Cahiers GUTenberg n° 19 — janvier 1995

Introduction pratique à SGML

Michel Goossens

CERN, Division CN, 1211 Genève 23, Suisse,
E-mail: `Michel.Goossens@cern.ch`

Résumé. La norme internationale SGML (*Standard Generic Markup Language*) traite du balisage structurel des documents sous leur forme électronique. Elle fut adoptée par l'ISO (organisation internationale de normalisation) en octobre 1986 et son importance est bien reflétée dans le fait que c'est une des rares normes ISO traduites en français. Elle est vite devenue très populaire, même en Europe, notamment par son acceptation dans le monde des éditeurs, des grandes entreprises internationales, des institutions gouvernementales et, plus récemment, par le biais de HTML, le langage hypertexte de W^3. Cet article explique les éléments de base de SGML en vue de permettre une meilleure compréhension des développements dans le monde du document électronique en général et de W^3 en particulier.

Abstract. *The international standard* SGML *(Standard Generic Markup Language) deals with the structural markup of electronic documents. It was adopted by ISO (the International Organisation for Standardisation) in October 1986.* SGML *soon became very popular, also in Europe, thanks in particular to its enthusiastic acceptance in the editing world, large multi-national companies, governmental organizations, and, more recently, by the ubiquity of* HTML*, the hypertext language of WWW. This article provides an introduction to the basic ideas of* SGML *and should allow the reader a better understanding of the latest developments in the field of electronic documents in general, and in WWW in particular.*

1. Pourquoi SGML ?

Ces dernières années on a vu un passage de plus en plus rapide du support papier au support électronique dans le domaine du livre. Ainsi le monde de l'édition rejoint-il d'autres secteurs d'activités où l'on gère une masse importante d'informations structurées, comme ceux de l'édition d'annuaires, de dictionnaires ou de recueils juridiques. Là aussi l'informatique est devenue un outil privilégié pour répondre aux besoins de gestion d'une masse d'informations volumineuse qui, de plus en plus souvent, nécessite une actualisation en permanence.

En stockant les données sous forme électronique on peut envisager de multiples produits à partir d'un même document source. Ainsi peut-on, à partir d'une collection d'adresses, non seulement produire un annuaire-papier, mais aussi développer un service interrogeable sur minitel, sur Internet, ou sur CD-ROM, voire imprimer automatiquement une série d'étiquettes. À partir d'un ensemble de textes de lois ou d'une série d'articles sur l'histoire balisés en SGML, on pourra éditer un code contenant toutes les lois ou une encyclopédie historique complète, publier une revue avec des mises à jour régulières, imprimer un ouvrage thématique contenant une sélection des textes sur un sujet particulier; on pourra également proposer un service de consultation sur minitel ou par W^3 sur réseau et déveloper un système hypertexte sur CD-ROM.

Ces diverses applications supposent que l'information n'est pas gardée sous une forme correspondant à celle d'un document imprimé particulier (c.-à-d. séquentielle), mais de telle façon que sa structure logique soit clairement mise en évidence.

Pour récapituler, les points forts d'un balisage en SGML sont :

- une amélioration de la qualité des sources des documents ;
- une rationalisation dans le traitement du document, due en grande partie à un cycle de travail plus rapide ;
- une réduction des coûts des publications ;
- la possibilité de réutilisation de l'information, d'où une plus-value (imprimés, hypertextes, bases de donnés).

1.1. Origine de SGML

Une représentation des documents basée sur la description de leur structure logique est essentielle pour les traiter électroniquement. Néanmoins, pour garantir que les documents soient réellement échangeables on a dû se mettre d'accord sur un langage commun pour mettre en œuvre ce type de représentation.

C'est ce qui fut fait lorsqu'en octobre 1986 ISO (l'Organisation internationale de Normalisation) adopta officiellement SGML *(Standard Generalized Markup Language)*, le langage standard de balisage généralisé, comme norme internationale (ISO 8879) [1]. Comme la norme SGML porte le sceau de l'ISO et comme elle a été accepté très vite par plusieurs organismes nationaux et internationaux et par les développeurs de logiciels, on peut être sûr que SGML occupe dorénavant une place importante dans le monde de l'édition.

1.2. Qui utilise SGML?

Avec l'apparition d'exigences nouvelles, liées à l'environnement informatique, le mode d'échange traditionnel des documents a été bouleversé et aujourd'hui SGML est présent dans pratiquement tout produit de gestion de document ou de traitement de texte.

Parmi les applications actuelles de SGML, il faut citer d'abord celle qu'a définie l'association des éditeurs américains *(AAP : American Association of Publishers)*. L'AAP [4] a sélectionné trois types de documents dans le secteur de l'édition : le livre, la publication en série et l'article. Pour chacun d'eux, une DTD (en anglais : *document type definition*, définition de classe de document, voir plus loin, spécialement la section 4) a été élaborée. Par ailleurs, l'AAP et l'EPS *(European Physical Society)* ont collaboré pour développer une méthode de balisage SGML des tableaux et des formules mathématiques, qui est décrite dans le document ISO 12083.

Une autre application qui est en développement depuis plusieurs années est CALS *(Computer-aided Acquisition and Logistic Support)* du ministère de la défense américaine (DoD). Cette « initiative » prévoit le passage du support papier aux supports électronique de la documentation des systèmes d'armement. Toute la documentation doit en être fournie sous une forme révisable et utiliser SGML.

Parmi les exemples d'utilisation de SGML, citons :

- l'Office des Publications de la Communauté Européenne (FORMEX) ;
- l'association des éditeurs allemands *(Börsenverein des Deutschen Buchhandels)* ;
- la *British Library* avec SGML *: Guidelines for editors and publishers* et SGML *: Guidelines for authors* ;
- en France, le Syndicat national de l'édition et le Cercle de la librairie ont défini une application pour les éditeurs français [3] ;
- l'Office des Publications de l'ISO (Genève) et le HMSO pour les brevets (Angleterre) ;
- *Oxford University Press* et *Virginia Polytechnic* (PhD, USA) ;
- le *Text Encoding Initiative* (textes classiques et commentaires) ;
- la plupart des manuels techniques, avec p. ex. DTD `DocBook` ou autres, utilisés par IBM, HP, OSF, O'Reilly, etc.
- les modules SGML entrée/sortie sur des systèmes de traitement de texte ou de base de données (Frame, Interleaf, Microsoft, Oracle, Wordperfect) ;
- McGraw-Hill *(Encyclopedia of Science and Technology)* ;
- l'industrie électronique (Pinacle), l'industrie aéronautique et les compagnies aérienne (Boeing, Airbus, Rolls Royce, Lufthansa, etc.) et l'industrie pharmaceutique ;
- les agences de presse ;

- les éditeurs ou systèmes de présentation SGML (Arbortext, EBT, Exoterica, Grif, Softquad) ; voir Seybold Report numéro [?] ;

- HTML et W^3 (évidemment !).

2. Principes de base de SGML

SGML est une méthode normalisée permettant de représenter l'information contenue dans un document indépendamment des systèmes utilisés pour sa saisie ou son traitement, et indépendamment de la forme physique finale.

SGML repose sur le principe du *balisage* logique des documents et met en œuvre ce principe sous la forme d'*un balisage logique généralisé*. SGML n'est pas seulement un langage de balisage, mais un outil pour construire plusieurs langages de balisage, autrement dit *un méta-langage*.

2.1. Différents types de balisage

Les « traitements de texte » qui se sont répandus ces dernières années utilisent un balisage de type « présentation » : des caractères de contrôle sont mêlés aux caractères (imprimables) du document. Ils marquent, par exemple, la fin de ligne ou gèrent les coupures de pages, ou servent à contrôler l'espace horizontal entre les caractères. En règle générale ces caractères sont non-standard et il est très malaisé de faire circuler entre différents systèmes des documents balisés de cette façon. D'autre part ce type de balisage correspond à une représentation spécifique du document et en décrit une présentation physique, avec un découpage en lignes et en pages. Même si cette représentation rend l'affichage et l'impression d'un document plus aisés, il n'autorise cependant que des traitements fort limités.

Pour augmenter les possibilités de traitement des documents on doit décrire leur structure logique en faisant abstraction de tout aspect physique. C'est ce qu'on appelle un *balisage logique* (ou *générique*). On y décrit la fonction logique des éléments qui composent le document (titre, sections, paragraphes, tableaux, éventuellement références bibliographiques, formules mathématiques,...) et les relations qui les lient.

La figure 1 montre quelques exemples de balisage pour un même texte. On y voit clairement les différences entre un balisage spécifique, où l'on doit insérer des indications explicites pour guider la mise en page du texte (p. ex. les commandes `\vskip` ou `.sp`) et un balisage générique, où seulement la fonction logique (chapitre et début de paragraphe) est indiquée.

Balises spécifiques

TeX	Script
`\vfil\eject`	
`\par\noindent`	`.pa`
`{\bf Chapitre 2 : Titre du chapitre}`	`.bd Chapitre 2 : Titre du chapitre`
`\par\vskip\baselineskip`	`.sp`

Balises génériques ou logiques

LaTeX	HTML (SGML)
`\chapter{Titre du chapitre}`	`<H1>Titre du chapitre</H1>`
`\par`	`<P>`

FIGURE 1 - *Différents types de balises*

2.2. Balisage logique généralisé

Le principe du balisage logique consiste à indiquer (en anglais *mark*) la structure d'un document. Ce balisage s'effectue en deux temps :

1. définition de l'ensemble des balises pour identifier les éléments d'un document et les règles formelles qui décrivent sa structure (c'est le rôle de la DTD) ;

2. introduction du balisage dans le document lui-même, selon les principes de cette définition formelle.

