情 報 学 基 礎 42－9
（1996. 7. 26）

SGMLにおけるDTD導出体系

今郷 詔, 西村美苗
imago@ic.rdc.ricoh.co.jp
(株) リコー 情報通信研究所

多様な環境間での文書交換に広く利用されているSGMLは文書記述メタ言語であり、文書の記述の前にDTD(文書型定義)が必要である。しかしどのような文書でも記述できる万能のDTDはなく、DTDの共通性と記述力とは相反する。アプリケーションはDTDに依存するので、DTDの種類が増えるとそれだけ開発・保守作業も増大する。

そこで我々は、共通性の高いDTD(ベースDTD)を一つ用意し、必要とされる様々なDTDをそこから導出するDTD導出体系を開発した。ベースDTDは文書要素の構文的な役割を規定し、導出DTDは文書要素の意味的な役割を規定する。導出DTDのインスタンスはベースDTDのインスタンスに変換することができ、アプリケーションの共通化が可能である。

An architecture to derive SGML DTDs

IMAGO Satosi, NISHIMURA Mina
Information and communication R&D Center, RICOH Co.,Ltd.

SGML, which is used for document interchange among various environment, is a meta language to describe documents. Before marking up a document, we need to prepare a DTD that defines a document structure.

In general, a DTD applicable to diverse document classes is incompatible with a DTD focusing on the semantic features of documents. If the number of DTDs grows, the costs of developing application programs for the DTDs would also skyrocket.

To apply a DTD focusing on the semantic features to diverse document classes, we developed a system which, from one base general DTD, derives a different DTD for each document class. Our system also has a function that translates derived DTD instances to base DTD instances. This function frees us from the burden of developing application programs separately for each of the derived DTDs.

1 はじめに

計算機ネットワークの普及により、サーバに置かれた文書を様々なクライアントからアクセスすることが一般化し、一つの文書を様々な環境で処理することが多くなってきた。また文書データベースにおいても、一次情報を多様に利用でき、二次情報の自動抽出も可能な文書形式が求められている。このような状況で必要になるのは、アプリケーションから独立しており安定した再加工可能な文書フォーマットである。この要求を最もよく満たすフォーマットがSGML[1] である。

SGML は文書記述言語と呼ばれているが、実は文書記述言語を記述する言語であり、実際に文書を記述する前に DTD (Document Type Definition) という文書の型を定義するデータを記述しておかねばならない。ISO 12083[2] や ISO/IEC TR 9573[3] で定義されている DTD のように、ある程度汎用になるよう設計された DTD はあるが、どのような文書でも記述できる万能の DTD といったものは存在しない。文書の種類・利用目的に応じて DTD を使い分ける必要がある。DTD を一から設計するのは大変なので、実際には公開されている DTD を基に、なんらかの改造・拡張を行なった DTD を用いている場合が多い [4]。

改造・拡張を行なった DTD のインスタンス[1]は、元の DTD には適合しないことが多い。SGML のアプリケーションは、DTD 毎に用意する必要があるので、改造・拡張を行う度にすべてのアプリケーションを対応させる作業が発生し、DTD の種類に比例したコストがかかる。また、元の DTD を修正する必要が生じた時、その DTD から派生した DTD すべてを人間が修正する必要があり、コストおよび作業の信頼性の面で問題がある。

我々は、基になる DTD とその改造という過程に着目し、体系的にこの過程を支援する技術である DTD 導出体系を開発した。DTD 導出体系を用いれば、DTD 毎にアプリケーションを対応させる作業を軽減し、元の DTD の修正を自動的に派生した DTD すべてに反映できる。本稿では、SGML の簡単な説明の後に DTD 導出体系の概要とその評価を述べる。

[1]DTD に基づいて記述した文書本体.

```
<!DOCTYPE Memo SYSTEM
<Memo Author="山田">
<Title>文書のタイトル</To>
<Body>文書の本体を記す。</Body>
</Memo>
```

図 1: SGML インスタンスの例

```
<!ELEMENT Memo  (Title, Body)>
<!ELEMENT Title (#PCDATA)    >
<!ELEMENT Body  (#PCDATA)    >
<!ATTLIST Memo
    Author CDATA #IMPLIED>
```

