee (IST/-) to Technical Committee IST/18, upon which the following bodies were represented:

British Computer Society
British Facsimile Industry Consultative Committee
British Printing Industries Federation
British Telecommunications plc
Computing Services Association
Department of Trade and Industry (Information Technology Division)
EEA (The Association of Electronics Telecommunications and Business Equipment Industries)
HM Treasury (Central Computer and Telecommunications Agency)
Her Majesty's Stationery Office
Joint Network Team
Ministry of Defence
National Computer Users' Forum
National Computing Centre Ltd.
SGML Users' Group
University of London
User Standards Forum for Information Technology (Institute of Data Processing Management)

This Draft for Development, having been prepared under the direction of the Information Systems Technology Standards Policy Committee, was published under the authority of the Board of BSI and comes into effect on 28 February 1991

© BSI 05-2000

The following BSI reference relates to the work on this Draft for Development:
Committee reference IST/18

ISBN 0 580 19309 8

Amendments issued since publication

Amd. No.	Date	Comments

Contents

	Page
Committees responsible	Inside front cover
National foreword	ii
0 Introduction	1
1 Scope and Field of Application	1
2 References	1
3 Definitions	2
4 Document Analysis for the Creation of a Document Type Definition	2
5 General Document	20
6 Letter and Memorandum	58
7 Spreadsheet	64
8 Mathematics	74
9 Tables	88
10 Integration of Computer Graphics Metafiles	89
11 Japanese	94
12 Linguistic Application — Scandinavian Runes	100
13 European Multi-lingual Document	106
14 The Use of Short References	106
15 Mixing Text in Languages Written from Left to Right and from Right to Left	108
Figure 1 — Sample pages from the body of an International Standard	5
Figure 2 — Sample title page	11
Figure 3 — Overall structure of the basic document	21
Figure 4 — Structure of the h2 element	22
Figure 5 — Structure of the phrase (% m.ph), paragraph (% m.p), and paragraph sequence (% m.pseq) models	23
Figure 6 — Structure of the section subelements	24
Figure 7 — Structure of all phrases	25
Figure 8 — Structure of all paragraph/section subelements	26
Figure 9 — Structure of all floating elements	27
Figure 10 — Overall structure of a spreadsheet	64
Figure 11 — Structure of a formula in a spreadsheet	67
Figure 12 — Japanese: horizontal typesetting (text written from left to right)	95
Figure 13 — Japanese: vertical typesetting (text written from top to bottom)	96
Figure 14 — Japanese: source document	99
Table 1 — Annual Rainfall for Head City	89
Publication(s) referred to	Inside back cover

National foreword

This Draft for Development has been prepared under the direction of the Information Systems Technology Standards Policy Committee (IST/-). It is identical with ISO/IEC/TR 9573:1988 “*Information processing — SGML support facilities — Techniques for using SGML*”, published jointly by the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC).

The text of ISO/IEC/TR 9573:1988 was made available to purchasers under the reference number DD 181/PR. DD 181:1991 supersedes DD 181/PR which is withdrawn.

This Draft for Development provides guidance on the use of the standard generalized markup language (SGML) which is specified in BS 6868 identical to ISO 8879.

This guidance has been published as a Draft for Development for the same reasons as those that led to the publication by ISO/IEC of TR 9573 as a technical report rather than as an International Standard.

It is not expected that this Draft for Development will be converted into a British Standard at a later date. However, when TR 9573 is revised, this Draft for Development will also be reviewed and an announcement regarding its future will be made.

Cross-references

International Standard	Corresponding British Standard
ISO 646:1983	BS 4730:1985 <i>Specification for UK 7-bit coded character set</i> (Identical exercising national options)
ISO 2022:1986	BS 6856:1987 <i>Specification for code extension techniques for United Kingdom 7-bit and 8-bit coded character sets</i> (Identical)
ISO 4873:1986	BS 6006:1987 <i>Specification for structure and rules for implementation of United Kingdom 8-bit coded character set</i> (Identical)
	BS 6692 <i>Coded character sets for text communication</i>
ISO 6937-1:1983	Part 1:1986 <i>General introduction</i> (Identical)
ISO 6937-2:1983	Part 2:1986 <i>Specification for Latin alphabetic and non-alphabetic graphic characters</i> (Identical)
	BS 6945 <i>Computer graphics: metafile for the storage and transfer of picture description information (CGM)</i>
ISO 8632-2:1987	Part 2:1988 <i>Character encoding</i> (Identical)
ISO 8632-4:1987	Part 4:1988 <i>Clear text encoding</i> (Identical)
ISO 8879:1986	BS 6868:1987 <i>Specification for standard generalized markup language (SGML) for text and office systems</i> (Identical)
ISO 9069:1988	BS 7138:1989 <i>Guide to SGML Document Interchange Format (SDIF)</i> (Identical)

The Technical Committee has reviewed the provisions of ISO 8859-6:1987, ISO 9070:1990, JIS X 0201¹⁾ and JIS X 0208¹⁾, to which reference is made in the text, and has decided that they are acceptable for use in conjunction with this Draft for Development.

This publication is not be regarded as a British Standard.

It is being issued in the Draft for Development series of publications and is of a provisional nature because certain recommendations contained therein may need modification in the light of practical experience in its use. It should be applied on this provisional basis so that information and experience of its practical operation may be obtained.

A review of this Draft for Development will be carried out not later than two years after its publication. Notification of the start of the review period, with a request for the submission of comments from users of this Draft for Development, will be made in an announcement in the appropriate issue of *BSI News*. Observations and constructive proposals for improvement which it is felt should receive attention before the official call for comments will be welcomed. These should be addressed to the Secretary of BSI Technical Committee IST/18 at BSI, 2 Park Street London W1 2BS.

According to the replies received, the responsible BSI Committee will judge whether the Draft for Development can be converted into a British Standard or what other action should be taken.

Summary of pages

This document comprises a front cover, an inside front cover, pages i to iv, pages 1 to 112, an inside back cover and a back cover.

This standard has been updated (see copyright date) and may have had amendments incorporated. This will be indicated in the amendment table on the inside front cover.

¹⁾ Japanese national standards may be obtained from the Foreign Section, BSI Sales Department, Linford Wood, Milton Keynes MK14 6LE. Tel: 0908 220022.

0 Introduction

ISO 8879, *Information processing — Text and office systems — Standard Generalized Markup Language (SGML)*, states the rules for the description and markup of documents for their publication and interchange. A basic document type, primarily for computer-assisted publishing, is provided as an example in clause E.1 of ISO 8879, but is not explained there. It is explained in this Technical Report to assist comprehension. Also given is a variety of examples on the use of SGML. Thus this Technical Report is complementary to ISO 8879, its principal purpose being to assist in the adoption of the language.

1 Scope and Field of Application

This Technical Report complements ISO 8879 by providing additional tutorial information. It is not intended, and should not be regarded, as an extension, modification, or interpretation of ISO 8879. The SGML language contains a number of components, some of which are optional features. The tutorial information covers the main components of the language only; notably a discussion of LINK, CONCUR, and DATATAG is outside the scope of this Technical Report.

The intended audience is mainly document type designers already familiar with the basic concepts of SGML, but requiring more tutorial information on techniques for using SGML for various applications. Subclauses 5.3 and 8.4 are written in the style of a “User Guide” and can be used as a basis for end-user documentation. For an introductory tutorial on SGML the annexes of ISO 8879 can be used.

This Technical Report includes notes on the analysis of a document prior to the writing of a formal document type definition, and a series of examples.

The principal example is for a general document type, formally defined as an example in clause E.1 of ISO 8879. Others of a general nature are for letter and memorandum, spreadsheet, mathematics, and the mixing of text and graphics. Those for language applications include Scandinavian runes, Japanese, a European multilingual document, and mixing text in languages written from left to right and from right to left.

NOTE Throughout this Technical Report terms like “is keyed in”, and “with keyboarding” are used. This does not necessarily imply that the markup is to be added explicitly by a user; for text entry one would expect structured, context sensitive, editors to be used, or the markup added by application programs, e.g. in the case of interchange of spreadsheets.

2 References

- ISO 646, *Information processing — ISO 7-bit coded character set for information interchange*.
- ISO 2022, *Information processing — ISO 7-bit and 8-bit coded character sets — Code extension techniques*.
- ISO 4873, *Information processing — ISO 8-bit code for information interchange — Structure and rules for implementation*.
- ISO 6937, *Information processing — Coded character sets for text communication*.
- ISO 8632-2, *Information processing systems — Computer graphics — Metafile for the storage and transfer of picture description information — Part 2: Character encoding*.
- ISO 8632-4, *Information processing systems — Computer graphics — Metafile for the storage and transfer of picture description information — Part 4: Clear text encoding*.
- ISO 8859-6, *Information processing — 8-bit single-byte coded graphic character sets — Part 6: Latin/Arabic alphabet*.
- ISO 8879, *Information processing — Text and office systems — Standard Generalized Markup Language (SGML)*.
- ISO 9069, *Information processing — SGML support facilities — SGML Document Interchange Format (SDIF)²⁾*.
- ISO 9070, *Information processing — SGML support facilities — Registration procedures for public text owner identifiers²⁾*.
- JIS X 0201 (formerly C 6220), *Code for Information Interchange*.
- JIS X 0208 (formerly C 6226), *Code of the Japanese Graphic Character Set for Information Interchange*.

²⁾ At present at the stage of draft.

3 Definitions

The definitions in ISO 8879 also apply to this Technical Report.

4 Document Analysis for the Creation of a Document Type Definition

4.1 General notes

The analysis of the different classes of documents in an organization that is going to apply SGML is of critical importance for the success of the application. The task of creating a document type definition (DTD) to reflect the structure of an existing class of documents (reports for example) with a view to applying SGML just to continue the production of paper copies poses very few problems. More difficult, but very important, is to analyse the future need of identifying components of documents with a view to processing the documents in a different way, such as creating a database, searchable with standard DBMS software, or multimedia publishing (for example microfiche and CD ROM), or producing secondary publications (for example of titles and abstracts only).

4.1.1 Steps of document analysis

The initial step of the analysis is to identify the different Document Type Definitions that are required. One area that requires careful consideration is whether a particular class of documents really has unique elements and structure (and thus requiring a separate DTD), or whether it differs from another class only in the presentation — layout — style. Consider, for example, one document class where the front matter consists of

- one page with title, author's name, and a reference number,
 - one page with an abstract,
- and another class where part of the first page consists of
- title, author's name, and a reference number,
 - an abstract,

followed, on the same page, by the body of the document. In most cases, in spite of the apparent differences, these two classes of documents are really only one document type with two styles of presentation.

The first step, performed for each type of document, is to identify the elements and any attributes required for each element. Techniques that can be applied in this area are discussed in more detail below for some typical parts of documents. In this analysis it is highly desirable to identify the elements that are common to more than one document type, such as paragraphs, footnotes, lists, and headings, in order to make a consistent set of document type definitions, which facilitates the keyboarding and makes it possible to merge parts of documents of different types more easily. Within any organization it is very likely that the body elements of many types of documents are identical, and differences only occur in the front and back matter. Thus the body elements of a letter class may well be the same as those of a report, whereas the “front matter” of a letter would be addressee, sender, date, and reference number, and that of a report, title, author, abstract, and table of contents.

The second step is to identify the hierarchical structure of the elements, starting from the complete document, then dividing it into major elements that in their turn contain other elements, and so on down to the character (text) level.

The third step is to evaluate the use of short references.

Once these three steps have been completed, the formal document type definition can be created with relative ease.

4.1.2 Design issues

In the design of the elements and attributes for a document type, the possibility of text that is automatically generated by the application software/text formatter should be considered. In this way a number of tasks will be facilitated when a document is revised or when parts of several documents are merged. Common cases are listed below:

- a) Numbering of elements such as headings, figures, tables, and footnotes. Rather than tagging

```
<head>2.1 Widget Factory
...
...
described in 2.1 ...
```

it is preferable to design the document type definition in such a way as to tag

```
<head id=widget>Widget Factory
...
...
described in <hdref refid=widget> . . .
```

and let the application software (or text formatter) number the headings and resolve the cross-references. It should be noted that the HDREF tag does not specify the form of the reference and it could have generated “Widget Factory on page 22”, or “section 2.1 on page 22” instead. Thus even the form of cross-references is, just as the presentation style, independent of the SGML source document.

b) Creation of the table of contents, and lists of figures and tables. It is preferable that the application software (or text formatter) accumulates the entries and that only the position where the information should be placed in the document is indicated by a tag such as <toc>, <figlist>, and <tablist>. In most document types the position of these elements in the document is always fixed and these tags can be omitted using tag minimization.

c) Creation of an index. It is preferable to be able to insert tags such as

```
<i1>words to be put in index</i1>
```

in the text and have (correct) page numbers added automatically and the entries collected and sorted by the text formatter and printed in a place indicated by an <index> tag, rather than having to create manually the sorted index entries with appropriate page numbers that have to be updated for any revision of the text.