Plusieurs documents spécifiques peuvent appartenir à une seule classe de documents construits selon la même structure générique. Par exemple, les deux articles (Figure 2) ont des structures spécifiques différentes, mais sont construits selon la même structure générique : un titre suivi d'un certain nombre de sections divisées éventuellement en sous-sections, et enfin une bibliographie. Ces documents appartiennent à la *classe de document* « article ».

La première étape consiste à définir formellement la structure de la classe de document « article ». Concrètement, il s'agit d'écrire, dans un langage défini par la norme SGML, que tout article est constitué d'un titre suivi de sections, etc. Cette écriture constitue la *définition du type de document « article »*.

Une *définition de la classe de document* (DTD) identifie donc les éléments qu'on peut trouver dans une classe de document (des sections, des sous-sections, etc.). Elle leur donne un nom, le plus souvent abrégé (p. ex. « sec » pour une section). Elle leur associe, si besoin, des *attributs* descriptifs et elle décrit les relations qui les lient : p. ex. il est interdit de trouver une bibliographie au début

```
   Article A                    Article B
Titre                        Titre
Section 1                    Section 1
    Sous-section 1.1             Sous-section 1.1
    Sous-section 1.2             Sous-section 1.2
Section 2                        Sous-section 1.3
Section 3                    Section 2
    Sous-section 3.1             Sous-section 2.1
    Sous-section 3.2             Sous-section 2.2
    Sous-section 3.3         Bibliographie
    Sous-section 3.4
Bibliographie
```

FIGURE 2 - *Deux instances d'une même classe de document « article »*

du document. En revanche, une section peut comporter ou non des sous-sections. Notons qu'il peut aussi y avoir des relations non hiérarchiques : p. ex. la relation qui lie un titre de section au renvoi qui y est fait, trois sections plus loin, n'est pas une relation hiérarchique. La définition d'une classe de document utilise les attributs associés aux éléments, pour gérer ce type de relations.

La deuxième étape consiste à baliser le document lui-même (l'article A ou l'article B) à l'aide des noms abrégés définis pour chaque élément. Avec « sec », on fabrique les *balises* `<sec>` pour marquer le début des sections, et `</sec>` pour marquer leur fin et d'une manière analogue on a `<ssec>` et `</ssec>` pour les sous-sections :

```
<article>
<tit>Introduction à SGML</tit>
<sec>Principes de base de SGML</sec>
<P>  ...
<ssec>Balisage logique généralisé</ssec>
<P>  ...
```

2.3. À quoi sert la DTD ?

Si l'on veut effectuer des traitements automatiques puissants sur les documents en prenant appui sur leur structure, il est nécessaire que celle-ci soit marquée sans erreur. Il est nécessaire aussi que cette structure soit cohérente : un document spécifique doit respecter la structure du type de document auquel il est déclaré appartenir, comme définie dans la DTD pour la classe de documents en question.

En effet une DTD définit :

- le *nom* des éléments qui peuvent être utilisés ;
- le *contenu* de chaque élément (section 4.2.1) ;

- *combien de fois* et dans quel *ordre* les éléments peuvent apparaître (section 4.2.3) ;
- si une balise de début ou de fin peut être *omise* (section 4.2.2) ;
- les *attributs* éventuels et leur valeurs par défaut (section 4.3) ;
- le nom des *entités* qui peuvent être utilisées (section 4.4).

3. Transmission des informations concernant un document

Le but de SGML est de représenter l'information contenue dans un document. Nous avons déjà expliqué à la section 2.2 que SGML opérait en deux phases pour définir la structure d'un document :

- une phase de déclaration ;
- une phase d'utilisation, à l'intérieur du document, des éléments déclarés.

Ce principe est utilisé pour la transmission de *toutes les informations concernant le document à échanger.*

Le jeu de caractères de base est le code Ascii, défini par la norme ISO 646. Ceci peut être changé par une déclaration en tête de document (comme montré aux lignes 1–76 de l'annexe à la page 51).

Un document n'est pas uniquement constitué de caractères saisis au clavier, mais peut contenir d'autres éléments, comme par exemple des caractères grecs ou des symboles mathématiques, ou encore des illustrations, des photos ou une partie provenant d'un autre document. Ces possibilités sont gérées par l'utilisation d'entités (voir la section 4.4).

Le système de balisage est construit à partir de délimiteurs, de symboles spéciaux, de mots-clés ayant une signification particulière. Par exemple si « `sec` » identifie l'élément « Section », `<sec>` est la balise qui, dans le document, marque le début d'une Section, les délimiteurs « `<` » et « `>` » indiquant respectivement le début et la fin de la balise. De même, la structure formelle du document (la définition de la classe de document) est écrite dans un langage spécifié par la norme.

La norme SGML ne fixe pas une fois pour toutes la structure des documents et les éléments qu'ils peuvent contenir c.-à-d. les délimiteurs et symboles spéciaux, mais spécifie simplement les règles de construction auxquelles ils doivent se conformer. SGML ne fixe pas non plus le langage de balisage, mais offre une *syntaxe abstraite* qui permet de construire des syntaxes particulières, selon ses besoins. La norme propose un exemple de syntaxe, appelée *la syntaxe concrète de référence*, qui est utilisée dans cet article. Il s'agit donc bien d'un *méta-langage*.

4. Structure d'une DTD

Pour mieux comprendre le fontionnement de SGML nous nous proposons d'explorer plus en détail un exemple réel d'une application SGML récente, c.-à-d. le niveau 2 de HTML, qui correspond aux fontions qu'offrent la plupart des programmes de visualisation HTML (comme Mosaic et Lynx) actuellement. La DTD de HTML2 est montrée en entier dans l'annexe (51 et suivantes). Pour identifier des lignes individuelles dans la DTD nous utiliserons ci-dessous la numérotation qui y a été ajoutée en première colonne.

La DTD commence par la partie « système », qui détermine le jeu de caractères utilisé et spécifie les options autorisées (lignes 1–76). Par exemple, dans le domaine de la minimisation, le paramètre `OMITTAG` (ligne 64) peut avoir la valeur `YES`, ce qui permettra l'omission de certaines balises, comme expliqué à la section 4.2.2. Par contre si sa valeur est `NO` aucune balise ne pourra être omise.

La deuxième partie (lignes 78-452) correspond à la définition de la classe de document `HTML`. Les éléments, leurs attributs et les entités propres à cette classe de document y sont déclarés.

À l'intérieur d'une DTD le début d'une déclaration est noté par les caractères « `<!` » et sa fin par « `>` ». Certaines sections d'une DTD peuvent être identifiées (marquées) par un mot clé qui permet de leur réserver un traitement spécial ou pour les (dés)activer en fonction de la valeur (`IGNORE` ou `INCLUDE`) de ce mot clé. La notation pour le début, respectivement la fin d'une telle *section marquée* est « `<![` *`mot_clé`* `[` » et « `]]>` ».

4.1. Commentaires

Il est toujours important d'introduire des commentaires dans un document ou dans une définition de classe de document pour rendre ceux-ci plus lisibles et pour faciliter leur mises-à-jour futures. Un commentaire SGML a la forme :

```
<!-- texte du commentaire -->
```

Un commentaire est limité par les doubles signes moins `--` et peut s'étendre sur plusieurs lignes, comme le montrent, p. ex., les lignes 329–334.

4.2. Éléments

4.2.1. Déclaration d'élément

Chaque élément de la structure logique d'un document doit être déclaré. Cette déclaration donne le nom de l'élément et indique, entre parenthèses, ce que peut ou doit contenir l'élément en question (son *modèle de contenu*).

```
<!ELEMENT nom n m (modèle de contenu)>
```

Par exemple les lignes 443 et 445 sont équivalentes à la déclaration [1] :

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

La partie entre le nom de l'élément « HTML » et le modèle de contenu « (HEAD, BODY) » décrit les possibilités de minimisation pour la balise <HTML> (voir « Omettre des balises » ci-dessous). Cette déclaration dit qu'un document HTML contient une en-tête (HEAD) *suivi* d'un corps (BODY). À la ligne 413 (et 409–411) nous voyons que l'en-tête doit contenir un titre (TITLE), et peut aussi avoir d'autres éléments (ISINDEX, BASE, META, etc.).

4.2.2. Omettre des balises

Il est possible que dans certains contextes on puisse rétablir automatiquement la présence d'une balise sous-entendue. Cette possibilité doit être spécifiée pour chaque élement en intercalant dans la déclaration d'un élément entre son nom et le modèle de contenu deux caractères (séparés par un blanc), qui correspondent, respectivement, à la balise d'ouverture et de fermeture. Il y a seulement deux valeurs possibles : - pour indiquer que la balise doit être présente et O (O majuscule et non pas un zéro) si elle peut être omise. Par exemple, pour les listes numérotées (OL) et non-numérotées (UL) et leurs éléments (LI) on a [2] :

```
<!ELEMENT (OL|UL) - -  (LI)+>            Ligne 318
<!ELEMENT LI    - O %flow>               Ligne 323
```

Le premier « - - » signifie que l'on est obligé de toujours spécifier les balises d'ouverture et de fermeture des déclarations des listes (c.-à-d. <OL>...</OL> et <UL>...</UL>) alors que le deuxième « - O » indique que pour les membres de la liste (<LI>...) la balise de fermeture n'est pas nécessaire (voir aussi les exemples de listes dans l'article de Christian Rolland).

4.2.3. Le modèle de contenu

Comme montré ci-dessus, le modèle de contenu fait appel à des opérateurs d'ordre et de choix (voir la table 1 pour une liste).

On a déjà rencontré l'opérateur de choix (|), qui spécifie que l'un des éléments spécifiés peut être présent (pas plus d'un à la fois). Considérons comme autre exemple une liste de description (<DL>) qui est déclarée (ligne 311) comme suit :

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

1. La forme utilisée dans le DTD à la ligne 445 fait appel à une entitée paramètre; voir la section 4.4.

2. La signification des symboles | et + est expliquée dans la section 4.2.3, voir spécialement la table 1 ; la définition de l'entité paramètre %flow est à la ligne 285, voir aussi la section 4.2.3.

TABLE 1 - *Symboles d'ordre et de choix*

<table>
<tr><th>symbole</th><th>description</th></tr>
<tr><td><code>,</code></td><td>tous doivent apparaître dans l'ordre indiqué (« et » ordonné)</td></tr>
<tr><td><code>&</code></td><td>tous doivent apparaître, dans n'importe quel ordre (« et » non ordonné)</td></tr>
<tr><td><code>|</code></td><td>un et un seul doit apparaître (« ou » exclusif)</td></tr>
<tr><td><code>+</code></td><td>élément obligatoire et répétable (1 fois ou plus)</td></tr>
<tr><td><code>?</code></td><td>élément optionnel (0 ou 1 fois)</td></tr>
<tr><td><code>*</code></td><td>élément optionnel et répétable (0 fois ou plus)</td></tr>
</table>

Ceci indique que, pour la liste de description, les deux balises `<DL>` *et* `</DL>` doivent toujours être présentes et qu'elle contient une ou plusieurs occurences (`(...)+`) de zéro ou plus de balises `<DT>` (`DT*`) qui peuvent être *suivies* (`,`) d'au plus une balise `<DD>` (`DD?`).

Une liste de plusieurs éléments pouvant apparaître dans un ordre quelconque se trouve par exemple en lignes 409–413, qui spécifient essentiellement qu'une en-tête HTML doit contenir, dans n'importe quel ordre, un titre (`TITLE`), zéro ou une fois les balises `<ISINDEX>` et `<BASE>` et zéro ou plus de fois `<META>` :

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

Un élément peut contenir d'autres éléments ou des caractères, ou éventuellement les deux (dans ce cas on parle de *contenu mixte*).

On peut indiquer au programme d'analyse (anglais : *parser*) le type de caractères utilisés. Pour cela on dispose des noms réservés suivants :

`PCDATA` *parsed character data* ou données textuelles analysées.
Ces caractères sont considérés comme traités par un *parser* et ils ne peuvent donc plus contenir des appels d'entités ou de balises. Par exemple (ligne 260), le titre HTML est de ce type : `<!ELEMENT TITLE - - (#PCDATA)>`

`RCDATA` *replaceable character data* ou données textuelles remplaçables.
Le *parser* ne s'attend à trouver que des caractères ou des appels d'entité, c.-à-d. les balises (de début ou de fin) sont interdites.

`CDATA` *character data* ou données textuelles.
Plus aucun traitement n'est nécessaire (le contenu peut cependant être traité par un processeur différent, p. ex. PostScript). Un numéro de téléphone dans un en-tête de lettre pourrait être déclaré : `<!ELEMENT TEL CDATA>`

`ANY` L'élément peut contenir du matériel du type `PCDATA` ou tout autre élément défini dans la DTD.

Table 2 - *Les mots-clés pour les types d'attribut*

mot-clé	*valeur de l'attribut*
`CDATA`	données textuelles (caractères quelconques)
`ENTITY(IES)`	nom(s) d'entité(s) générale(s)
`ID`	l'identificateur SGML d'un élément
`IDREF(S)`	valeur(s) d'appel d'identificateur(s) d'élément
`NAME(S)`	nom(s) SGML
`NMTOKEN(S)`	unité(s) lexicale(s) nominale(s)
`NOTATION`	nom de notation
`NUMBER(S)`	nombre(s)
`NUTOKEN(S)`	unité(s) lexicale(s) numérique(s)

`EMPTY` L'élément a un *contenu vide*. Il pourra être qualifié par d'éventuels attributs (voir la section 4.3). Par exemple la balise `<IMG>`, qui permet de spécifier des images en HTML : `<!ELEMENT IMG - O EMPTY>`

Certains éléments peuvent être utilisés n'importe où dans un document. Dans ce cas, il est opportun de les déclarer *inclus* dans l'élément document. Plus généralement, un élément inclus dans le modèle de contenu d'un autre élément est un élément qui peut appartenir à tous les constituants du dit élément. Dans ce cas la syntaxe `+(...)` est utilisée. D'une façon analogue on peut vouloir exclure certains types d'éléments ; la syntaxe `-(...)` sera alors utilisée. Par exemple, pour le formulaire électronique `<FORM>` de HTML on a la définition suivante :

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

Ceci veut dire que l'élément `<FORM>` peut contenir tout ce qui est spécifié par l'entité `%body.content` (lignes 337, 157–159, 178–182 et 279–283). Tous ces éléments peuvent contenir à n'importe quel niveau des balises `<INPUT>`, `<SELECT>` ou `<TEXTAREA>`. En revanche, la définition d'un formulaire `<FORM>` n'autorise pas la récursion (`-(FORM)`).

4.3. Attributs

Tous les attributs possibles pour chaque élément présent dans la DTD doivent être déclarés. Pour des raisons de clarté les déclarations d'attributs se font habituellement juste après les éléments auxquels ils se réfèrent.

Une déclaration d'attribut comporte :

- le nom du ou des élément(s) auquel elle se rapporte ;
- le nom de l'attribut ;
- soit le *type de l'attribut*, indiqué par un mot-clé (voir table 2), soit, entre parenthèses, la liste des valeurs que peut prendre cet attribut ;

TABLE 3 - *Les mots-clés pour les valeurs par défaut*

mot-clé	*description*
#FIXED	L'attribut a une valeur fixe, et ne peut prendre que cette valeur.
#REQUIRED	Une valeur doit obligatoirement être spécifiée par l'utilisateur.
#CURRENT	Si une valeur n'est spécifiée, la valeur par défaut utilisée sera la dernière valeur spécifiée
#CONREF	La valeur sera utilisée pour les références croisées.
#IMPLIED	Si une valeur n'est spécifiée, le système de traitement définira une valeur.

– une valeur par défaut (sous la forme d'une des valeurs autorisées, spécifiée entre guillemets, ou d'un mot-clé, voir table 3).

La déclaration d'attribut a donc la forme suivante :

```
<!ATTLIST élément_qualifié attribut (valeurs) "défaut">
```

Ainsi pour la liste de déclaration (**<DL>**) (lignes 312–313) on dispose de l'attribut « compact » pour indiquer que les éléments de la liste doivent être reserrés.

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

Cette déclaration spécifie que la seule valeur possible est **COMPACT** et que le système se chargera de mettre une valeur par défaut (**#IMPLIED**, voir table 3).

On peut aussi vouloir un nombre, comme pour indiquer la largeur de la ligne avec la balise **<PRE>** (lignes 290–291)

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

Le type de l'attribut est ici « nombre entier » (mot-clé: **NUMBER**) ; si aucun nombre n'est spécifié par l'utilisateur, le *parser* prendra une valeur par défaut fournie par le système (**#implied**).

Pour une comparaison détaillée d'une définition dans la DTD (lignes 229–233 et 206–214) et la description d'une commande[3] on peut considérer la balise **<A>**, qui permet de traiter les hyperliens en HTML.

Finalement, considérons à nouveau l'élément **<IMG>** (image) et ses attributs (lignes 241–426) qui sont essentiellement donnés par la déclaration suivante :

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

3. Voir l'article de Christian ROLLAND dans ce numéro, section « Parlons hypertexte ».

Le premier fait référence à l'entité paramètre `%URI` (voir lignes 140–152) qui définit un identificateur uniforme de ressources (en anglais : *Uniform Resource Identifier*). Cet attribut est *obligatoire* (`#REQUIRED`). Les autres attributs sont optionnels avec une valeur par défaut définie par le système (`#IMPLIED`). Dans le cas de l'alignement (`ALIGN`), un choix entre trois possibilités est offerte.

4.4. Entités

Des entités peuvent être utiles dans plusieurs circonstances :

- définition de notations raccourcies pour des suites de caractères saisies fréquemment (entités générales) ; p. ex. on pourrait définir :

  ```
  <!ENTITY GUT "GUTenberg">
  ```

- définition de notations pour saisir des caractères spéciaux, accents ou symboles (entités générales et caractères). Un exemple d'une entité caractère (ligne 168) est :

  ```
  <!ENTITY amp CDATA "&#38;" -- << et >> commercial "&" (ampersand) -->
  ```

  L'ISO a défini plusieurs ensembles d'entités caractères standardisés, p. ex. pour les caractères nationaux (voir annexe A.2 à la page 57), les symboles graphiques, mathématiques, etc. :

- inclusion de fichiers externes (entités externes) ;

- définition de variables dans une DTD (entités paramètres).