There are also a number of issues that are technically less important, but that can be very important for the ease of use of an SGML application. These issues include the following:

a) Keystroke minimization in generic identifiers. If the tags are keyed by the author, it is preferable that the generic identifiers for those tags that are not very commonly used are longer and easier to remember rather than being short and perhaps more cryptic. For example <isbn> is preferable to <bn>, even if it costs two more keystrokes. For commonly used tags, for example for paragraphs, short tags are preferable. If the tags are keyed by copy-typists, the keystroke minimization savings may be more important than the generic identifiers being easy to remember. However, a list of generic identifiers and attributes could be placed next to the keyboard for easy reference. If the tags are generated by function keys or similar, the advantages of minimizing keystrokes are only small savings on overheads for storage and transmission. In this Technical Report, generic identifiers are of the form used in clause E.1 of ISO 8879.

b) Tag minimization. With parsers supporting the OMITTAG feature, many tags can be omitted. Thus instead of keyboarding

```
<h1><h1t>Widgets</h1t>
<p>Widgets are . . .</p>
</h1>
```

the DTD would be constructed to permit

```
<h1>Widgets
<p>Widgets are . . .
```

The design of a document type definition has, however, to strike a good balance, in cases where a tag can only be omitted under certain conditions, between the saving in keystrokes and the extra effort required to learn whether a tag can be omitted in a particular circumstance. One example where tag minimization has been used excessively is lists where the document type definition has been designed to make it possible to omit the tag for the first list item.

```
<ul>
First list item.
<li>Second list item.
<li>Last list item.
</ul>
```

NOTE The tag identifies an “unordered list” (see 5.3.10), and identifies each “list item”.

It should be noted, however, that it is never an error to key a tag that could have been omitted.

c) Attribute minimization. With parsers supporting the SHORTTAG feature, the attribute name can be omitted from the specification of an attribute value if the value is limited to those of a name token group. In order to ease the use, however, the designer has to bear this in mind in order to make the values meaningful. Thus the definition

<! - -	ELEMENTS	NAME	VALUE	DEFAULT	- ->
<!ATTLIST	tabref	page	(page no)	no	
		refid	IDREF	#REQUIRED	>

with the keying

<tabref refid=x page> (or <tabref refid=x page=page>)

is preferable to

<! - -	ELEMENTS	NAME	VALUE	DEFAULT	- ->
<!ATTLIST	tabref	page	(yes no)	no	
		refid	IDREF	#REQUIRED	>

with the keying

<tabref refid=x yes> (or <tabref refid=x page=yes>)

d) Short references. With parsers supporting the SHORTREF feature, some elements may be tagged automatically during the parsing. For example, paired “” characters could be used as a convenient way of entering <q> and </q> tags. A further discussion of this feature, applied to quotes, is given in clause 14.

Whilst SGML is designed to permit the widest possible variety of applications, it is also clear that if all applications use different tags for common elements, text interchange will not be facilitated by the use of SGML. It is thus hoped that designers of document type definitions will study those publicly available that are of good quality and use the same definitions for elements that are common to both applications.

4.2 Example of the analysis of body text

Consider the pages of body text shown in Figure 1. The following elements can readily be identified:

- A, B, and C: headed sections. Call these elements H2, H3, and H4 respectively. An H1 element is also present in other parts of the document.
- D: paragraph. Call this element P.
- E: note. Call this element NOTE.
- F and G: numbered (i.e. ordered), sometimes nested, lists. Call this type of list element OL.
- H: figure. Call this element FIG.
- I: footnote. Call this element FN.
- J: reference to footnote. Call this element FNREF.
- K: list of terms and their definitions, a glossary in fact. Call this element GL.
- L: reference to heading number. Call this element HDREF.
- M: reference to figure. Call this element FIGREF.

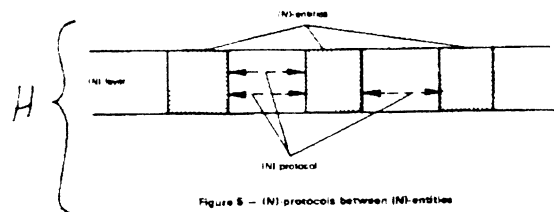


Figure 5 - (N)-protocols between (N)-entities

A 5.3 Communication between peer entities

B 5.3.1 Definitions

5.3.1.1 (N)-connection: An association established by the (N)-layer between two or more (N+1)-entities for the transfer of data.

5.3.1.2 (N)-connection-endpoint: A terminator at one end of an (N)-connection within an (N)-service access point.

5.3.1.3 multi endpoint connection: A connection with more than two connection-endpoints.

5.3.1.4 correspondent (N)-entities: (N)-entities with an (N+1)-connection between them.

5.3.1.5 (N)-relay: An (N)-function by means of which an (N)-entity forwards data received from one correspondent (N)-entity to another correspondent (N)-entity.

5.3.1.6 (N)-data source: An (N)-entity that sends (N+1)-service data units (see 5.6.1.7) on an (N+1)-connection.¹⁾

5.3.1.7 (N)-data sink: An (N)-entity that receives (N+1)-service data units on an (N+1)-connection.¹⁾

5.3.1.8 (N)-data transmission: An (N)-facility which conveys (N)-service data units from one (N+1)-entity to one or more (N+1)-entities.¹⁾

5.3.1.9 (N)-duplex transmission: (N)-data transmission in both directions at the same time.¹⁾

5.3.1.10 (N)-half duplex transmission: (N)-data transmission in either direction, one direction at a time; the choice of direction is controlled by an (N+1)-entity.¹⁾

¹⁾ These definitions are not for use in this International Standard but are for use in future OSI standards.

5.3.1.11 (N)-simplex transmission: (N)-data transmission in one pre-assigned direction.¹⁾

5.3.1.12 (N)-data communication: An (N)-function which transfers (N)-protocol data units (see 5.6.1.3) according to an (N)-protocol over one or more (N+1)-connections.¹⁾

5.3.1.13 (N)-two-way simultaneous communication: (N)-data communication in both directions at the same time.

5.3.1.14 (N)-two-way alternate communication: (N)-data communication in both directions, one direction at a time.

5.3.1.15 (N)-one-way communication: (N)-data communication in one pre-assigned direction.

B 5.3.2 Description

For information to be exchanged between two or more (N+1)-entities, an association shall be established between them in the (N)-layer using an (N)-protocol.

NOTE - Classes of protocols may be defined within the (N)-protocols. The precise definition of the term 'classes of protocols' is for further study.

This association is called an (N)-connection. (N)-connections are provided by the (N)-layer between two or more (N)-service access points. The terminator of an (N)-connection at an (N)-service access point is called an (N)-connection endpoint. A connection with more than two connection endpoints is termed a multi-endpoint connection. (N)-entities with a connection between them are termed correspondent (N)-entities.

(N+1)-entities can communicate only by using the services of the (N)-layer. There are instances where services provided by the (N)-layer do not permit direct access between all of the (N+1)-entities which have to communicate. If this is the case, communication can still occur if some other (N+1)-entity can act as a relay between them (see figure 6).

The fact that communication is relayed by a chain of (N+1)-entities is known neither by the (N)-layer nor by the (N+2)-layer.

5.9.1.2 application management application-entity: An application entity which executes application management functions.

5.9.1.3 OSI resources: Data processing and data communication resources which are of concern to OSI.

5.9.1.4 systems management: Functions in the Application Layer related to the management of various OSI resources and their status across all layers of the OSI architecture.

5.9.1.5 systems management application-entity: An application entity which executes systems management functions.

5.9.1.6 layer management: Functions related to the management of the (N)-layer partly performed in the (N)-layer itself according to the (N)-protocol of the layer activities such as activation and error control and partly performed as a subset of systems management.

B 5.9.2 Introduction

Within the OSI architecture there is a need to recognize the special problems of initiating, terminating, and monitoring activities and assisting in their harmonious operations, as well as handling abnormal conditions. These have been collectively considered as the management aspects of the OSI architecture. These concepts are essential to the operation of the interconnected open systems and therefore are included in the comprehensive description of the Reference Model described in subsequent clauses of this International Standard.

The management activities which are of concern are those which imply actual exchanges of information between open systems. Only the protocols needed to conduct such exchanges are candidates for standardization within the OSI architecture.

This clause describes key concepts relevant to the management aspects, including the different categories of management activities and the positioning of such activities within the OSI architecture.

B 5.9.3 Categories of management activities

Only those management activities which imply actual exchanges of information between remote management entities are pertinent to the OSI architecture. Other management activities local to particular open systems are outside its scope.

Similarly, not all resources are pertinent to OSI. This International Standard considers only OSI resources, i.e. those data processing and data communication resources which are of concern to OSI.

The following categories of management activities are identified:

- a) application management;
- b) systems management; and
- c) layer management.

5.9.3.1 Application management

Application management relates to the management of OSI application processes. The following list is typical of activities which fall into this category but it is not exhaustive.

- a) initialization of parameters representing application processes;
- b) initiation, maintenance and termination of application processes;
- c) allocation and de-allocation of OSI resources to application processes;
- d) detection and prevention of OSI resource interference and deadlock;
- e) integrity and commitment control;
- f) security control; and
- g) checkpointing and recovery control.

The protocols for application management reside within the Application Layer, and are handled by application management application entities.

5.9.3.2 Systems management

Systems management relates to the management of OSI resources and their status across all layers of the OSI architecture. The following list is typical of activities which fall into this category but it is not exhaustive.

- a) activation/deactivation management which includes:
 - 1) activation, maintenance and termination of OSI resources distributed in open systems, including physical media for OSI;
 - 2) some program loading functions;
 - 3) establishment/maintenance/release of connections between management entities; and
 - 4) open systems parameter initialization/modification;
- b) monitoring which includes:
 - 1) reporting status or status changes; and
 - 2) reporting statistics; and
- c) error control which includes:
 - 1) error detection and some of the diagnostic functions; and
 - 2) reconfiguration and restart.

The protocols for systems management reside in the application layer, and are handled by systems management application entities.

Figure 1 — Sample pages from the body of an International Standard

In order to establish links between the elements that are references and the elements that are referred to, an ID attribute is needed for the heading, figure, and footnote elements, and a REFID attribute is required for the references to heading, figure, and footnote elements.

4.2.1 Headed sections

The headed section elements all have the same structure: a heading title followed by the content of that section. The content of the heading titles is just characters.

NOTE The hierarchical structure of document elements used in this example means that the content of a headed section lasts until the next headed section of the same level or higher level, rather than ending at the end of the heading title.

Example of keying:

```
<h2>Communication between peer-entities
<h3>Definitions
```

The content of each level is optional and repeatable paragraphs, and similar elements. The parameter entity %s.zz is used for these (defined below), followed by optional and repeatable headings of a lower level. The element definition is thus

```
<!-- ELEMENTS MIN CONTENT (EXCEPTIONS) -->
<!ELEMENT h1 - 0 (h1t, (%s.zz;)*,h2*) -- clause -->
<!ELEMENT h2 - 0 (h2t, (%s.zz;)*,h3*) -- subclause -->
<!ELEMENT h3 - 0 (h3t, (%s.zz;)*,h4*) -- subsubclause -->
<!ELEMENT h4 - 0 (h4t, (%s.zz;)* -- subsubsubclause-->
<!ELEMENT (h1t|h2t|h3t|h4t)
0 0 (#PCDATA) -- heading title -->
```

NOTE 1 The markup minimization for the elements in this example is discussed in 4.2.10.

NOTE 2 In many applications there would be no need to differentiate between heading titles of different levels and thus “th” could be used instead of “h1t”, “h2t”, “h3t”, and “h4t”.

The content of %s.zz has been identified in this body text as p and note, ol and gl. The following parameter entities can be defined to describe this:

```
<!ENTITY % ps.ul.d “ol” -- unit-item lists -->
<!ENTITY % ps.list “%ps.ul.d;|gl” -- all lists -->
<!ENTITY % ps.zz “(%ps.list;)” -- para/sect subelements -->
<!ENTITY % s.p.d “p|note” -- simple paragraphs -->
<!ENTITY % s.zz “(%s.p.d;)|(%ps.zz;)” -- sect subelements -->
```

NOTE The multiple levels of parameter entities shown here may at first appear too complex for this simple example of body text. It is shown below, however, that it is very convenient to structure the definition with the use of parameter entities. The naming convention used is explained in 5.5.

4.2.2 Paragraph and note elements

Example of keying of paragraph and note elements:

```
<p>For information to be exchanged between two or more . . .
using an (N)-protocol.
<note>Classes of protocols may be defined within the (N)-protocols.
<p>This association . . .
```

Paragraphs and notes are very similar in structure and can be defined as

```
<!-- ELEMENTS MIN CONTENT (EXCEPTIONS) -->
<!ELEMENT (%s.p.d;) 0 0 %m.p; -- paragraphs and notes -->
```

The content of these two elements has been identified to be characters, ol, href, figref, and fnref. Thus the following can be defined:

```
<!ENTITY % p.rf.ph “href|figref” -- header and figure refs -->
<!ENTITY % p.rf.d “fnref” -- footnote reference -->
<!ENTITY % p.zz.ph “(%p.rf.ph;)|(%p.rf.d)” -- all references -->
<!ENTITY % m.p “(#PCDATA|(%p.zz.ph;)|(%ps.zz;))*” -->
```

4.2.3 List elements

Two types of list have been identified: an ordered list whose items are ordered and will be numbered in some kind of sequence, and a glossary list whose items comprise a term followed by a definition of that term.