Notons que, contrairement aux noms des éléments (balises) SGML et leurs attributs qui peuvent être saisis en minuscules, majuscules ou un mélange des deux, les noms d'entités doivent être entrés textuellement, en respectant la casse des caractères.

Les entités générales sont déclarées dans la DTD. La déclaration donne le nom symbolique de l'entité, suivi de son contenu, qui peut contenir des balises, des appels d'entités, etc., qui seront interprétés au moment où l'entité sera développée.

Pour faire référence à une entité générale on utilise un *appel d'entité*, qui a la forme :

```
&nom_de_l'entité;
```

Par exemple, si l'on veut utiliser la définition de `GUT` montré ci-dessus, on saisira dans le texte `&GUT;` ce qui générera la suite de caractères équivalente « GUTenberg ».

Le contenu d'une entité peut être stocké dans un fichier différent de celui qui est utilisé pour le document en cours (entité dite *externe*). Il s'agit par exemple d'incorporer au document en cours un tableau ou une figure qui a été créé ailleurs. Au lieu de reproduire *in extenso* le contenu de l'entité dans la déclaration, on indique un nom de fichier où le matériel à inclure est stocké. Ce nom de fichier devra être précédé du mot-clé `"SYSTEM"`, p. ex. pour le système d'exploitation Unix, la déclaration pourrait avoir une forme comme :

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

À l'intérieur d'une DTD on utilise des entités paramètres, qui permettent de considérablement augmenter la modularité de la définition des différents éléments de la DTD. Des exemples simples sont (lignes 157, 159, 190) :

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

Ces entités sont utilisées, p. ex. aux lignes 225, 258, 328 et 337.

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

5. Quelques outils SGML

Comme SGML est maintenant très répandu dans le domaine de la production de documents (voir la section 1.2) de multiples solutions commerciales ou autres sont disponibles pour augmenter la productivité, le confort et la convivialité d'utilisation des systèmes SGML. Dans cette section nous présentons deux outils intéressants et disponibles dans le domaine public.

5.1. Valider un document SGML

Il est souvent important et utile de pouvoir valider un document SGML (ou HTML, le cas échéant). Pour cela il existe, entre autres, le programme d'analyse (*parser*) `sgmls`, qui est disponible dans le domaine public[4]. `sgmls` est développé par James Clark (`jjc@jclark.com`) en se basant sur un programme antérieur `arcsgml`, écrit par Charles Goldfarb, considéré par beaucoup comme le père de SGML et qui a écrit un ouvrage clé [5] traitant du standard SGML, qui devrait faire partie de la bibliothèque de chaque utilisateur sérieux de SGML.

L'appel de `sgmls` est :

```
sgmls  [-deglprsuv] [-cfichier] [-inom] [nom_de_ficher]
```

4. Par exemple sur l'archive de logiciels SGML `ftp.ifi.uio.no` dans le répertoire `/pub/SGML`, on trouve, à part `sgmls`, beaucoup d'exemples de DTD et d'information au sujet de SGML, Hytime, DSSSL, etc.

Nous ne décrirons pas les différentes options de ce programme, mais nous nous limiterons à traiter quelques documents simples avec cet outil fort intéressant.

```
<HTML>
<!-- Un commentaire -->
<HEAD>
  <TITLE>Document test HTML</TITLE>
</HEAD>
<!-- Début du corps du document -->
<BODY>
<DL>
  <DT>terme 1<DD>donnée 1
  <DT>terme 2<DD>donnée 2
  <DT>terme 3
  <DT>terme 4<DD>donnée 4<DD>donnée 4 bis
</DL>
</BODY>
</HTML>
```

Si nous traitons ce fichier par **sgmls** (en plaçant la DTD de l'annexe devant le fichier), on obtient :

```
sgmls: Warning at a, line 165 in declaration parameter 5:
       Could not find external parameter entity "ISOlat1"
sgmls: Error at a, line 166 at ";":
       Reference to non-existent parameter entity "ISOlat1" ignored
AVERSION CDATA -//IETF//DTD HTML//EN//2.0
(HTML
(HEAD
(TITLE
-Document test HTML
)TITLE
)HEAD
(BODY
ACOMPACT IMPLIED
(DL
(DT
-terme 1
)DT
(DD
-donnée 1\n
)DD
(DT
-terme 2
)DT
(DD
-donnée 2\n
)DD
(DT
```

```
-terme 3\n
)DT
(DT
-terme 4
)DT
(DD
-donnée 4
)DD
(DD
-donnée 4 bis
)DD
)DL
)BODY
)HTML
```

Les deux messages d'erreur au début indiquent, à juste titre, que le fichier d'entités paramètres externe `ISOlat1` n'est pas disponible. Ceci n'empêche pas le programme de continuer à traiter le document et de montrer ses différents éléments. Le début et la fin d'un élément avec identificateur générique (`ig`) sont indiqués, repectivement, par « `(ig` » et « `)ig` » en début de ligne. Un attribut est caractérisé par la lettre `A` en première colonne, directement suivi par le nom de l'attribut, sa valeur puis un spécificateur (voir tables 2 et 3). Un `\n` représente une fin d'enregistrement (anglais : *record end*).

Pour un document incorrect le programme `sgmls` nous indique une erreur :

```
<HTML>                              ligne 453
<BODY>                                    454
  <P>texte dans un paragraphe             455
</BODY>                                   456
</HTML>                                   457
```

Les messages d'erreur sont numérotés en utilisant les numéros de lignes donnés à droite dans le document précédent.

```
AVERSION CDATA -//IETF//DTD HTML//EN//2.0
(HTML
(HEAD
sgmls: SGML error at a, line 454 at ">":
       BODY element not allowed at this point in HEAD element
(P
-texte dans un paragraphe
)P
sgmls: SGML error at a, line 457 at ">":
       HEAD element ended prematurely; required subelement omitted
)HEAD
sgmls: SGML error at a, line 457 at ">":
       HTML element ended prematurely; required BODY omitted
)HTML
```

Déjà à la deuxième ligne (454) **sgmls** prévient que nous avons omis de spécifier une en-tête (avec `<HEAD>`) et que la balise `<BODY>` n'y est pas autorisée. Puis à la dernière ligne, en trouvant la balise `</HTML>` **sgmls** se plaint que l'en-tête (qui n'y est pas) et le document HTML lui-même (balise `<HTML>`) se sont terminés prématurément.

```
<HTML>                          ligne 453
<HEAD>                                454
<TITLE>titre</TITLE>                  455
</HEAD>                               456
<BODY>                                457
<LI>                                  458
</BODY>                               459
</HTML>
```

La sortie produite par **sgmls** montre que ce programme HTML est correct jusqu'au point où il y a l'élément de liste `<LI>` isolé (ligne 458) ce qui n'est guère apprécié de **sgmls**, qui indique que cette balise y est utilisée en dehors d'un contexte correct (qui doit être, d'après les lignes 318–319, dans la DTD à l'intérieur d'une liste du type `<OL>`, `<UL>`, `<MENU>` ou `<DIR>`).

```
(HTML
(HEAD
(TITLE
-titre
)TITLE
)HEAD
(BODY
)BODY
sgmls: SGML error at a, line 458 at ">":
       Out-of-context LI start-tag ended HTML document element (and parse)
)HTML
```

5.2. Outils d'analyse de documents SGML

Earl Hook (`ehood@convex.com`) a développé plusieurs outils `perlSGML`[5], écrits en **perl**, permettant d'analyser un document ou DTD SGML :

- **dtd2html** produit un document HTML à partir d'une DTD SGML qui permet une navigation hypertexte à travers une DTD SGML ;
- **dtddiff** compare deux DTDs et montre les différences éventuelles ;
- **dtdtree** produit une visualisation de l'arborescence hiérarchique caractérisant les relations entre les différents éléments définis dans une DTD ;
- **stripsgml** enlève les balises SGML d'un texte et essaie de traduire les appels d'entités de caractères standard en Ascii.