Two approaches to the design of list constructs are described here. In the first approach, each type of list is defined as a different element. In the second, only one (or a very few) forms of list elements are defined and the different kinds of lists are distinguished by attributes. The merits of the two approaches are discussed at the end of this subclause.

4.2.3.1 Separately identified elements

Call the two types of lists: ol and gl. The ol (ordered list) requires an identification of the start and end of the list, an identification of each element in the list (call these li — list item), and a mechanism for nesting the lists. Each list item can contain more than one paragraph. The elements can be defined as

```
<!--      ELEMENTS      MIN  CONTENT (EXCEPTIONS) -->
<!ELEMENT  (%ps.ul.d;)  -   -   (li*)                -- unit item lists -->
<!ELEMENT  li           -   0   %m.pseq;              -- list item  -->
```

with % m.pseq defined as

```
<!ENTITY   %   m.pseq      "(p, ((%s.p.d;) | (%ps.zz;))*)"      -- para seq -->
```

and keyed

```
<ol>
<li>activation/deactivation management which includes:
<ol>
<li>activation, maintenance and termination of OSI resources distributed in open systems, including physical media for OSI;
...
<li>open systems parameter initialization/modification;
</ol>
<li>monitoring which includes:
...
</ol>
```

NOTE Another approach to the nesting of lists would be to define different generic identifiers for each level. In the keying of the tags it may thus be clearer which is the current level. With the nesting of different types of lists, however, this advantage is outweighed by the complexity caused by the multitude of generic identifiers required.

The gl (glossary list) is similar to the ordered list, except that each list item is composed of a term followed by its definition. Thus the elements can be defined:

```
<!--      ELEMENTS      MIN  CONTENT (EXCEPTIONS) -->
<!ELEMENT  gl           -   -   ((gt,gd)*)            -- glossary list -->
<!ELEMENT  gt           -   0   (#PCDATA)             -- glossary term-->
<!ELEMENT  gd           -   0   %m.pseq;              -- glossary definition -->
```

NOTE The parameter entity %m.pseq is re-used, since the content of both the li element and the gd element is the same.

and keyed

```
<gl>
<gt>(N) -connection
<gd>An association established by the (N) -layer between two or more (N+1) -entities for the transfer of data.
<gt>(N) -connection-endpoint
<gd>A terminator at one end of an (N) -connection within an (N)-service-access-point.
...
</gl>
```

4.2.3.2 Common list element

In this design, a single list element is defined with one attribute that indicates the form of the list and another that describes whether the list forms any kind of sequence:

```
<!--      ELEMENTS      MIN  CONTENT      (EXCEPTIONS) -->
<!ELEMENT  list         -   -   ((lt?, li)+)          -- lists      -->
<!ELEMENT  lt           -   0   (#PCDATA)            -- list term  -->
<!ELEMENT  li           -   0   (%m.pseq;)           -- list item  -->
<!--      ELEMENTS      NAME    VALUES      DEFAULT    -->
<!ATTLIST  list         form      (display | gloss)    display
                                seq      (ord | unord)    unord          >
```

With this design, the two types of list might be keyed (and here we show an ordered list within a glossary list, for variety):

```
<list gloss>
<lt>colour
<li>The colour of the transparent background.
```

This is expressed as three percentages, each representing a primary colour:

```
<list ord>
<li>Red
<li>Green
<li>Blue
</list>
<lt>transmissibility
<li>The amount of light that will be transmitted through the background, expressed as a percentage.
</list>
```

4.2.3.3 Discussion of the two approaches to lists

The first design approach has the advantage of fewer keystrokes, and may be appropriate in a very well-defined or bounded application. The second approach is more suited to general, or evolving, applications:

— Treating the attributes of the lists as true SGML attributes gives flexibility. For example, if one were to provide the option of a glossary in which the items were numbered, then the markup:

```
<list gloss ord>
```

follows naturally from that already described. With the first approach one would need to define a new element, or add an “ordered” attribute to the *gl* element (which would be inconsistent with the use of “ordered” as part of the GI of the *ol* element).

Similarly, a “flowed” list (in which the list is flowed as part of a sentence or paragraph) can be supported by defining *flow* as an alternative to *display* for the *form* = attribute.

— If a document is being edited without the aid of an SGML syntax-directed editing system it is easier to change a list from one type to another if the end-tag does not need alteration.

On the other hand this approach implies that in implementing an SGML syntax-directed editing system it is not sufficient to use the DTD to “prompt” and verify valid sub-elements, since certain combinations of attributes imply that only a subset of the sub-elements permitted by the DTD are actually valid. In a `<list unord>` element only the `<li>` sub-elements are valid whereas the DTD also permits `<lt>` sub-elements to occur.

4.2.4 Footnote element

Footnotes require identification of the start and end of the element, and an optional id attribute for referencing. The element can be defined:

```
<!-- ELEMENTS MIN CONTENT (EXCEPTIONS) -->
<!ELEMENT fn - - %m.pseq; -(%i.float;) -- footnote -->
<!-- ELEMENTS NAME VALUE DEFAULT -->
<!ATTLIST fn id ID #IMPLIED >
```

and keyed

... (N-1) -connection. `<fn id=futosi>` These definitions are not for use in this International Standard but are for use in future OSI standards.

```
</fn>
```

NOTE The `-(%i.float;)` is explained below.

4.2.5 Figure element

The figure element in this example is composed of a figure body and a figure caption. The figure body is text and/or a drawing. The definition of the element can be

```
<!-- ELEMENTS MIN CONTENT (EXCEPTIONS) -->
<!ELEMENT fig - - (figbody, figcap) -(%i.float;) >
<!ELEMENT figbody 0 0 ((artwork | %m.pseq;)* >
<!ELEMENT figcap - 0 (#PCDATA) >
<!ELEMENT artwork - 0 EMPTY >
<!-- ELEMENTS NAME VALUE DEFAULT -->
<!ATTLIST fig id ID #IMPLIED >
<!ATTLIST artwork name ENTITY #REQUIRED >
```

and keyed

```
<fig>
<artwork name=nprotcl>
<figcap>(N) -protocol between (N) -entities
</fig>
```

4.2.6 Floating elements

The placement of figures and footnotes is, in general, not fixed; they are floating elements. They are best defined as includable in any place, except inside a figure or footnote. They can thus be defined as

```
<!ENTITY % doctype "standard" >
<!ENTITY % i.float "fig | fn" -- floating elements -->
<!-- ELEMENTS MIN CONTENT (EXCEPTIONS) -->
<!ELEMENT %doctype - - (frontm, body, annex?) +(%i.float;)>
```

4.2.7 Reference elements

References to headings, figures, and footnotes all have the same structure. Their content is “empty”, i.e. it is generated by the text formatter or the application software. These elements can be defined as

```
<!-- ELEMENTS MIN CONTENT (EXCEPTIONS) -->
<!ELEMENT (%p.rf.ph; | %p.rf.d;)
- 0 EMPTY -- generated references -->
<!-- ELEMENTS NAME VALUE DEFAULT -->
<!ATTLIST (%p.rf.ph; | %p.rf.d;)
refid IDREF #REQUIRED >
```

and keyed

... a relay between them <figref refid=figxy>.

4.2.8 Summary of the definition of the body element

Based on the two pages of body text shown in Figure 1, the document type definition for the body components would be

```
<!-- parameter entities -->
<!ENTITY % ps.ul.d "ol" -- unit-item lists -->
<!ENTITY % ps.list "%ps.ul.d; | gl" -- all lists -->
<!ENTITY % ps.zz "(%ps.list;)" -- para/sect subelements -->
<!ENTITY % s.p.d "p | note" -- simple paragraphs -->
<!ENTITY % s.zz "(%s.p.d;) | (%ps.zz;)" -- sect subelements -->
<!ENTITY % p.rf.ph "href | figref" -- header and figure refs -->
<!ENTITY % p.rf.d "fnref" -- footnote reference -->
<!ENTITY % p.zz.ph "(%p.rf.ph;) | (%p.rf.d)" -- all references -->
<!ENTITY % m.p "(#PCDATA | (%p.zz.ph;) | (%ps.zz;))*" >
<!ENTITY % m.pseq "(p,((%s.p.d;) | (%ps.zz;))*)" -- para seq -->
<!ENTITY % i.float "fig | fn" -- floating elements -->
<!-- ELEMENTS MIN CONTENT (EXCEPTIONS) -->
<!-- headings -->
<!ELEMENT h1 - 0 (h1t, (%s.zz;)*, h2*) -- clause -->
<!ELEMENT h2 - 0 (h2t, (%s.zz;)*, h3*) -- subclause -->
<!ELEMENT h3 - 0 (h3t, (%s.zz;)*, h4*) -- subsubclause -->
<!ELEMENT h4 - 0 (h4t, (%s.zz;)* -- subsubsubclause -->
<!ELEMENT (h1t | h2t | h3t | h4t)
0 0 (#PCDATA) -- heading title -->
<!-- paragraphs and notes -->
<!ELEMENT (%s.p.d;) 0 0 %m.p; -- paragraphs and notes -->
<!ELEMENT fn - - %m.pseq; -(%i.float;) -- footnote -->
<!-- lists -->
<!ELEMENT (%ps.ul.d;) - - (li*) -- unit item lists -->
<!ELEMENT li - 0 %m.pseq; -- list item -->
<!ELEMENT gl - - ((gt, gd)*) -- glossary list -->
<!ELEMENT gt - 0 (#PCDATA) -- glossary term -->
<!ELEMENT gd - 0 %m.pseq; -- glossary definition -->
<!-- figure -->
<!ELEMENT fig - - (figbody, figcap) -(%i.float;) >
<!ELEMENT figbody 0 0 ((artwork | %m.pseq;)* >
<!ELEMENT figcap - 0 (#PCDATA) >
<!ELEMENT artwork - 0 EMPTY >
<!-- references -->
<!ELEMENT (%p.rf.ph; | %p.rf.d;)
- 0 EMPTY -- generated references -->
```

<!--	ELEMENTS	NAME	VALUE	DEFAULT	-->
<!ATTLIST	fn	id	ID	#IMPLIED	>
<!ATTLIST	fig	id	ID	#IMPLIED	>
<!ATTLIST	artwork	name	ENTITY	#REQUIRED	>
<!ATTLIST	(%p.rf.ph; %p.rf.d;)	refid	IDREF	#REQUIRED	>

4.2.9 Additional elements

A further analysis of more documents of the same type, however, reveals additional elements that need incorporating in the definition, for example

- List of notes. Call this element NL (note list).
- List in which entries are not identified by alphanumeric characters but which are each preceded by, for example, a dash. Call this element UL (unordered list).
- Examples. Call this element XMP.
- References to headings, figures, tables, and footnotes in external documents. Call this element EXTREF.

The two new types of lists encountered have the same structure as the ol already included. Owing to the use of parameter entities in the definition, the addition of these two elements can be accomplished by merely changing the definition of %ps.ul.d to

```
<!ENTITY % ps.ul.d "ol|ul|nl" -- unit-item lists -->
```

The example element on the other hand requires the addition of a new element:

```
<!-- ELEMENTS MIN CONTENT (EXCEPTIONS) -->
<!ELEMENT xmp -- %m.pseq; -- example -->
```

For the references to the elements of external documents, the extref element can be defined:

```
<!-- ELEMENTS MIN CONTENT (EXCEPTIONS) -->
<!ELEMENT extref - 0 EMPTY -- reference to external doc. -->
<!-- ELEMENTS NAME VALUE DEFAULT -->
<!ATTLIST extref docnum CDATA #REQUIRED
refid IDREF #REQUIRED
type (hd|fig|tab|fn) hd >
```

NOTE To resolve the reference, the application software/text formatter will need to interrogate a database of external references. Similarly, the application software/text formatter is required to update this database with information on all elements with an ID attribute, and the use of any references to other documents, for each document formatted. This information will, in addition, assist in determining the impact of a revision of a document on other documents.

4.2.10 Markup minimization

The possibility of markup minimization by the use of OMITTAG is more a question of taste rather than a technical issue. Experience has shown that the definition of tags that can be omitted in this example strikes a good balance between savings in keystrokes, and consistency and ease of learning the omissible tags. It is, however, by no means the only good scheme.

4.3 Example of the analysis of title matter

Consider the title matter in Figure 2. Seven main elements can readily be identified: title (A), list of authors (B), list of institutes (C), date (D), abstract (E), notes (F), and publication and copyright information (G). For this kind of document all these elements are required and thus the title matter element can be defined as

```
<!-- ELEMENTS MIN CONTENT (EXCEPTIONS) -->
<!ELEMENT titlem - - (title, authlist, instlist, date, abstract) >
```

leaving the notes (F) and publication and copyright information (G) aside for the moment.

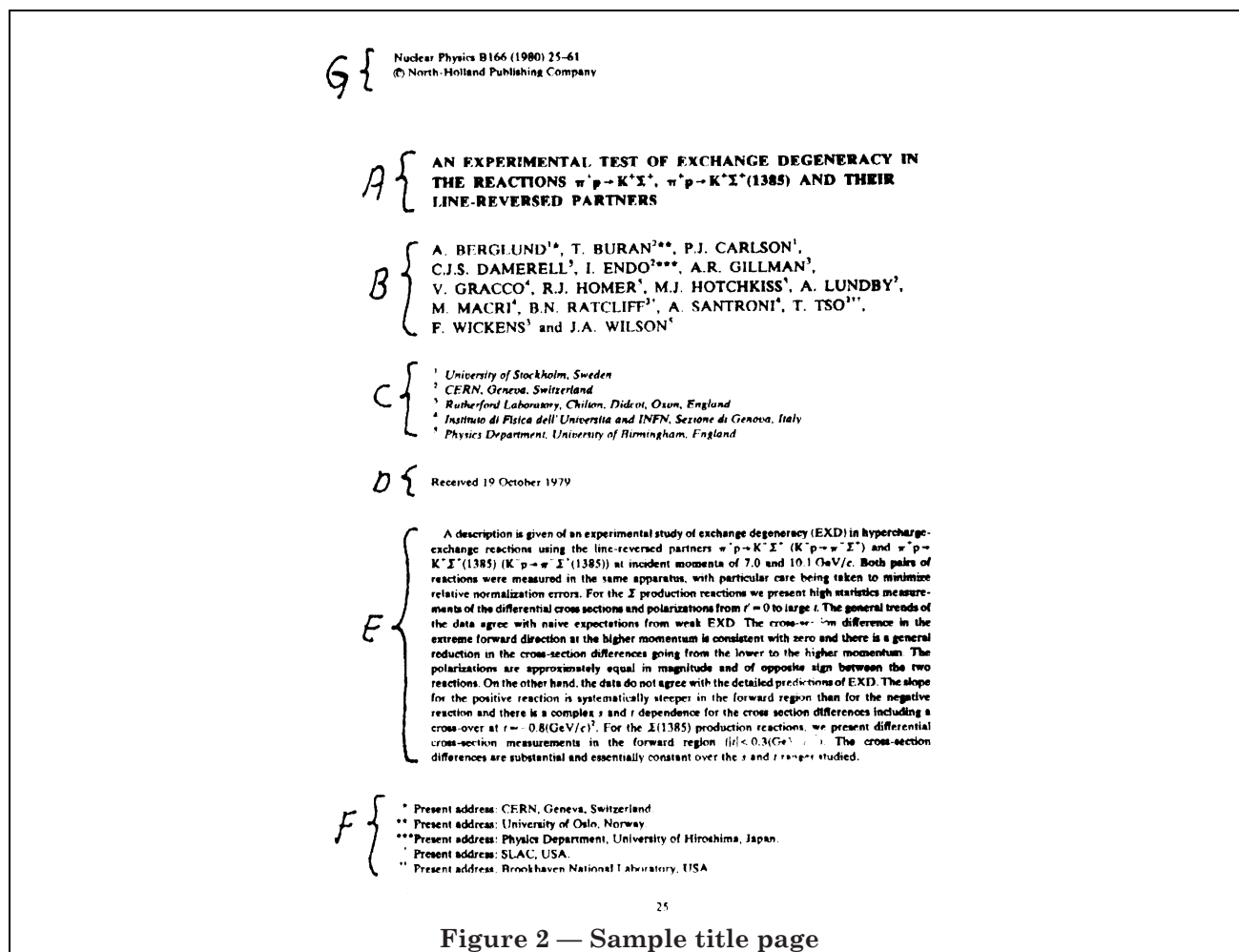


Figure 2 — Sample title page

4.3.1 Title element

A { AN EXPERIMENTAL TEST OF EXCHANGE DEGENERACY IN THE REACTIONS $\pi^+p \rightarrow K^+\Sigma^+$, $\pi^+p \rightarrow K^+\Sigma^+(1385)$ AND THEIR LINE-REVERSED PARTNERS

If the aim is only to produce a printed title page, the definition of the title element could simply be

```
<!-- ELEMENTS MIN CONTENT (EXCEPTIONS) -->
<!ELEMENT title - - (#PCDATA) >
```

and keyed

```
<title>An Experimental Test of Exchange Degeneracy in the Reactions
 $\pi^+p \rightarrow K^+\Sigma^+$ ,  $\pi^+p \rightarrow K^+\Sigma^+(1385)$  and their Line-Reversed Partners
</title>
```

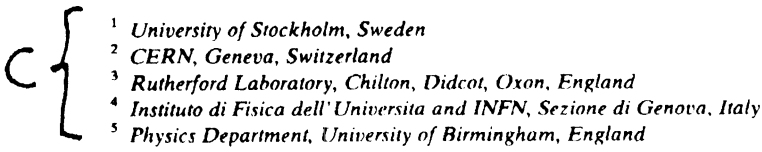
If the aim is also to make a database of articles or a secondary publication of an index to a series of articles, there may be a need to identify the reactions studied, and the element definitions might be

```
<!-- ELEMENTS MIN CONTENT (EXCEPTIONS) -->
<!ELEMENT title - - ((#PCDATA|reaction)+) >
<!ELEMENT reaction - - (#PCDATA) >
```

and keyed

```
<title>An Experimental Test of Exchange Degeneracy in the Reactions
<reaction> $\pi^+p \rightarrow K^+\Sigma^+$  </reaction>, <reaction> $\pi^+p \rightarrow K^+\Sigma^+(1385)$  </reaction>
and their Line-Reversed Partners
</title>
```

4.3.2 Instlist element



Similarly the list of institutes may require different degrees of structure depending on the application. Two extremes are

<!--	ELEMENTS	MIN	CONTENT (EXCEPTIONS)	-->
<!ELEMENT	instlist	- -	(inst+) -- list of institutes	-->
<!ELEMENT	inst	- 0	(#PCDATA) -- institute	-->
<!--	ELEMENTS	NAME	VALUE	DEFAULT
<!AT&LIST	inst	id	ID	#IMPLIED

which is keyed

```
<instlist>
<inst id=usto>University of Stockholm, Sweden.
...
<inst id=ubir>Physics Department, University of Birmingham, England.
</instlist>
```

and

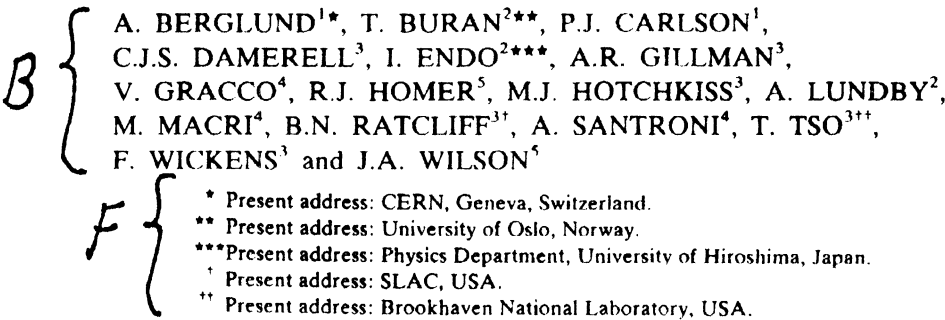
<!--	ELEMENTS	MIN	CONTENT (EXCEPTIONS)	-->
<!ELEMENT	instlist	- -	(inst+) -- list of institutes	-->
<!ELEMENT	inst	- 0	(dept*, instname*, instqual*, city*, county*, country*)	>
<!ELEMENT	dept	- 0	(#PCDATA) -- department	-->
<!ELEMENT	instname	- 0	(#PCDATA) -- name of institute	-->
<!ELEMENT	instqual	- 0	(#PCDATA) -- qualifier	-->
<!ELEMENT	city	- 0	(#PCDATA) -- city	-->
<!ELEMENT	county	- 0	(#PCDATA) -- county	-->
<!ELEMENT	country	- 0	(#PCDATA) -- country	-->
<!--	ELEMENTS	NAME	VALUE	DEFAULT
<!AT&LIST	inst	id	ID	#IMPLIED

which is keyed

```
<instlist>
<inst id=usto><instname>University of Stockholm<country>Sweden
<inst id=cern><instname>CERN<city>Geneva<country>Switzerland
<inst id=iral><instname>Rutherford Laboratory<city>Chilton
<city>Didcot<country>Oxon<country>England
<inst id=ugen><instname>Istituto di Fisica dell' Universita and INFN
<instqual>Sezione di Genova<country>Italy
<inst id=ubir><dept>Physics Department
<instname>University of Birmingham<country>England
</instlist>
```

NOTE The data entry for a list of this complexity is clearly well suited for being menu driven, with fields for the various elements of the institute information.

4.3.3 Authlist element



The list of authors contains, in addition to the names, references, to institutes and in many cases also to explanatory notes. Since each note is almost always associated with one author, this element is considered part of the authors' list, but placed at the bottom of the page by the text formatter. The authors' names, with a view to database applications, are divided into initials and surname. Thus the definition is

<!--	ELEMENTS	MIN	CONTENT (EXCEPTIONS)		-->
<!ELEMENT	authlist	- -	((author anote)+)		>
<!ELEMENT	author	- 0	(initials, surname)		>
<!ELEMENT	initials	0 0	(#PCDATA)		>
<!ELEMENT	surname	- 0	(#PCDATA)		>
<!ELEMENT	anote	- 0	(#PCDATA)		>
<!--	ELEMENTS	NAME	VALUE	DEFAULT	-->
<!ATTLIST	author	refid	IDREFS	#REQUIRED	>
<!ATTLIST	anote	id	ID	#IMPLIED	>

and is keyed

```
<authlist>
<author refid="usto pacern">A<surname>Berglund
<anote id=pacern>Present address: CERN, Geneva, Switzerland.
...
<author refid=lral>F<surname>Wickens
<author refid=ubir>JA<surname>Wilson
...
<authlist>
```

The explanatory notes of different kind, e.g. "Present address", and "Paid in part by", could be considered as different elements with the definition

<!--	ELEMENTS	MIN	CONTENT (EXCEPTIONS)		-->
<!ELEMENT	panote	- 0	(#PCDATA)		>
<!ELEMENT	ppnote	- 0	(#PCDATA)		>
<!--	ELEMENTS		NAME	VALUE	DEFAULT
<!ATTLIST	(panote ppnote)		id	ID	#IMPLIED

and keyed

```
<panote id=pacern>CERN, Geneva, Switzerland
<ppnote id=ppslac>SLAC, USA
```

Alternatively the type could be indicated by an attribute and the element defined

<!--	ELEMENTS	MIN	CONTENT (EXCEPTIONS)		-->
<!ELEMENT	anote	- 0	#PCDATA		>
<!--	ELEMENTS	NAME	VALUE	DEFAULT	-->
<!ATTLIST	anote	id	ID	#IMPLIED	>
		type	(pa pp none)	none	>

and keyed

```
<anote id=pacern type=pa>CERN, Geneva, Switzerland
<anote id=ppslac type=pp>SLAC, USA
```

4.3.4 Date element

The date in this journal article can be defined as

<!--	ELEMENTS	MIN	CONTENT (EXCEPTIONS)	-->
<!ELEMENT	date	- 0	(#PCDATA)	>

and keyed

```
<date>19 October 1979
```

In other cases, different dates of different types, may be necessary, for example

<!--	ELEMENTS	MIN	CONTENT (EXCEPTIONS)		-->
<!ELEMENT	titlem	- -	(title, authlist, instlist, date+,abstract)		>
<!ELEMENT	date	- 0	(#PCDATA)		>
<!--	ELEMENTS	NAME	VALUE	DEFAULT	-->
<!ATTLIST	date	type	(receive publish revise)	publish	>

and keyed

```
<date type=receive>19 October 1979
```

```

<date type=publish>20 January 1980
<date type=revise >15 March 1981
<date type=publish>20 May 1981
<date type=revise >30 November 1982
<date type=publish>3 February 1983

```

or separate elements may be defined for the different types of dates

<!--	ELEMENTS	MIN	CONTENT (EXCEPTIONS)	-->
<!ELEMENT	title	- -	(title, authlist, instlist, (reccdate pubdate revdate)*, abstract)	>
<!ELEMENT	reccdate	- 0	(#PCDATA)	>
<!ELEMENT	pubdate	- 0	(#PCDATA)	>
<!ELEMENT	revdate	- 0	(#PCDATA)	>

and keyed

```

<reccdate>19 October 1979
<pubdate>20 January 1980
<revdate>15 March 1981
<pubdate>20 May 1981
<revdate>30 November 1982
<pubdate>3 February 1983

```

In some cases the date can be defined as a simple character string. In other applications there may be a requirement to identify the day, month, and year as well as the calendar. For example

<!--	ELEMENTS	MIN	CONTENT (EXCEPTIONS)	-->
<!ELEMENT	date	- 0	(day, month, year)	>
<!ELEMENT	day	- 0	(#PCDATA)	>
<!ELEMENT	month	- 0	(#PCDATA)	>
<!ELEMENT	year	- 0	(#PCDATA)	>

<!--	ELEMENTS	NAME	VALUE	DEFAULT	-->
<!ATTLIST	date		calendar (gregor julian hegira mundi)	gregor	>

which is keyed

```

<date calendar=gregor><day>4<month>May<year>1987
<date calendar=julian><day>12<month>February<year>1492
<date calendar=mundi><day>23<month>Adar<year>5744
<date calendar=hegira><day>10<month>Ramadan<year>1370

```

In some applications it may be desirable that the application software (or text formatter) generates the current date, rather than the date being always keyboarded.