5. Ce système est disponible sur le site `ftp.uci.edu` dans le répertoire `pub/dtd2html`.

Examinons le programme `dtdtree` d'un peu plus près. Quand nous traitons la DTD de HTML2, nous obtenons une représentation, qui s'avère fort utile pour comprendre les relations entre les éléments du langage HTML. Pour chaque élément on y voit les éléments qu'il peut contenir. Les trois points « `...` » indiquent que le contenu de l'élément en question a déjà été décrit précédemment. Des lignes contenant des entrées avec des parenthèses indiquent la liste d'éléments qui peuvent être inclus (`(I)` et `(Ia)`) ou sont exclus (`(X)` et `(Xa)`) dans le modèle de contenu de l'élément. Ci-dessous nous imprimons en trois colonnes la sortie générée par le programme `dtdtree` pour la DTD HTML2. Pour plus de clarté quelques blocs répétés ont été éliminés et remplacés par `*|**|**|` en début de ligne et quelques lignes ont été raccourcies (marquées de `***` en fin de ligne).

```
HTML
 |
 |_body
 |  |
 |  |_#PCDATA
 |  |_a
 |  |  | (X): a
 |  |  |
 |  |  |_#PCDATA
 |  |  |_b ...
 |  |  |_br ...
 |  |  |_cite ...
 |  |  |_code ...
 |  |  |_em ...
 |  |  |_h1 ...
 |  |  |_h2 ...
 |  |  |_h3 ...
 |  |  |_h4 ...
 |  |  |_h5 ...
 |  |  |_h6 ...
 |  |  |_i ...
 |  |  |_img ...
 |  |  |_kbd ...
 |  |  |_samp ...
 |  |  |_strong ...
 |  |  |_tt ...
 |  |  |_var ...
 |  |
 |  |_address
 |  |  |
 |  |  |_#PCDATA
 |  |  |_a ...
 |  |  |_b ...
 |  |  |_br ...
 |  |  |_cite ...
 |  |  |_code ...
 |  |  |_em ...
 |  |  |_i ...
 |  |  |_img ...
 |  |  |_kbd ...
 |  |  |_p ...
 |  |  |_samp ...
 |  |  |_strong ...
 |  |  |_tt ...
 |  |  |_var ...
 |  |
 |  |_b
 |  |  |
*|**|**| Comme address
 |  |
 |  |_blockquote
 |  |  |
 |  |  |_#PCDATA
 |  |  |_a ...
 |  |  |_address ...
 |  |  |_b ...
 |  |  |_blockquote ...
 |  |  |_br ...
 |  |  |_cite ...
 |  |  |_code ...
 |  |  |_dir ...
 |  |  |_dl ...
 |  |  |_em ...
 |  |  |_form ...
 |  |  |_h1 ...
 |  |  |_h2 ...
 |  |  |_h3 ...
 |  |  |_h4 ...
 |  |  |_h5 ...
 |  |  |_h6 ...
 |  |  |_hr ...
 |  |  |_i ...
 |  |  |_img ...
 |  |  |_isindex ...
 |  |  |_kbd ...
 |  |  |_listing ...
 |  |  |_menu ...
 |  |  |_ol ...
 |  |  |_p ...
 |  |  |_pre ...
 |  |  |_samp ...
 |  |  |_strong ...
 |  |  |_tt ...
 |  |  |_ul ...
 |  |  |_var ...
 |  |  |_xmp ...
 |  |
 |  |_br
 |  |  |
 |  |  |_EMPTY
 |  |
 |  |_cite
 |  |  |
*|**|**| Comme address
 |  |
 |  |_code
 |  |  |
*|**|**| Comme address
 |  |
 |  |_dir
 |  |  | (X): ***
 |  |  |
 |  |  |_li
 |  |     | (Xa): ***
 |  |     |
*|**|*****| Comme dd
 |  |
 |  |_dl
```

```
|   |   |
|   |   |_dd
|   |   |   |
|   |   |   |_#PCDATA
|   |   |   |_a ...
|   |   |   |_b ...
|   |   |   |_blockquote ...
|   |   |   |_br ...
|   |   |   |_cite ...
|   |   |   |_code ...
|   |   |   |_dir ...
|   |   |   |_dl ...
|   |   |   |_em ...
|   |   |   |_form ...
|   |   |   |_i ...
|   |   |   |_img ...
|   |   |   |_isindex ...
|   |   |   |_kbd ...
|   |   |   |_listing ...
|   |   |   |_menu ...
|   |   |   |_ol ...
|   |   |   |_p ...
|   |   |   |_pre ...
|   |   |   |_samp ...
|   |   |   |_strong ...
|   |   |   |_tt ...
|   |   |   |_ul ...
|   |   |   |_var ...
|   |   |   |_xmp ...
|   |   |
|   |   |_dt
|   |       |
|   |       |_#PCDATA
|   |       |_a ...
|   |       |_b ...
|   |       |_br ...
|   |       |_cite ...
|   |       |_code ...
|   |       |_em ...
|   |       |_i ...
|   |       |_img ...
|   |       |_kbd ...
|   |       |_samp ...
|   |       |_strong ...
|   |       |_tt ...
|   |       |_var ...
|   |
|   |_em
|   |   |
```

```
*|**|**| Comme h1
 |   |
 |   |_form
 |   |   | (I): ***
 |   |   | (X): form
 |   |   |
 |   |   |_#PCDATA
 |   |   |_a ...
 |   |   |_address ...
 |   |   |_b ...
 |   |   |_blockquote ...
 |   |   |_br ...
 |   |   |_cite ...
 |   |   |_code ...
 |   |   |_dir ...
 |   |   |_dl ...
 |   |   |_em ...
 |   |   |_h1 ...
 |   |   |_h2 ...
 |   |   |_h3 ...
 |   |   |_h4 ...
 |   |   |_h5 ...
 |   |   |_h6 ...
 |   |   |_hr ...
 |   |   |_i ...
 |   |   |_img ...
 |   |   |_input
 |   |   |   | (Ia): ***
 |   |   |   | (Xa): form
 |   |   |   |
 |   |   |   |_EMPTY
 |   |   |
 |   |   |_isindex ...
 |   |   |_kbd ...
 |   |   |_listing ...
 |   |   |_menu ...
 |   |   |_ol ...
 |   |   |_p ...
 |   |   |_pre ...
 |   |   |_samp ...
 |   |   |_select
 |   |   |   | (Ia): ***
 |   |   |   | (Xa): form
 |   |   |   |
 |   |   |   |_option
 |   |   |       | (Ia): ***
 |   |   |       | (Xa): form
 |   |   |       |
 |   |   |       |_#PCDATA
```

```
 |   |   |
 |   |   |_strong ...
 |   |   |_textarea
 |   |   |   | (Ia): ***
 |   |   |   | (Xa): form
 |   |   |   |
 |   |   |   |_#PCDATA
 |   |   |
 |   |   |_tt ...
 |   |   |_ul ...
 |   |   |_var ...
 |   |   |_xmp ...
 |   |
 |   |_h1
 |   |   |
 |   |   |_#PCDATA
 |   |   |_a ...
 |   |   |_b ...
 |   |   |_br ...
 |   |   |_cite ...
 |   |   |_code ...
 |   |   |_em ...
 |   |   |_i ...
 |   |   |_img ...
 |   |   |_kbd ...
 |   |   |_samp ...
 |   |   |_strong ...
 |   |   |_tt ...
 |   |   |_var ...
 |   |
 |   |_h2
 |   |   |
*|**|**| Comme h1
 |   |
 |   |_h3
 |   |   |
*|**|**| Comme h1
 |   |
 |   |_h4
 |   |   |
*|**|**| Comme h1
 |   |
 |   |_h5
 |   |   |
*|**|**| Comme h1
 |   |
 |   |_h6
 |   |   |
*|**|**| Comme h1
```

```
 |  |
 |  |_hr
 |  |  |
 |  |  |_EMPTY
 |  |
 |  |_i
 |  |  |
*|**|**| Comme h1
 |  |
 |  |_img
 |  |  |
 |  |  |_EMPTY
 |  |
 |  |_isindex
 |  |  |
 |  |  |_EMPTY
 |  |
 |  |_kbd
 |  |  |
*|**|**| Comme h1
 |  |
 |  |_listing
 |  |  |
 |  |  |_CDATA
 |  |
 |  |_menu
 |  |  | (X): ***
 |  |  |
 |  |  |_li ...
 |  |
 |  |_ol
 |  |  |
 |  |  |_li ...
 |  |
 |  |_p
```

```
 |  |  |
*|**|**| Comme h1
 |  |
 |  |_pre
 |  |  |
 |  |  |_#PCDATA
 |  |  |_a ...
 |  |  |_b ...
 |  |  |_br ...
 |  |  |_cite ...
 |  |  |_code ...
 |  |  |_em ...
 |  |  |_hr ...
 |  |  |_i ...
 |  |  |_kbd ...
 |  |  |_samp ...
 |  |  |_strong ...
 |  |  |_tt ...
 |  |  |_var ...
 |  |
 |  |_samp
 |  |  |
*|**|**| Comme h1
 |  |
 |  |_strong
 |  |  |
*|**|**| Comme h1
 |  |
 |  |_tt
 |  |  |
*|**|**| Comme h1
 |  |
 |  |_ul
 |  |  |
 |  |  |_li ...
```

```
 |  |
 |  |_var
 |  |  |
*|**|**| Comme h1
 |  |
 |  |_xmp
 |     |
 |     |_CDATA
 |
 |_head
 |  |
 |  |_base
 |  |  |
 |  |  |_EMPTY
 |  |
 |  |_isindex ...
 |  |_link
 |  |  |
 |  |  |_EMPTY
 |  |
 |  |_meta
 |  |  |
 |  |  |_EMPTY
 |  |
 |  |_nextid
 |  |  |
 |  |  |_EMPTY
 |  |
 |  |_title
 |     |
 |     |_#PCDATA
 |
 |_plaintext
    |
    |_CDATA
```

Documenter une DTD

Pour documenter une DTD (donc un langage SGML particulier) on peut utiliser le programme `dtd2html` qui produit, à partir de la DTD en question et d'un fichier de description des éléments, une représentation hypertexte (en HTML) des différents élements du langage SGML utilisé. Ceci permet aux utilisateurs d'un système de documentation basé sur SGML de trouver relativement facilement l'information au sujet des éléments d'une DTD dont ils ont besoin pour baliser leur document. Par exemple, pour le cas de HTML2, la figure 3 montre cette représentation visualisée avec le programme `mosaic`.