4.3.5 Abstract element

The abstract contains, in this case, paragraphs of text that consist only of characters. In a more general case paragraphs contain many more subelements that are described in clause 5.

<!--	ELEMENTS	MIN	CONTENT (EXCEPTIONS)	-->
<!ELEMENT	abstract	- -	(p+)	>
<!ELEMENT	p	- 0	(#PCDATA)	>

and keyed

```

<abstract><p>A description is given . . . ranges studied.
</abstract>

```

4.3.6 Article element

The title matter under consideration is of a journal article. For the discussion of the publication and copyright information there is a need to look at the element one level higher than the titlem element. This element, article, is composed of publication information, copyright, and a keyword list, followed by titlem and the body of the article. The content of the publication information is automatically generated by the application software (or text formatter) based on information one level higher than the article element. Similarly the copyright notice is also automatically generated. Thus

<!--	ELEMENTS	MIN	CONTENT (EXCEPTIONS)	-->
<!ELEMENT	article	- -	(publinfo, copyrigt, keywlist, titlem, body)	>
<!ELEMENT	publinfo	- 0	EMPTY -- generated	-->
<!ELEMENT	copyrigt	- 0	EMPTY -- generated	-->
<!ELEMENT	keywlist	- -	(keyw+)	>
<!ELEMENT	keyw	- 0	(#PCDATA)	>

4.3.7 Complete definition of the title matter

A full definition of the title matter could thus be

<!--	ELEMENTS	MIN	CONTENT (EXCEPTIONS)		-->
<!ELEMENT	article	- -	(publinfo, copyrigt, keywlist, titlem, body)		>
<!ELEMENT	publinfo	- 0	EMPTY	-- generated	-->
<!ELEMENT	copyrigt	- 0	EMPTY	-- generated	-->
<!ELEMENT	keywlist	- -	(keyw+)		>
<!ELEMENT	keyw	- 0	(#PCDATA)		>
<!ELEMENT	titlem	- -	(title, authlist, instlist, date, abstract)		>
<!ELEMENT	title	- -	((#PCDATA reaction)+)		>
<!ELEMENT	reaction	- -	(#PCDATA)		>
<!ELEMENT	authlist	- -	((author anote)+)		>
<!ELEMENT	author	- 0	(initials, surname)		>
<!ELEMENT	initials	0 0	(#PCDATA)		>
<!ELEMENT	surname	- 0	(#PCDATA)		>
<!ELEMENT	anote	- 0	(#PCDATA)		>
<!ELEMENT	instlist	- -	(inst+) -- list of institutes		-->
<!ELEMENT	inst	- 0	(#PCDATA)	-- institute	-->
<!ELEMENT	date	- 0	(#PCDATA)		>
<!ELEMENT	abstract	- -	(p+)		>
<!ELEMENT	p	- 0	(#PCDATA)	-- paragraph	-->
<!--	ELEMENTS	NAME	VALUE	DEFAULT	-->
<!ATTLIST	inst	id	ID	#IMPLIED	>
<!ATTLIST	author	refid	IDREFS	#REQUIRED	>
<!ATTLIST	anote	id	ID	#IMPLIED	>

and keyed

```

<article>
<publinfo>
<copyrigt>
<keywlist>
<keyw>exchange degeneracy
<keyw>line reversal
<keyw>hypercharge exchange
</keywlist>
<titlem>
<title>An Experimental Test of Exchange Degeneracy in the Reactions
<reaction> $\pi^+ p \rightarrow K^+ \Sigma^+$ </reaction>, <reaction> $\pi^+ p \rightarrow K^+ \Sigma^+$  (1385)</reaction> and their Line-reversed Partners
</title>
<authlist>
<author refid="usto pacern">A<surname>Berglund
...
<author refid=lral>F<surname>wickens
<author refid=ubir>JA<surname>Wilson
<anote id=pacern>Present address: CERN, Geneva, Switzerland.
...

```

```
</authlist>
<instlist>
<inst id=usto>University of Stockholm, Sweden.
...
<inst id=lral>Rutherford Laboratory, Chilton, Didcot, Oxon, England.
<inst id=ubir>Physics Department, University of Birmingham, England.
</instlist>
<date>19 October 1979
<abstract><p>A description is given . . . ranges studied.
</abstract>
</title>
<body>
...
```

4.4 Example of the analysis of tabular material

Consider the following table:

Grandeur de filetage		Charge d'essai, % Tolérances du filetage		
au-dessus de	jusqu'à	6H	7H	6G
—	M2,5	100	—	95,5
M2,5	M7	100	95,5	97
M7	M16	100	96	97,5
M16	M39	100	98	98,7

One approach would be to consider the table as a “tree structure” being composed of two headed columns. The first column further contains two headed columns, each with four rows, and the second column three headed columns, each with four rows. The element definition would be

```
<!--      ELEMENTS  MIN  CONTENT (EXCEPTIONS)      -->
<!ELEMENT table      -  -  (col+)                  >
<!ELEMENT col        -  -  (head?, (row+ | col+))    >
<!ELEMENT row        -  0  (#PCDATA)                >
<!ELEMENT head       -  0  (#PCDATA, subhead?)      >
<!ELEMENT subhead    -  0  (#PCDATA)                >
```

and the example table keyed

```
<table>
<col><head>Grandeur de filetage
<col><head>au-dessus de <row> M2,5 <row> M7 <row> M16 </col>
<col><head>jusqu'à <row> M2,5 <row> M7 <row> M16 <row> M39 </col>
</col>
<col><head>Charge d'essai, % <subhead> Tolérances du filetage
<col><head> 6H <row> 100 <row> 100 <row> 100 <row> 100 </col>
<col><head> 7H <row> - <row> 95,5 <row> 96 <row> 98 </col>
<col><head> 6G <row> 95,5 <row> 97 <row> 97,5 <row> 98,7 </col>
</col>
</table>
```

Another approach would be to consider the “tree structure” as being composed of a row of headings, each of two levels, followed by rows of table entries. The element definition would be

```
<!--      ELEMENTS  MIN  CONTENT (EXCEPTIONS)      -->
<!ELEMENT table      -  -  (head1+, row+)          >
<!ELEMENT head1      -  0  (#PCDATA, subhead?, head2*) >
<!ELEMENT head2      -  0  (#PCDATA, subhead?)      >
<!ELEMENT subhead    -  0  (#PCDATA)                >
<!ELEMENT row        -  0  (col+)                  >
<!ELEMENT col        -  0  (#PCDATA)                >
```

and the example table keyed

```
<table>
<head1>Grandeur de filetage
<head2>au-dessus de <head2>jusqu'à
```

```

<head1>Charge d'essai, % <subhead> Tolérances du filetage
<head2> 6H <head2> 7H <head2> 6G
<row><col> - <col> M2,5 <col> 100 <col> - <col> 95,5
<row><col> M2,5 <col> M7 <col> 100 <col> 95,5 <col> 97
<row><col> M7 <col> M16 <col> 100 <col> 96 <col> 97,5
<row><col> M16 <col> M39 <col> 100 <col> 98 <col> 98,7
</table>

```

NOTE An element definition for tables based on this approach is given in clause 9.

Another approach would be to describe it “geometrically” as a series of rows, each containing two or five columns, the widths indicated by attributes. Rules are explicitly indicated by tags and attributes. The first two rows are “headers” of two levels. The element definition would be

<!--	ELEMENTS	MIN	CONTENT (EXCEPTIONS)	-->
<!ELEMENT	table	- -	(head1, head2?, row+)	>
<!ELEMENT	(head1 head2)	- 0	(colh+, hbar?)	>
<!ELEMENT	colh	- 0	(#PCDATA, subhead?)	>
<!ELEMENT	subhead	- 0	(#PCDATA)	>
<!ELEMENT	hbar	- 0	EMPTY	>
<!ELEMENT	row	- 0	(col+, hbar)	>
<!ELEMENT	col	- 0	(#PCDATA)	>
<!--	ELEMENTS	NAME	VALUE	DEFAULT
<!ATTLIST	table	id	ID	#IMPLIED
		width	CDATA	#REQUIRED
		frame	(framed none)	none
<!ATTLIST	(head1 head2 row) widths		CDATA	#CURRENT
		vbars	(on off)	off

and the example table keyed

```

<table width="87mm" framed>
<head1 widths="36mm 51mm" vbars=on>
<colh>Grandeur de filetage
<colh>Charge d'essai, %<subhead>Tolérances du filetage
<head2 widths="18mm 18mm 17mm 17mm 17mm" vbars=on>
<colh>au-dessus de <colh>jusqu'à<colh>6H<colh>7H<colh>6G
<hbar>
<row widths="18mm 18mm 17mm 17mm 17mm" vbars=on>
<col - <col>M2,5 <col>100 <col> - <col> 95,5
<hbar>
<row vbars=on>
<col> M2,5 <col>M7 <col>100 <col> 95,5 <col> 97
<hbar>
<row vbars=on>
<col> M7 <col>M16 <col>100 <col> 96 <col> 97,5
<hbar>
<row vbars=on>
<col> M16 <col>M39 <col>100 <col> 98 <col> 98,7
</table>

```

NOTE Layout information would normally rarely be indicated in the tags. Tables, however, is one example where layout parameters cannot always be determined by the application. Using parsers not supporting the LINK feature these have to be specified using attributes, whereas LINK would normally be used if a parser supporting LINK is available.

4.5 Example of the analysis of a simple mathematical formula

For the description of mathematical formulae a number of “languages” have been proposed and implemented. The basic information that any such description needs to convey is the identification of the components of the formula and their relationships (that is structure). Mathematics is thus well suited to be described by an SGML application. Consider, for example

The components that need identification are

- A: the operator. Call this element INTEGRAL.
- B: and C: the lower and upper limit. Call these elements FROM and TO.
- D: the integrand. Call this element OF.

The element definition of the integral can thus be

<!--	ELEMENTS	MIN	CONTENT (EXCEPTIONS)	-->
<!ELEMENT	integral	- -	(from, to, of)	>

The integrand is, in this example, a fraction composed of a numerator and a denominator. The fraction elements can be defined as

<!ENTITY % f_oper	"integral frac"	-- partial list of operations	-->	
<!--	ELEMENTS	MIN	CONTENT (EXCEPTIONS)	-->
<!ELEMENT	frac	- -	(number, over)	>
<!ELEMENT	numer	0 0	(#PCDATA %f_oper)+	>
<!ELEMENT	over	- 0	(#PCDATA %f_oper)+	>

and keyed

`<frac> a <over> x </frac>`

In the example, the numerator and denominator are both just #PCDATA, but in a more general case they can be #PCDATA and/or any other operation, such as another fraction. Thus the content has been defined as (#PCDATA| %f_oper;)+. Similarly, the upper and lower limits can take either #PCDATA and/or other operations as content.