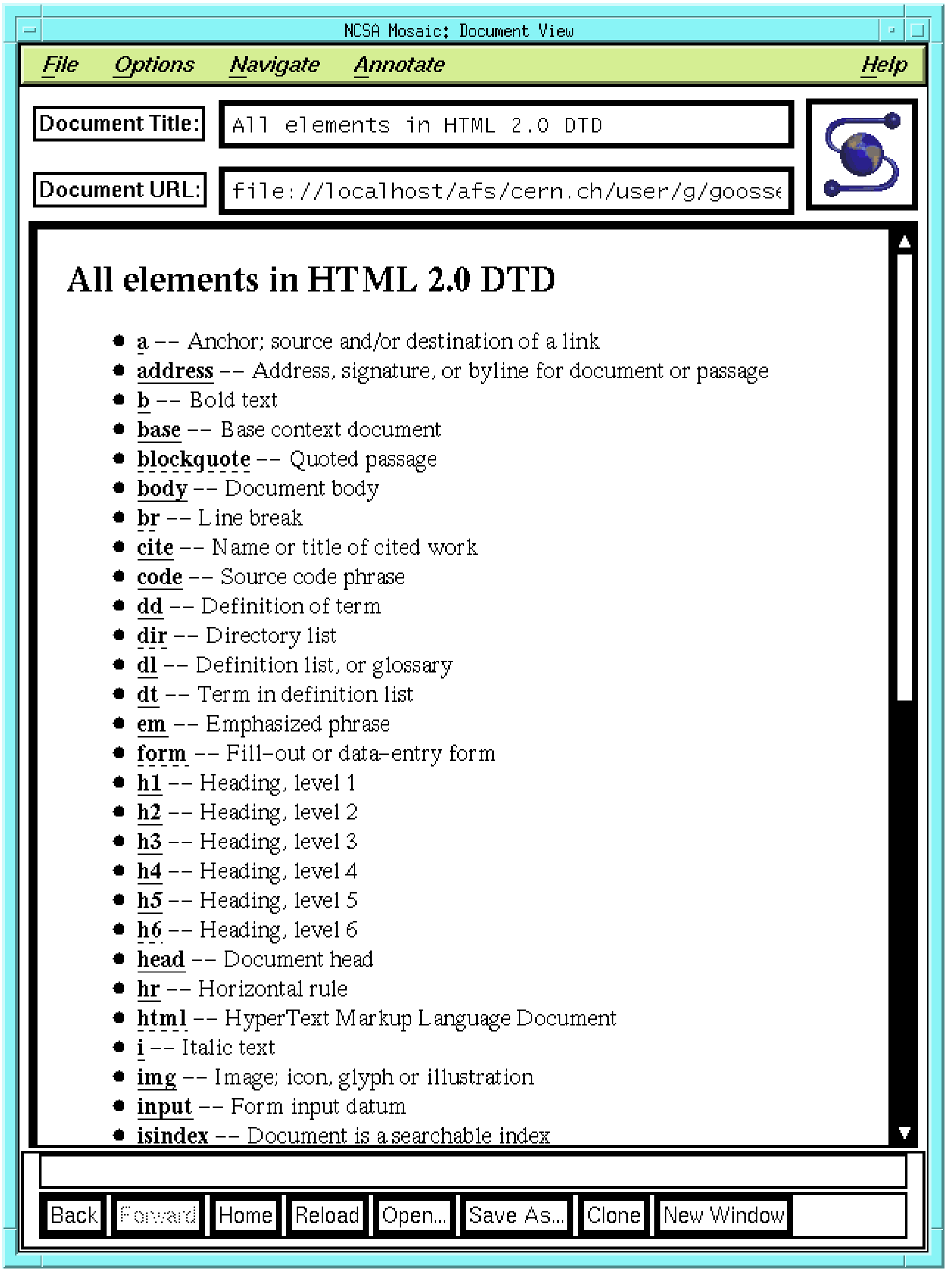

FIGURE 3 - *Description hypertexte des éléments d'une DTD* (HTML*2*) *visualisée à l'aide du programme* `mosaic`

Recherche d'information et index

Un moteur de recherche par expression régulière dans la DTD de HTML2 est disponible en spécifiant l'URL suivant à un programme de visualisation HTML (figure 4) :

`http://hopf.math.nwu.edu/html2.0/dosearch.html`

Un index de plus de 1100 mots et phrases est disponible à l'URL (Figure 5) :

`http://hopf.math.nwu.edu/html2.0/docindex.html`

6. Autres normes dans le domaine du document électronique

SGML fait partie d'un vaste projet conçu par l'organisation internationale de normalisation (ISO) qui doit aboutir à la conception d'un modèle normalisé de langages chargés de prendre en compte l'ensemble des processus qui traitent de la création, de l'échange et du traitement du texte. Ce modèle comprend plusieurs normes, certaines déjà adoptées, d'autres encore en discussion (voir [6, 7]).

SGML (Standard Generalized Markup Language)

C'est la norme ISO 8879, décrite dans cet article. Elle concerne la création et d'édition des documents. Un standard complémentaire est SDIF (ISO 9096) qui décrit un format pour échanger des documents SGML. Hytime (ISO 10744) décrit un formalisme pour la représentation hypermédia de documents. Le langage Hytime [8, 9] permet la description de situations dépendant du temps (p. ex. les CD-I).

DSSSL (Document Style Semantics and Specification Language)

Cette pré-norme ISO DIS 10179 [10], qui pourra être adoptée début 1995 permettra d'exprimer l'ensemble des concepts et des actions nécessaires pour passer de la structure logique d'un document à sa mise en forme physique. Ceci concerne essentiellement la composition des documents, mais il est prévu que DSSSL permette de définir d'autres types de mise en forme, comme le chargement d'une base de données.

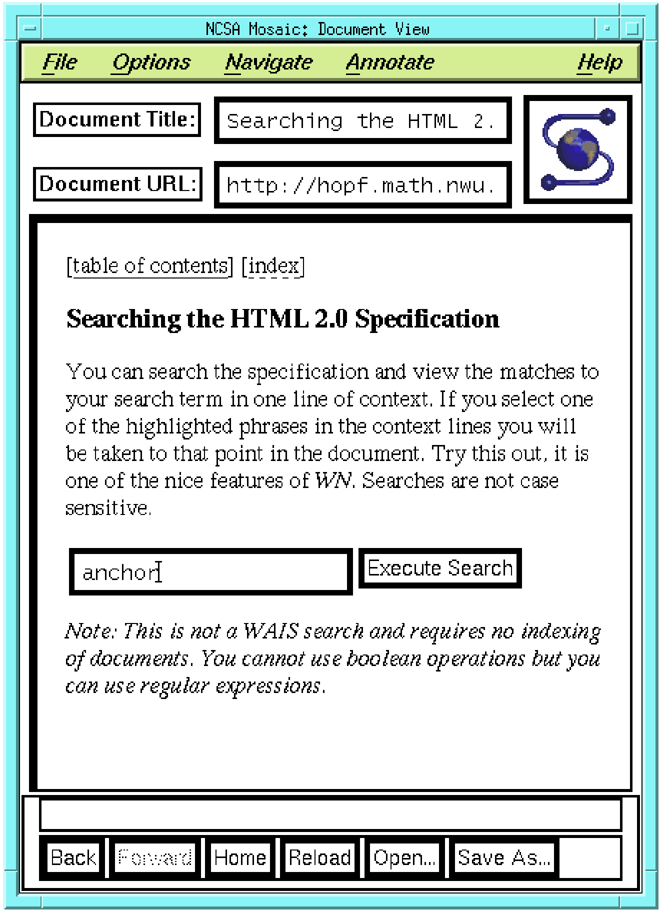

FIGURE 4 - *Une recherche dans la DTD* HTML*2*

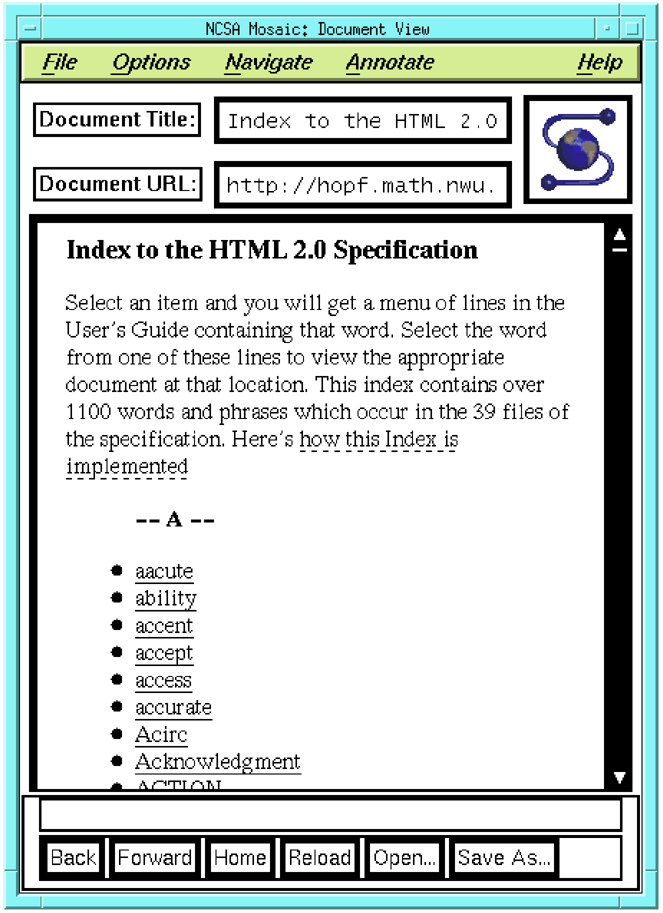

FIGURE 5 - *Un index pour la DTD* HTML*2*

SPDL (Standard Page Description Language)

Cette pré-norme ISO DIS 10180 [11] définit un formalisme pour la description des documents dans leur forme finale, entièrement composée, non révisable. Il concerne la fonction de réalisation des documents. Dans sa structure le langage ressemble très fort à PostScript, ce qui n'est pas surprenant puisque PostScript est depuis longtemps le standard « par défaut » pour la description des pages.

Polices

Pour permettre l'échange de documents on doit aussi définir un format pour les polices de caractères. Ce standard ISO 9541 [12] décrit une méthode pour nommer et grouper les glyphes ou collections de glyphes indépendante de tout codage spécifique.

Remerciements

Je tiens à remercier M. Janne SAARELA pour son aide dans l'installation du programme `sgmls` et des discussions stimulantes et Mme Lysiane BESSON et MM. Christian ROLLAND et Arnaud TADDEI, qui ont eu la gentillesse de lire une version préliminaire de cet article et y ont suggéré plusieurs améliorations.

Références bibliographiques

[1] Organisation internationale de normalisation. *Langage normalisé de balise généralisé (SGML).* ISO 8879-1986(F), ISO Genève, 1986.

[2] E. van HERWIJNEN. *Practical SGML (Second Edition).* Wolters-Kluwer Academic Publishers, Boston, 1994.

[3] D. VIGNAUD. *L'édition structurée des documents.* Éditions du Cercle de la Librairie, Paris, 1990.

[4] American National Standards Institute. *American National Standard for Electronic Manuscript Preparation and Markup.* ANSI/NISO Z39.59-1988, 1988.

[5] Charles F. GOLDFARB. *The SGML Handbook.* Oxford University Press, 1992.

[6] Michel GOOSSENS et Eric van HERWIJNEN. Introduction à SGML, DSSSL et SPDL. *Cahiers GUTenberg*, n° 12, pages 37–56, décembre 1991.

[7] Michel GOOSSENS et Eric van HERWIJNEN. Scientific Text Processing. *International Journal of Modern Physics C*, vol. 3(3), pages 479–546, juin 1992.

[8] Charles F. GOLDFARB. HyTime: A standard for structured hypermedia interchange. *IEEE Computer*, pages 81–84, août 1991.

[9] International Organization for Standardization. *Hypermedia/Time-based Structuring Language (Hytime).* ISO 10744, ISO Geneva, 1992.

[10] International Organization for Standardization. *Document Style Semantics and Specification Language.* ISO DIS 10179.2, ISO Geneva, 1994.

[11] International Organization for Standardization, newblock *Standard Page Description Language.* ISO DIS 10180, ISO Geneva, 1991.

[12] International Organization for Standardization. *Font information interchange (trois parties).* ISO 9541-1,2,3, ISO Geneva, 1991 et 1993.

Annexes

A.1. Définition complète de la DTD du langage HTML2

```
<!SGML  "ISO 8879:1986"
--
        Document Type Definition for the HyperText Markup Language
        as used by the World Wide Web application (HTML DTD).

        NOTE: This is a definition of HTML with respect to
        SGML, and assumes an understanding of SGML terms.

        If you find bugs in this DTD or find it does not compile
        under some circumstances please mail www-bug@info.cern.ch
--

CHARSET
         BASESET  "ISO 646:1983//CHARSET
                   International Reference Version (IRV)//ESC 2/5 4/0"
         DESCSET  0   9   UNUSED
                  9   2   9
                  11  2   UNUSED
                  13  1   13
                  14  18  UNUSED
                  32  95  32
                  127 1   UNUSED
     BASESET   "ISO Registration Number 100//CHARSET
               ECMA-94 Right Part of Latin Alphabet Nr. 1//ESC 2/13 4/1"
     DESCSET   128 32 UNUSED
               160 95 32
               255  1 UNUSED


CAPACITY        SGMLREF
                TOTALCAP        150000
                GRPCAP          150000

SCOPE    DOCUMENT
SYNTAX
         SHUNCHAR CONTROLS 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
                           19 20 21 22 23 24 25 26 27 28 29 30 31 127 255
         BASESET  "ISO 646:1983//CHARSET
                   International Reference Version (IRV)//ESC 2/5 4/0"
         DESCSET  0 128 0
         FUNCTION RE          13
                  RS          10
                  SPACE       32
                  TAB SEPCHAR  9
         NAMING   LCNMSTRT ""
                  UCNMSTRT ""
                  LCNMCHAR ".-"
                  UCNMCHAR ".-"
                  NAMECASE GENERAL YES
                           ENTITY  NO
         DELIM    GENERAL  SGMLREF
```

```
                 SHORTREF SGMLREF
         NAMES    SGMLREF
         QUANTITY SGMLREF
                 NAMELEN  32
                 TAGLVL   100
                 LITLEN   1024
                 GRPGTCNT 150
                 GRPCNT   64

FEATURES
  MINIMIZE
    DATATAG  NO
    OMITTAG  YES
    RANK     NO
    SHORTTAG NO
  LINK
    SIMPLE   NO
    IMPLICIT NO
    EXPLICIT NO
  OTHER
    CONCUR   NO
    SUBDOC   NO
    FORMAL   YES
  APPINFO    NONE
>

<!DOCTYPE HTML [

<!--    html.dtd

        Document Type Definition for the HyperText Markup Language (HTML DTD)

        $Id: html.dtd,v 1.19 1994/09/23 22:46:51 connolly Exp $

        Author: Daniel W. Connolly <connolly@hal.com>
        See Also: html.decl, html-0.dtd, html-1.dtd
                  http://www.hal.com/%7Econnolly/html-spec/index.html
                  http://info.cern.ch/hypertext/WWW/MarkUp2/MarkUp.html
-->

<!ENTITY % HTML.Version
        "-//IETF//DTD HTML//EN//2.0"

        -- Typical usage:

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


<!--================== Feature Test Entities ==============================-->

<!ENTITY % HTML.Recommended "IGNORE"
        -- Certain features of the language are necessary for compatibility
           with widespread usage, but they may compromise the structural
           integrity of a document. This feature test entity enables
           a more prescriptive document type definition that eliminates
           the above features.
        -->

<![ %HTML.Recommended [
        <!ENTITY % HTML.Deprecated "IGNORE">
]]>

<!ENTITY % HTML.Deprecated "INCLUDE"
        -- Certain features of the language are necessary for compatibility
           with earlier versions of the specification, but they tend
           to be used an implemented inconsistently, and their use is
           deprecated. This feature test entity enables a document type
           definition that eliminates these features.
        -->

<!ENTITY % HTML.Highlighting "INCLUDE">
<!ENTITY % HTML.Forms "INCLUDE">
```

```
<!--================== Imported Names ====================================-->

<!ENTITY % Content-Type "CDATA"
        -- meaning a MIME content type, as per RFC1521
        -->

<!ENTITY % HTTP-Method "GET | POST"
        -- as per HTTP specification
        -->

<!ENTITY % URI "CDATA"
        -- The term URI means a CDATA attribute
           whose value is a Uniform Resource Identifier,
           as defined by
        "Universal Resource Identifiers" by Tim Berners-Lee
        aka http://info.cern.ch/hypertext/WWW/Addressing/URL/URI_Overview.html
        aka RFC 1630

        Note that CDATA attributes are limited by the LITLEN
        capacity (1024 in the current version of html.decl),
        so that URIs in HTML have a bounded length.

        -->


<!-- DTD "macros" -->

<!ENTITY % heading "H1|H2|H3|H4|H5|H6">

<!ENTITY % list " UL | OL | DIR | MENU " >


<!--================ Character mnemonic entities ==========================-->

<!ENTITY % ISOlat1 PUBLIC
  "-//IETF//ENTITIES Added Latin 1 for HTML//EN">
%ISOlat1;

<!ENTITY amp CDATA "&#38;"     -- ampersand          -->
<!ENTITY gt CDATA "&#62;"      -- greater than       -->
<!ENTITY lt CDATA "&#60;"      -- less than          -->
<!ENTITY quot CDATA "&#34;"    -- double quote       -->


<!--=================== 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">

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

<!ELEMENT (%font;|%phrase) - - (%text)+>

]]>

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

<!ELEMENT BR    - O EMPTY>


<!--================== 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 -- forward relationship type --
        REV %linkType #IMPLIED -- reversed relationship type
                              to referent data --
        URN CDATA #IMPLIED -- universal resource number --

        TITLE CDATA #IMPLIED -- advisory only --
        METHODS NAMES #IMPLIED -- supported public methods of the object:
                                        TEXTSEARCH, GET, HEAD, ... --
        ">

<![ %HTML.Recommended [
        <!ENTITY % A.content   "(%text)+"
        -- <H1><a name="xxx">Heading</a></H1>
                is preferred to
           <a name="xxx"><H1>Heading</H1></a>
        -->
]]>

<!ENTITY % A.content   "(%heading|%text)+">

<!ELEMENT A     - - %A.content -(A)>

<!ATTLIST A
        HREF %URI #IMPLIED
        NAME %linkName #IMPLIED
        %linkExtraAttributes;
        >

<!--=================== Images ==========================================-->

<!ENTITY % img.alt.default "#IMPLIED"
        -- ALT attribute required in Level 0 docs -->

<!ELEMENT IMG    - O EMPTY --  Embedded image -->
<!ATTLIST IMG
        SRC %URI;  #REQUIRED     -- URI of document to embed --
        ALT CDATA %img.alt.default;
        ALIGN (top|middle|bottom) #IMPLIED
        ISMAP (ISMAP) #IMPLIED
        >


<!--=================== Paragraphs=======================================-->

<!ELEMENT P     - O (%text)+>


<!--=================== Headings, Titles, Sections ======================-->

<!ELEMENT HR    - O EMPTY -- horizontal rule -->

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

<!ELEMENT TITLE - -  (#PCDATA)
          -- The TITLE element is not considered part of the flow of text.
             It should be displayed, for example as the page header or
             window title.
          -->


<!--=================== Text Flows ======================================-->

<![ %HTML.Forms [
        <!ENTITY % block.forms "| FORM | ISINDEX">
]]>

<!ENTITY % block.forms "">

<![ %HTML.Deprecated [
        <!ENTITY % preformatted "PRE | XMP | LISTING">
]]>

<!ENTITY % preformatted "PRE">

<!ENTITY % block "P | %list | DL
        | %preformatted
```

```
        | BLOCKQUOTE %block.forms">

<!ENTITY % flow "(%text|%block)*">

<!ENTITY % pre.content "#PCDATA | A | HR | BR">
<!ELEMENT PRE - - (%pre.content)+>

<!ATTLIST PRE
        WIDTH NUMBER #implied
        >

<![ %HTML.Deprecated [

<!ENTITY % literal "CDATA"
        -- special non-conforming parsing mode where
           the only markup signal is the end tag
           in full
        -->

<!ELEMENT XMP - -  %literal>
<!ELEMENT LISTING - -  %literal>
<!ELEMENT PLAINTEXT - O %literal>

]]>


<!--=================== Lists ============================================-->

<!ELEMENT DL    - -  (DT*, DD?)+>
<!ATTLIST DL
        COMPACT (COMPACT) #IMPLIED>

<!ELEMENT DT    - O (%text)+>
<!ELEMENT DD    - O %flow>

<!ELEMENT (OL|UL) - -  (LI)+>
<!ELEMENT (DIR|MENU) - -  (LI)+ -(%block)>
<!ATTLIST (%list)
        COMPACT (COMPACT) #IMPLIED>

<!ELEMENT LI    - O %flow>

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

<![ %HTML.Recommended [
        <!ENTITY % body.content "(%heading|%block|HR|ADDRESS)*">
        -- <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>

<!ELEMENT BLOCKQUOTE - - %body.content>

<![ %HTML.Recommended [
        <!ENTITY % address.content "(%text)*">
]]>
<!ENTITY % address.content "(%text|P)*">
<!ELEMENT ADDRESS - - %address.content>


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

<![ %HTML.Forms [

<!ELEMENT FORM - - %body.content -(FORM) +(INPUT|SELECT|TEXTAREA)>
<!ATTLIST FORM
        ACTION %URI #REQUIRED
        METHOD (%HTTP-Method) GET
        ENCTYPE %Content-Type; "application/x-www-form-urlencoded"
        >
```

```

<!ENTITY % InputType "(TEXT | PASSWORD | CHECKBOX |
                        RADIO | SUBMIT | RESET |
                        IMAGE | HIDDEN )">
<!ELEMENT INPUT - O EMPTY>
<!ATTLIST INPUT
        TYPE %InputType TEXT
        NAME CDATA #IMPLIED -- required for all but submit and reset --
        VALUE CDATA #IMPLIED
        SRC %URI #IMPLIED -- for image inputs --
        CHECKED (CHECKED) #IMPLIED
        SIZE CDATA #IMPLIED -- like NUMBERS,
                                but delimited with comma, not space --
        MAXLENGTH NUMBER #IMPLIED
        ALIGN (top|middle|bottom) #IMPLIED
        >

<!ELEMENT SELECT - - (OPTION+)>
<!ATTLIST SELECT
        NAME CDATA #REQUIRED
        SIZE NUMBER #IMPLIED
        MULTIPLE (MULTIPLE) #IMPLIED
        >

<!ELEMENT OPTION - O (#PCDATA)>
<!ATTLIST OPTION
        SELECTED (SELECTED) #IMPLIED
        VALUE CDATA #IMPLIED
        >

<!ELEMENT TEXTAREA - - (#PCDATA)>
<!ATTLIST TEXTAREA
        NAME CDATA #REQUIRED
        ROWS NUMBER #REQUIRED
        COLS NUMBER #REQUIRED
        >

]]>


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

<!ENTITY % head.link "& LINK*">

<![ %HTML.Recommended [
        <!ENTITY % head.nextid "">
]]>
<!ENTITY % head.nextid "& NEXTID?">

<!ENTITY % head.content "TITLE & ISINDEX? & BASE? & META*
                         %head.nextid
                         %head.link">

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

<!ELEMENT LINK - O EMPTY>
<!ATTLIST LINK
        HREF %URI #REQUIRED
        %linkExtraAttributes; >

<!ELEMENT ISINDEX - O EMPTY>

<!ELEMENT BASE - O EMPTY>
<!ATTLIST BASE
        HREF %URI; #REQUIRED
        >

<!ELEMENT NEXTID - O EMPTY>
<!ATTLIST NEXTID N %linkName #REQUIRED>

<!ELEMENT META - O EMPTY    -- Generic Metainformation -->
<!ATTLIST META
        HTTP-EQUIV  NAME    #IMPLIED  -- HTTP response header name  --
        NAME        NAME    #IMPLIED  -- metainformation name       --
        CONTENT     CDATA   #REQUIRED -- associated information     --
        >