The complete definition of the elements for the integral and the fraction is thus

<!ENTITY % f_oper	"integral frac"	-- partial list of operations	-->	
<!--	ELEMENTS	MIN	CONTENT (EXCEPTIONS)	-->
<!ELEMENT	integral	- -	(from, to, of)	>
<!ELEMENT	(from to of)	- 0	(#PCDATA %f_oper)+	>
<!ELEMENT	frac	- -	(number, over)	>
<!ELEMENT	numer	0 0	(#PCDATA %f_oper)+	>
<!ELEMENT	over	- 0	(#PCDATA %f_oper)+	>

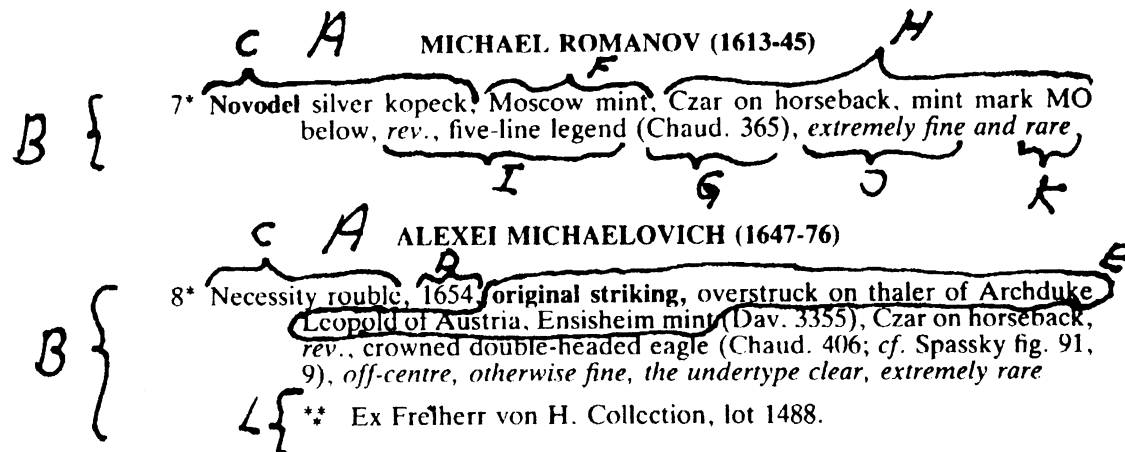
and this example keyed

`<integral><from> 0 <to> <frac> 1 <over> a </frac> <of>
<frac> a <over> x </frac> dx </integral>`

NOTE A complete element definition for mathematics is given in clause 8.

4.6 Example of the analysis of an auction catalogue

Consider the following entries from an auction catalogue:



The following main elements can be identified:

- A: header (in this case the name of a ruler), that can be followed by explanatory text. Call these elements H2 and P.

NOTE H1 is used to separate the catalogue into countries.

- B: lot description. Call this element LOT. Lots that are illustrated are shown with an asterisk after the number, so an attribute to specify this is needed. Call this attribute ILL.

Each LOT element is divided into several components:

- C: denomination. Call this element DENOM.
- D: date. Call this element DATE.
- E: general comment. Call this element GENCOM.
- F: mint. Call this element MINT.
- G: reference. Call this element REF.
- H: description of obverse. Call this element OBV.
- I: description of reverse. Call this element REV.
- J: condition. Call this element GRADE.
- K: rarity. Call this element RARITY.
- L: provenance. Call this element PROVEN.

For some entries, of less valuable items, the description is shorter. For this reason a number of the elements are optional.

The elements for these catalogue entries can be defined as

<!ENTITY	% p.em.ph	"hp1 hp2 hp3"	-- emphasized phrases	-->
<!--	ELEMENTS	MIN	CONTENT (EXCEPTIONS)	-->
<!ELEMENT	h2	- 0	(h2t, p*, lot+)	>
<!ELEMENT	h2t	0 0	(#PCDATA)	>
<!ELEMENT	p	0 0	(#PCDATA %p.em.ph)+	>
<!ELEMENT	lot	- 0	(denom, date?, gencom?, mint?, (obv, rev)?, ref*, grade, rarity?, proven?)	>
<!ELEMENT	denom	- 0	(#PCDATA %p.em.ph)*	>
<!ELEMENT	date	- 0	(#PCDATA)	>
<!ELEMENT	gencom	- 0	(#PCDATA ref %p.em.ph)*	>
<!ELEMENT	(obv rev)	- 0	(#PCDATA %p.em.ph)*	>
<!ELEMENT	ref	- 0	(#PCDATA %p.em.ph)*	>
<!ELEMENT	mint	- 0	(#PCDATA %p.em.ph)*	>
<!ELEMENT	grade	- 0	(#PCDATA %p.em.ph)*	>
<!ELEMENT	rarity	- 0	(#PCDATA %p.em.ph)*	>

```

<!ELEMENT proven - 0 (p+) >
<!ELEMENT (hp1|hp2|hp3) - 0 (#PCDATA) >
<!-- ELEMENTS NAME VALUE DEFAULT -->
<!ATTLIST lot ill (yes|no) no >

```

and these two entries keyed

```

<h2>Michael Romanov (1613-45)
<lot ill=yes><denom><hp2>Novodel</hp2> silver kopek <mint>Moscow
<obv>Czar on horseback, mint mark MO below<rev>five-line legend
<ref>Chaud. 365 <grade>extremely fine<rarity>rare
<h2>Alexei. Michaelovich (1647-76)
<lot ill=yes><denom>Necessity Rouble<date>1654<gencom><hp2>original striking,</hp2> overstruck on thaler of
Archduke Leopold of Austria, Ensishheim mint<ref>Dav 3355.
<obv>Czar on horseback<rev>crowned double-headed eagle
<ref>Chaud. 406; <hp1>cf</hp1> Spassky Figure 91, Figure 9
<grade>off centre, otherwise fine, the undertype clear
<rarity>extremely rare<proven>Freiherr von H. Collection, lot 1488.

```

4.7 The DTD with relation to the size of a document

In general, the body elements described in clause 5 are common to most document classes within an organization. Some elements may not be frequently used in short documents; figures and tables may occur rarely in letters. From the point of view of consistency, and thus easier learning, there appears to be no compelling reason to exclude such elements from the document type definition of shorter documents. On the other hand in some systems there could be a performance implication as the complexity of the application program or text formatter could be affected. In any event, the layout — presentation — style normally does vary with document length and document type.

The front and back matter elements, on the other hand, show, in general, many differences depending on document type and thus average length.

In some organizations there is a requirement for special elements, for example mathematical formulae. Such elements, however, are likely to be required in all document types regardless of document length. In other words, the element repertoire required is more likely to be a function of the content of the documents in an organization rather than a function of the average length of the documents of each type.

5 General Document

5.1 General notes

The aim of this clause is to provide an example of a practical document type definition for a “general” document that can be used as a basis for application designers when developing applications. It contains all the elements necessary for many applications, and has been constructed in such a way that other elements can be added if the application so requires.

5.2 Document structure

A general document element (**general**) consists of front matter (**frontm**) (optional), followed by the main body (**body**), followed by an appendix section (**appendix**) (optional), followed by back matter (**backm**). Both the start and end-tag of the element are required. Index entries (**ix**) and “floating elements” (**%i.float**) (figures and footnotes) are included elements, and these elements can thus be interspersed anywhere in the document.

```

general-----frontm-----titlep-----title-----tline-----%m.ph ==>
|
|-----docnum-----#PCDATA
|-----date-----#PCDATA
|-----abstract-----h1t-----%m.ph ==>
|               |
|               |-----%s.zz ==>
|               |-----h2 ==>
|
|-----author-----#PCDATA
|-----address-----aline-----%m.ph ==>
|-----%s.zz ==>
|
|-----%fm.d-----abstract-----h1t-----%m.ph ==>
|               |
|               |-----%s.zz ==>
|               |-----h2 ==>
|
|               |-----preface-----h1t-----%m.ph ==>
|               |
|               |-----%s.zz ==>
|               |-----h2 ==>
|
|-----h1-----h1t-----%m.ph ==>
|               |
|               |-----%s.zz ==>
|               |-----h2 ==>
|
|-----toc-----EMPTY
|-----figlist-----EMPTY
|
|-----body-----h0-----h0t-----%m.ph ==>
|               |
|               |-----%s.zz ==>
|               |-----h1-----h1t-----%m.ph ==>
|               |               |
|               |               |-----%s.zz ==>
|               |               |-----h2 ==>
|
|               |-----h1-----h1t-----%m.ph ==>
|               |
|               |-----%s.zz ==>
|               |-----h2 ==>
|
|-----appendix-----h1-----h1t-----%m.ph ==>
|               |
|               |-----%s.zz ==>
|               |-----h2 ==>
|
|-----backm-----%bm.d-----glossary-----h1t-----%m.ph ==>
|               |               |
|               |               |-----%s.zz ==>
|               |               |-----h2 ==>
|
|               |-----bibliog-----h1t-----%m.ph ==>
|               |               |
|               |               |-----%s.zz ==>
|               |               |-----h2 ==>
|
|               |-----h1-----h1t-----%m.ph ==>
|               |               |
|               |               |-----%s.zz ==>
|               |               |-----h2 ==>
|
|-----index-----EMPTY

+ix
+%.i.float

```

Figure 3 — Overall structure of the basic document

A front matter element (**frontm**) consists of a title page (**titlep**), followed by an abstract (**abstract**), or a preface (**preface**), or headed section 1 elements (**h1**) (optional and repeatable), followed by a table of contents (**toc**) (optional), followed by a list of figures (**figlist**) (optional). The end-tag of the front matter can be omitted.

The title page element (**titlep**) consists of a title (**title**), and a document number (**docnum**) (optional), and a date (**date**) (optional), and an abstract (**abstract**) (optional), and author (**author**), or address (**address**), or “section subelements” (**%s.zz**) (optional and repeatable). All these may be entered in any order. The end-tag of the title page can be omitted.

A title element (**title**) consists of one or more title lines (**tline**), each in its turn consisting of “phrase models” (**%m.ph**). The end-tag of the title element can be omitted. The end-tag of the title lines can be omitted (and the start-tag of the first title line).

A document number (**docnum**), or a date (**date**), or an author element (**author**) consists of character data (**#PCDATA**). The end-tag of these elements can be omitted.

An address element (**address**) consists of one or more address lines (**aline**), each in its turn consisting of “phrase models” (**%m.ph**). No tag of the address element can be omitted. The end-tag of the address lines can be omitted (and the start-tag on the first address line).

An abstract (**abstract**), or a preface element (**preface**) consists of a title (**h1t**), followed by “section subelements” (**%s.zz**) (optional and repeatable), followed by headed section 2 elements (**h2**) (optional and repeatable). The end-tag of the abstract or preface element can be omitted.

A table of contents (**toc**), or a list of figures element (**figlist**) has no content; the content is generated by the application software based on information contained in other elements. The end-tag of these elements can be omitted.

A body element (**body**) consists of either one or more headed section 0 elements (**h0**), or one or more headed section 1 elements (**h1**). The end-tag of the body element can be omitted.

A headed section 0 element (**h0**) consists of a title (**h0t**), followed by “section subelements” (**%s.zz**) (optional and repeatable), followed by one or more headed section 1 elements (**h1**).

A headed section 1 element (**h1**) consists of a title (**h1t**), followed by “section subelements” (**%s.zz**) (optional and repeatable), followed by headed section 2 elements (**h2**) (optional and repeatable). The end-tag can be omitted.

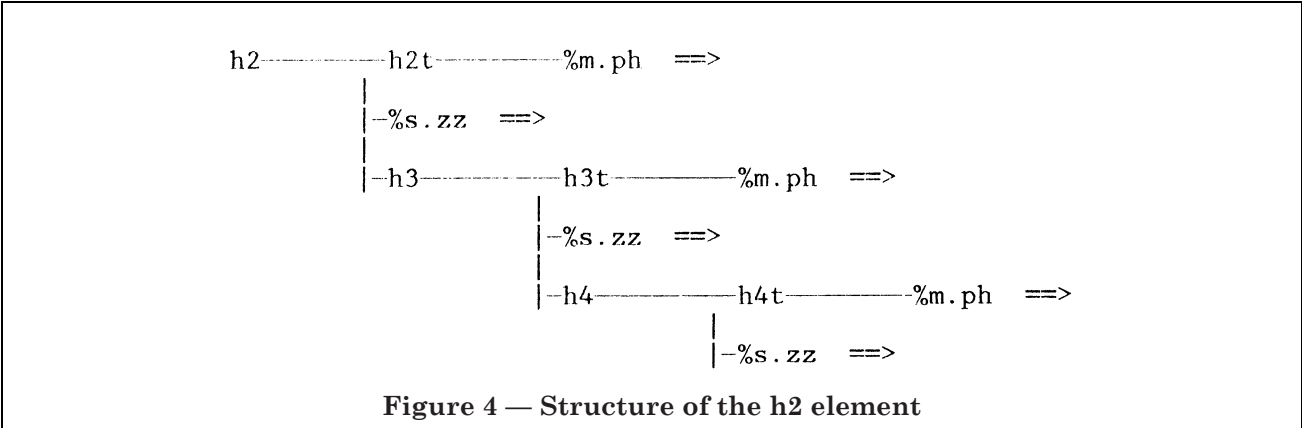


Figure 4 — Structure of the h2 element

A headed section 2 element (**h2**) consists of a title (**h2t**), followed by “section subelements” (**%s.zz**) (optional and repeatable), followed by headed section 3 elements (**h3**) (optional and repeatable). The end-tag can be omitted.

A headed section 3 element (**h3**) consists of a title (**h3t**), followed by “section subelements” (**%s.zz**) (optional and repeatable), followed by headed section 4 elements (**h4**) (optional and repeatable). The end-tag can be omitted.

A headed section 4 element (**h4**) consists of a title (**h4t**), followed by “section subelements” (**%s.zz**) (optional and repeatable). The end-tag can be omitted.

The title elements of headed sections (**h0t – h4t**) consist of “phrase models” (**%m.ph**). Both the start and end-tags can be omitted.

An appendix element (**appendix**) consists of one or more headed section 1 elements (**h1**).

A back matter element (**backm**) consists of a glossary (**glossary**), or a bibliography (**bibliog**), or a headed section 1 element (**h1**) (optional and repeatable), followed by an index (**index**) (optional). The end-tag can be omitted.

A glossary (**glossary**), or a bibliography element (**bibliog**) consists of a title (**h1t**), followed by “section subelements” (**%s.zz**) (optional and repeatable), followed by headed section 2 elements (**h2**) (optional and repeatable). The end-tag of these elements can be omitted.

An index element (**index**) has no content; the content is generated by the application software based on information contained in other elements. The end-tag can be omitted.

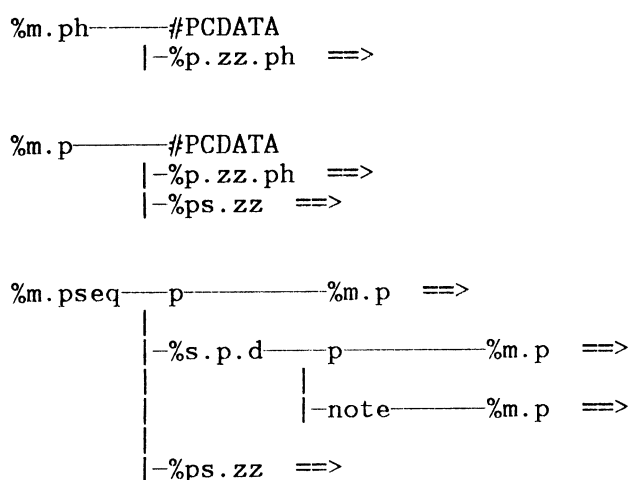


Figure 5 — Structure of the phrase (%m.ph), paragraph (%m.p), and paragraph sequence (%m.pseq) models

A “phrase model” (**%m.ph**) is defined as character data (**#PCDATA**) or “phrases” (**%p.zz.ph**) (optional and repeatable).

A “paragraph model” (**%m.p**) is defined as character data (**#PCDATA**), or “phrases” (**%p.zz.ph**), or “paragraph/section subelements” (**%ps.zz**) (optional and repeatable).

A “paragraph sequence” (**%m.pseq**) is defined as a paragraph (**p**), followed by a “simple paragraph” (**%s.p.d**) or “paragraph/section subelements” (**%ps.zz**) (optional and repeatable).

A “simple paragraph” (**%s.p.d**) is defined as a paragraph (**p**) or a note (**note**).

A paragraph (**p**) or a note element (**note**) consists of “paragraph models” (**%m.p**). Both the start and end-tag of these elements can be omitted.

NOTE The start-tag can be omitted only under certain, very restricted, conditions.

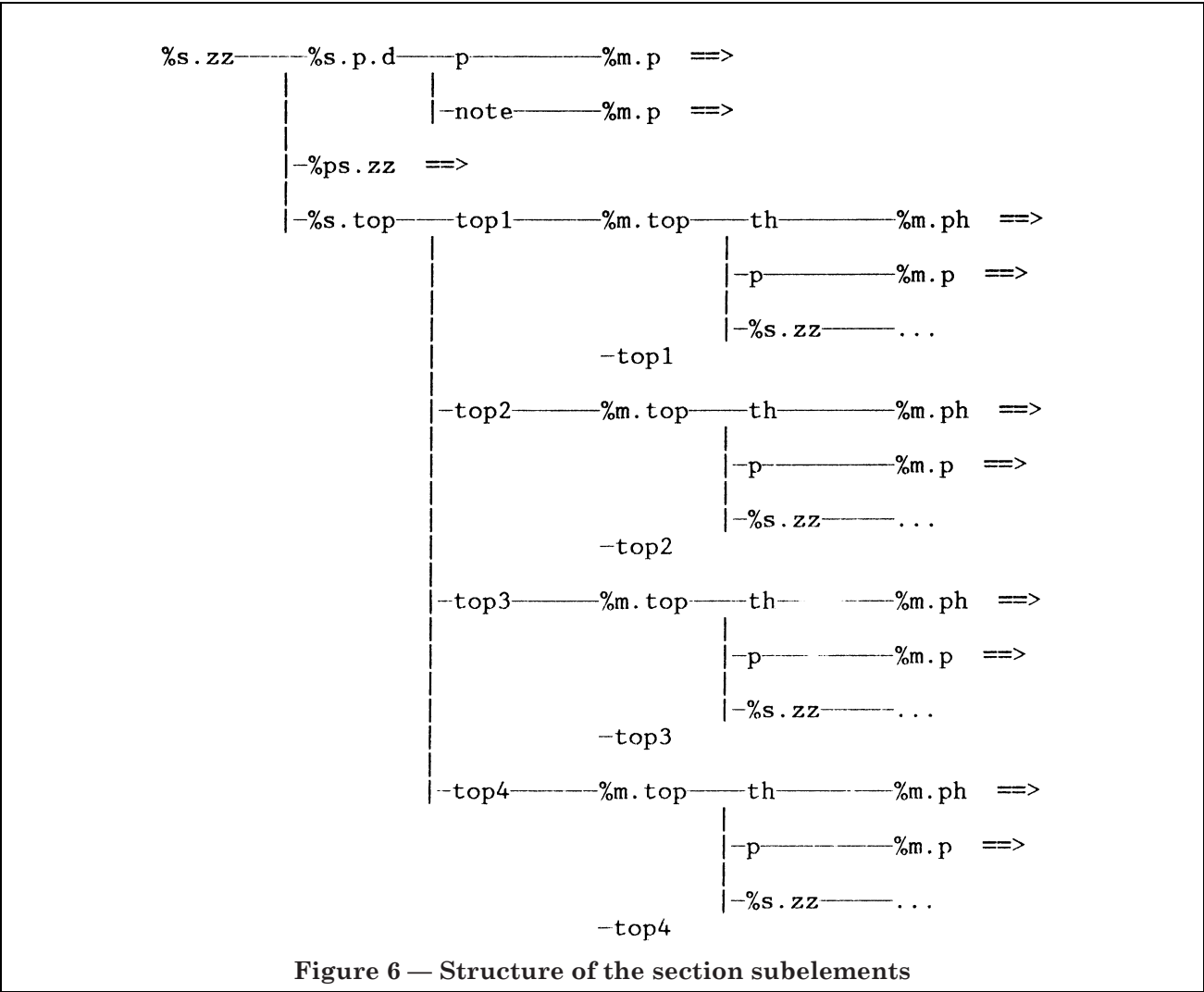


Figure 6 — Structure of the section subelements

A “section subelement” (**%s.zz**) is defined as a “simple paragraph” (**%s.p.d**), or a “paragraph/section subelement” (**%ps.zz**), or a “topic” (**%s.top**).

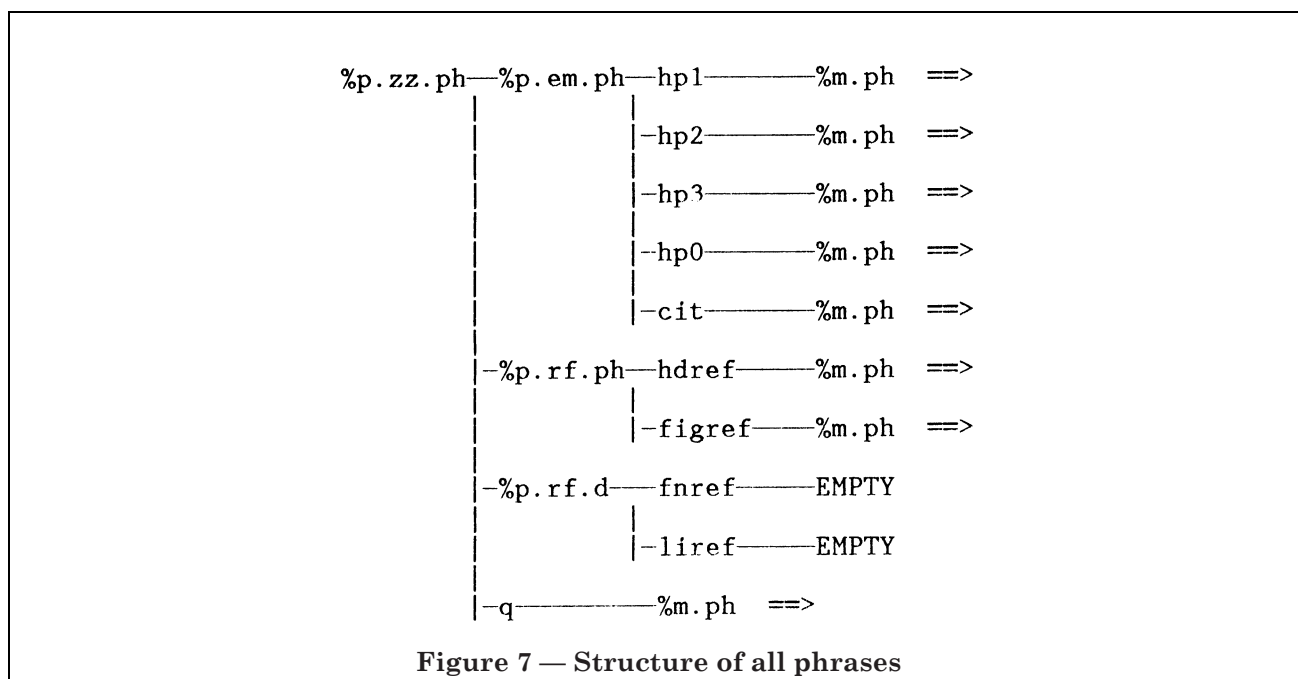
A topic of level one (**top1**) consists of a topic heading (**th**) (optional), followed by a paragraph (**p**), followed by “section subelements” (**%s.zz**) (optional and repeatable). Topics of level one are excluded elements; thus they cannot occur further at any level of this topic level. The end-tag can be omitted.

A topic heading consists of a “phrase model” (**%m.ph**). The end-tag can be omitted.

A topic of level two (**top2**) consists of a topic heading (**th**) (optional), followed by a paragraph (**p**), followed by “section subelements” (**%s.zz**) (optional and repeatable). Topics of level two are excluded elements; thus they cannot occur further at any level of this topic level. The end-tag can be omitted.

A topic of level three (**top3**) consists of a topic heading (**th**) (optional), followed by a paragraph (**p**), followed by “section subelements” (**%s.zz**) (optional and repeatable). Topics of level three are excluded elements; thus they cannot occur further at any level of this topic level. The end-tag can be omitted.

A topic of level four (**top4**) consists of a topic heading (**th**) (optional), followed by a paragraph (**p**), followed by “section subelements” (**%s.zz**) (optional and repeatable). Topics of level four are excluded elements; thus they cannot occur further at any level of this topic level. The end-tag can be omitted.



A “phrase” (**%p.z.z.ph**) is defined as an “emphasized phrase” (**%p.em.ph**), or a “reference phrase” (**%p.rf.ph**), or a “reference (empty)” (**%p.rf.d**), or a “quotation” (**q**).

An “emphasized phrase” (**%p.em.ph**) is defined as a highlighted phrase of level zero to three (**hp0 – hp3**), or a citation (**cit**).

A highlighted phrase of level zero to three (**hp0 – hp3**), or a citation (**cit**) consists of “phrase models” (**%m.ph**). No markup minimization is permitted.

NOTE As a “phrase model” contains character data, or “phrases” the definition is recursive, which implies that highlighted phrases can be nested.

A “reference phrase” (**%p.rf.ph**) is defined as a header reference (**hdref**), or a figure reference (**figref**).

A header reference element (**hdref**), or a figure reference (**figref**) consists of “phrase models” (**%m.ph**). No markup minimization is permitted.

A “reference (empty)” (**%p.rf.d**) is defined as a footnote reference (**fnref**), or a list item reference (**liref**).

A footnote reference element (**fnref**), or list item reference (**liref**) has no content; the content is generated by the application software.

A (short) quotation element (**q**) consists of a “phrase model” (**%m.ph**). No markup minimization is permitted.