```

```
<!--================ Document Structure ==================================-->

<![ %HTML.Deprecated [
        <!ENTITY % html.content "HEAD, BODY, PLAINTEXT?">
]]>
<!ENTITY % html.content "HEAD, BODY">

<!ELEMENT HTML O O  (%html.content)>
<!ENTITY % version.attr "VERSION CDATA #FIXED &#34;%HTML.Version;&#34;">

<!ATTLIST HTML
        %version.attr;-- report DTD version to application --
        >

]>
```

A.2. Jeu d'entités ISOlatin1

Pour avoir une idée de la façon dont sont définis les jeux d'entités de type caractère en pratique, nous montrons ci-dessous le début du fichier correspondant au standard ISO 8859-1, et repris comme entité publique `ISOlat1` dans le contexte d'ISO 8879 SGML. Ceci peut être comparé à la table dans l'annexe de l'article de Christian ROLLAND dans ce *Cahier GUTenberg*, page 82.

```
<!-- (C) International Organization for Standardization 1986
     Permission to copy in any form is granted for use with
     conforming SGML systems and applications as defined in
     ISO 8879, provided this notice is included in all copies.
-->
<!-- Character entity set. Typical invocation:
     <!ENTITY % ISOlat1 PUBLIC
       "ISO 8879-1986//ENTITIES Added Latin 1//EN" "iso-lat1.ent">
     %ISOlat1;
-->
<!ENTITY aacute "&#225;"--=small a, acute accent-->
<!ENTITY Aacute "&#193;"--=capital A, acute accent-->
<!ENTITY acirc  "&#226;"--=small a, circumflex accent-->
<!ENTITY Acirc  "&#194;"--=capital A, circumflex accent-->
<!ENTITY agrave "&#224;"--=small a, grave accent-->
<!ENTITY Agrave "&#192;"--=capital A, grave accent-->
<!ENTITY aring  "&#229;"--=small a, ring-->
<!ENTITY Aring  "&#197;"--=capital A, ring-->
<!ENTITY atilde "&#227;"--=small a, tilde-->
<!ENTITY Atilde "&#195;"--=capital A, tilde-->
<!ENTITY auml   "&#228;"--=small a, dieresis or umlaut mark-->
<!ENTITY Auml   "&#196;"--=capital A, dieresis or umlaut mark-->
<!ENTITY aelig  "&#230;"--=small ae diphthong (ligature)-->
<!ENTITY AElig  "&#198;"--=capital AE diphthong (ligature)-->
<!ENTITY ccedil "&#231;"--=small c, cedilla-->
<!ENTITY Ccedil "&#199;"--=capital C, cedilla-->
<!ENTITY eth    "&#240;"--=small eth, Icelandic-->
<!ENTITY ETH    "&#208;"--=capital Eth, Icelandic-->
<!ENTITY eacute "&#233;"--=small e, acute accent-->
```

```
<!ENTITY Eacute "&#201;"--=capital E, acute accent-->
<!ENTITY ecirc  "&#234;"--=small e, circumflex accent-->
<!ENTITY Ecirc  "&#202;"--=capital E, circumflex accent-->
<!ENTITY egrave "&#232;"--=small e, grave accent-->
<!ENTITY Egrave "&#200;"--=capital E, grave accent-->
<!ENTITY euml   "&#235;"--=small e, dieresis or umlaut mark-->
<!ENTITY Euml   "&#203;"--=capital E, dieresis or umlaut mark-->
<!ENTITY iacute "&#237;"--=small i, acute accent-->
<!ENTITY Iacute "&#205;"--=capital I, acute accent-->
<!ENTITY icirc  "&#238;"--=small i, circumflex accent-->
<!ENTITY Icirc  "&#206;"--=capital I, circumflex accent-->
```