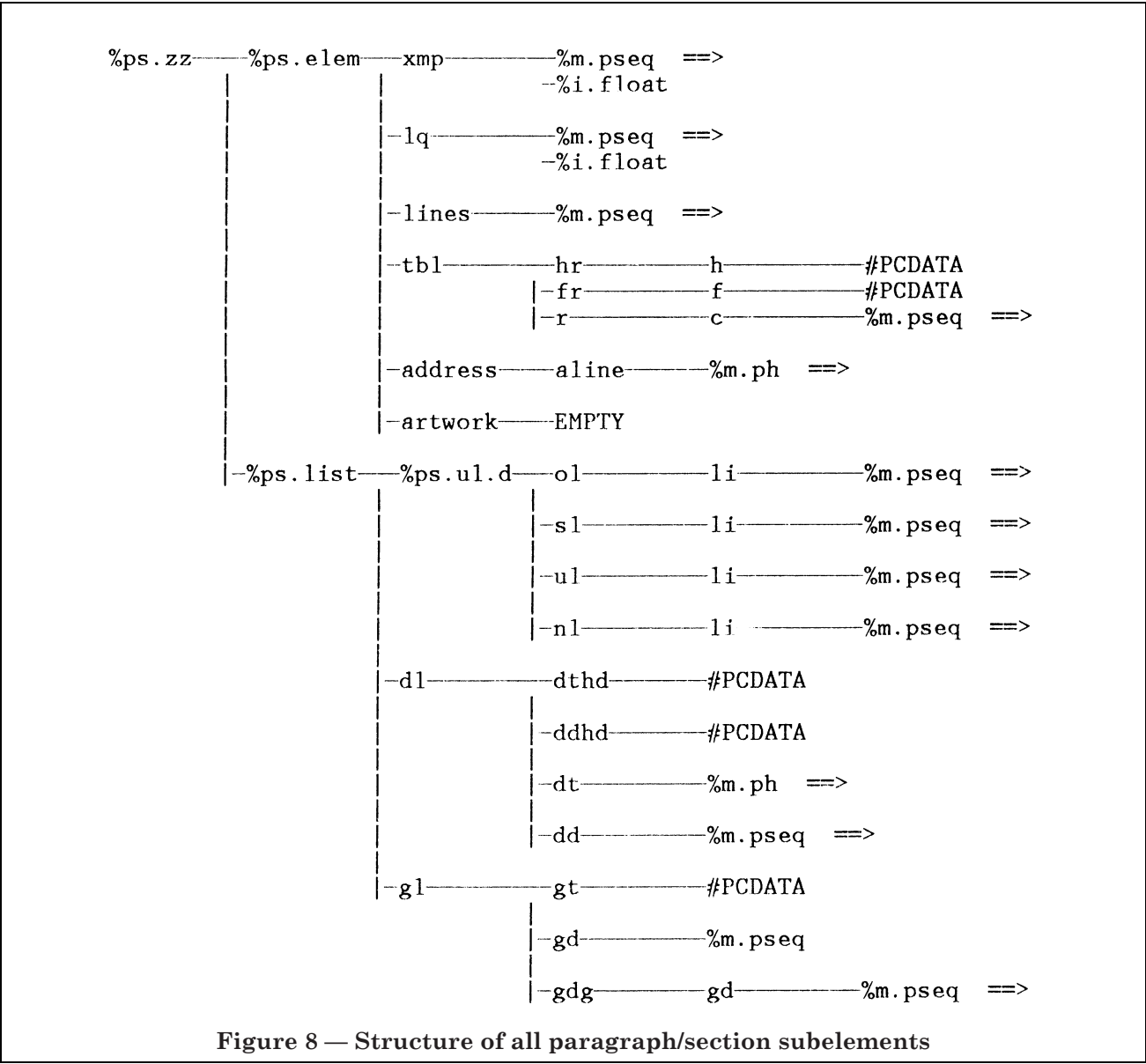


Figure 8 — Structure of all paragraph/section subelements

A “paragraph/section subelement” (**%ps.zz**) is defined as “other elements” (**%ps.elem**) or “lists” (**%ps.list**). “Other elements” (**%ps.elem**) is defined as example (**xmp**), or long quotation (**lq**), or lines (**lines**) or table (**tbl**), or address (**address**), or artwork (**artwork**).

An example (**xmp**), or a long quotation (**lq**) consists of a “paragraph sequence” (**%m.pseq**). “Floating elements” (**%i.float**) are excluded elements; thus they cannot occur further at any level in an example or a long quotation. No markup minimization is permitted.

A lines element (**lines**) consists of a “paragraph sequence” (**%m.pseq**). Both the start-tag and the end-tag can be omitted.

A table element (**tbl**) consists of a heading row (**hr**) (optional and repeatable), followed by a footing row (**fr**) (optional and repeatable), followed by one or more rows (**r**). No markup minimization is permitted.

A heading row (**hr**) consists of one or more heading row cells (**h**). The end-tag can be omitted.

A footing row (**fr**) consists of one or more footing row cells (**f**). The end-tag can be omitted.

A heading row cell (**h**) or footing row cell (**f**) consists of character data (**#PCDATA**). Both the start and end-tag can be omitted.

A row (**r**) consists of one or more cells (**c**). Both the start and end-tags can be omitted.

A cell (**c**) consists of a “paragraph sequence” (**%m.pseq**). Both the start and end-tags can be omitted.

An artwork element (**artwork**) has no content. Using attributes, space is reserved for pasted-in material.

“Lists” (**%ps.list**) are defined as “unit item lists” (**%ps.ul.d**), or definition list (**dl**), or glossary list (**gl**).

“Unit item lists” (**%ps.ul.d**) are defined as ordered list (**ol**), or simple list (**sl**), or unordered list (**ul**), or note list (**nl**). No markup minimization is permitted.

Ordered list (**ol**), or simple list (**sl**), or unordered list (**ul**), or note list elements (**nl**) consists of list items (**li**) (optional and repeatable).

List items (**li**) consist of “paragraph sequences” (**%m.pseq**). The end-tag can be omitted.

A definition list (**dl**) consists of optional headings for the term (**dthd**) and definition (**ddhd**), followed by optional and repeatable blocks of definition term (**dt**) (repeatable) and definition (**dd**). No markup minimization is permitted.

A definition heading (**dthd**, **ddhd**) consists of character data (**#PCDATA**). The end-tag can be omitted.

A definition term (**dt**) consists of a “phrase model” (**%m.ph**). The end-tag can be omitted.

A definition (**dd**) consists of a “paragraph sequence” (**%m.pseq**). The end-tag can be omitted.

A glossary list (**gl**) consists of an optional and repeatable block of a glossary term (**gt**), followed by a glossary definition (**gd**) or a glossary definition group (**gdg**). No markup minimization is permitted.

A glossary term (**gt**) consists of character data (**#PCDATA**). The end-tag can be omitted.

A glossary definition (**gd**) consists of a “paragraph sequence” (**%m.pseq**). The end-tag can be omitted.

A glossary definition group (**gdg**) consists of one or more glossary definitions (**gd**). The end-tag can be omitted.

“Floating elements” (**%i.float**) are defined as figures (**fig**) and footnotes (**fn**).

A figure (**fig**) consists of a figure body (**figbody**), followed by an optional block of figure caption (**figcaption**), followed by an optional figure description (**figdesc**). “Floating elements” are excluded elements; thus a figure or a footnote cannot occur inside a figure. No markup minimization is permitted.

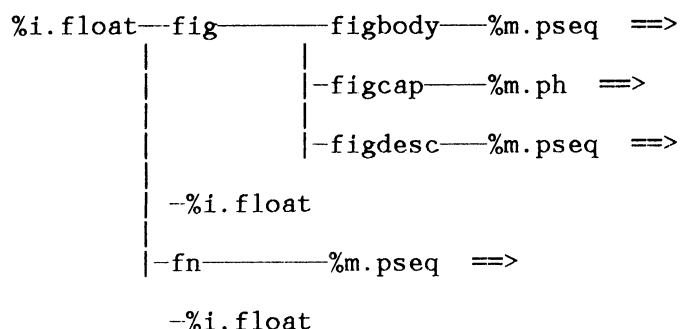


Figure 9 — Structure of all floating elements

A figure body (**figbody**) consists of a “paragraph sequence” (**%m.pseq**). Both the start and end-tags can be omitted.

A figure caption (**figcaption**) consists of a “phrase model” (**%m.ph**). The end-tag can be omitted.

A figure description (**figdesc**) consists of a “paragraph sequence” (**%m.pseq**). The end-tag can be omitted.

A footnote (**fn**) consists of a “paragraph sequence” (**%m.pseq**). “Floating elements” are excluded elements thus a figure or a footnote cannot occur inside a footnote. No markup minimization is permitted.

An index entry (**ix**) consists of character data (**#PCDATA**). The end-tag can be omitted.

5.3.2 Title page elements

title page `<titlep>`

`<titlep>` indicates the beginning of information to appear on the title page(s) of the document. The title page information may consist of, keyed in any order, a title, an optional document number, an optional date, an optional abstract, optional author information, and optional section subelements. `<titlep>` is followed immediately by another tag. Valid only in the front matter element.

title (of document) `<title>` `stitle =`

This tag precedes the title of the document. The short title attribute (`stitle =`) may be used to input a short version of the title to be used, typically, as a running head or foot. No end-tag is necessary. Valid only in the title page element. The content of the title element is one or more title lines (tline elements). The start and end-tag of these can be omitted.

```
<title stitle="Good SGML Practice">
The Handbook of
Good SGML Practice
```

document number `<docnum>`

If a document is to be given a number or code, it is preceded by `<docnum>`. No end-tag is necessary. Valid only in the title page element.

```
<docnum> ISO 8879
```

date `<date>`

This is the date associated with the publication of the document. Whereas in draft stages it could indicate the day and the month as well as the year, the final version could merely indicate the year. In draft stages the date could be system-supplied. No end-tag is necessary.

```
<date>1987-06-29
```

author `<author>`

The name of the author is preceded by `<author>`. The element may be repeated in the case of there being more than one author. No end-tag is necessary. Valid only in the title page element.

```
<author>A.Someone
<author>B.Else
```

address `<address>` `</address>`

The `<address>` tag indicates the start of an address. It is followed immediately by repeatable address line (aline) elements, where the end-tags can be omitted. When all the lines of the address have been keyed in, `</address>` is input to indicate the end of the address. The element may be repeated in the case of there being more than one address. Valid only in the title page element.

```
<address>
<aline>1, rue de Varembe
<aline>CH-1211 Genève 20
<aline>Suisse
</address>
```

5.3.3 Front matter elements

abstract `<abstract>` `id =` `stitle =`

The abstract or summary of a document is preceded by `<abstract>`. It is followed, after a required title, by a tag for one of the "section subelements". The value of the identifier attribute (`id =`) can be used when referring to the abstract and must be a unique reference in the document. The short title attribute (`stitle =`) may be used to input a short version of the heading to be used, typically, as a running head or foot. Valid only in the front matter.

```
<abstract>Abstract
<p>This report aims to show examples of good usage of SGML in
a variety of application areas.
```

preface `<preface>` id = stitle =

The `<preface>` tag is keyed in before a preface. The element may be repeated in the case of there being more than one preface. It is followed, after a required title, by a tag for one of the “section subelements”. The value of the identifier attribute (id =) can be used when referring to the preface heading and must be a unique reference in the document. The short title attribute (stitle =) may be used to input a short version of the heading to be used, typically, as a running head or foot. No end-tag is necessary. Valid only in the front matter.

```
<preface>Introduction
<p>ISO 8879 contains, without much explanation, an example of
a document type definition . . .
```

5.3.4 Back matter elements

glossary `<glossary>` id = stitle =

The glossary is preceded by `<glossary>`. It is followed, after a required title, by a tag for one of the “section subelements”. The value of the identifier attribute (id =) can be used when referring to the glossary and must be a unique reference in the document. The short title attribute (stitle =) may be used to input a short version of the heading to be used, typically, as a running head or foot. No end-tag is necessary. Valid only in the back matter.

```
<glossary>Glossary
<gl>
<gt>Beaujolais Nouveau<gd>A wine from the Beaujolais district
bottled and distributed very young.
</gl>
```

bibliography `<bibliog>` id = stitle =

The `<bibliog>` tag precedes any list of bibliographic details to be input. It is followed, after a required title, by a tag for one of the “section subelements”. The value of the identifier attribute (id =) can be used when referring to the bibliography and must be a unique reference in the document. The short title attribute (stitle =) may be used to input a short version of the heading to be used, typically, as a running head or foot. Valid only in the back matter.

5.3.5 Headed sections

headed section 0-4 `<h0>`, `<h1>`, `<h2>` id = stitle =
`<h3>`, `<h4>` id =

These tags are used to identify the main sections of the document. Level 0 could identify parts of a large document, level 1 chapter titles, the others being used for the subsections. The value of the identifier attribute (id =) can be used when referring to the heading (section) and must be a unique reference in the document. The short title attribute (stitle =) may be used to input a short version of the heading to be used, typically, as a running head or foot. No end-tag is necessary.

```
<h1 id=BAS stitle=“Becoming Acquainted with SGML”>
Becoming Acquainted with the Standard Generalized Markup
Language
```

5.3.6 Topics

topics of level 1-4 `<top1>`, `<top2>`, `<top3>`, `<top4>`

These tags are used to identify topic-subdivisions in the document. They can be followed by an optional topic heading tag. The end-tag can be omitted.

NOTE Topics differ from headed sections in that headed sections must be used in strict head level order: e.g. `<H1> Text <H2> Text. . .` For topics, however, there are no strict levels: e.g. `<top3> Text <top1> Text <top2> Text </top2> </top1> </top3>` is permitted. It is, however, not permitted to imbed one topic of a particular level within another topic of the same level.

topic heading `<th>`

The `<th>` tag is used to identify a topic heading. It immediately follows one of the topic tags. The end-tag can be omitted.

```
<top1><th>Important considerations
<p>When using SGML it is important to . . .
```