図 2: DTD の例

2 SGML と DTD

2.1 SGML とは

SGML 文書の本体 (インスタンス) はエレメントと呼ばれる部品から構成される。エレメントは階層構造をなし、エレメントはタグと呼ばれる<name>や</name> のようなマークで先頭と末尾とを表現する。インスタンスの例を図 1に示す。

エレメントの名前や相互関係などの特性は DTD でエレメントタイプとして定義される。DTD は文書作成の度に作成するのではなく、文書交換を行なう組織内であらかじめ用意しておくのが普通である。DTD の例を図 2に示す。この DTD には図 1のインスタンスが適合する。

SGML 自体は文書構造を定義している訳ではなく、DTD で個々の具体的な文書構造を定義することになるので、SGML のアプリケーションは DTD の内容に従って処理を行なうことになる。

2.2 DTD 運用上の問題点

ある一つの文書を SGML で表現する場合でも、考え得る DTD は 1 つだけではない。図 3に示すように、一つの DTD で幅広い種類の文書を表現しようとすると、それぞれの文書の構成要素の持つ意味の細かな差を捨象して、一つのエレメントタイプがなるべく多くの文書で使用できるようにしなければならない。意味の細かな差が重要ならば、表現できる文書の種類を絞らざるを得ない。どの程度の共通性が必要かは、

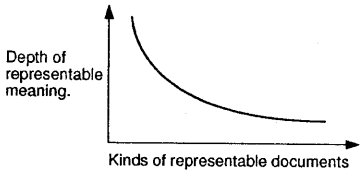

図 3: 1つのDTDで表現できる意味と文書種類

対象とする文書の使用目的に依存する。

SGMLのエレメントには様々な特性があり、DTDを設計するときにどの特性を用いて情報を表現するかは設計者の裁量次第で、次のような選択肢がある。

- ある情報をエレメントとして表現するか、属性として表現するか[5]。

  情報の内容に別のエレメントが含まれる可能性がある場合はエレメントとして表現しなくてはならないが、単なる文字列のみを内容とする場合はエレメントとしても属性としても表現できる。

- 異なる種類の情報を別々のエレメントタイプとして表現するか、一つのエレメントタイプとして属性で区別するか。

  エレメントタイプを細かく分けると入力者は簡単になるが、1つのDTDが適用できる範囲が狭くなる。またDTDの種類が増えるので、アプリケーション作成の手間が大きくなる。

どう情報を表現するかは、どのようなアプリケーションを使うかに大きく依存する。例えば画面表示を行なうアプリケーションでは、紙に比べて解像度がかなり劣るため、細かなスタイル調整は難しい。したがってDTDの共通性を高めてアプリケーション作成の手間を減らすことが望ましい。一方エディタでの入力を考えると、属性の入力はエレメントの入力より直感的でないことが多いので、なるべくエレメントタイプを分けておいた方がよい。

しかし文書インスタンスに対応するDTDは1つだけであり、すべてのアプリケーションやユーザの要求を満たすことはできない。

DTD導出体系はDTDを系統立てて導出することによって、一つの文書インスタンスを共通性の高いDTDと、エレメントタイプを細かく区別した粒度の細かなDTDの2種類のDTDに対応させることができる。

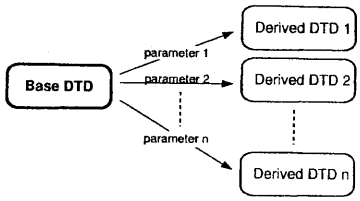

図 4: DTDの導出

3 DTD導出体系

3.1 基本的な考え方

図 4に示すように、共通性の高い基本となるDTD (ベースDTD) に対して、個々の文書型特有の意味分けが必要な部分についてのパラメータを与え、新たなDTD (導出DTD) を機械的に導出できれば、基本となるDTDに変更があってもその変更を導出したそれぞれのDTDに自動的に反映できる。

また導出の際に指定するパラメータを利用して、導出したDTDのインスタンスと、基本となるDTDのインスタンスとを相互変換できれば、文書作成後にアプリケーションに都合のよいDTDを選択できる。どちらのインスタンスも内包する情報自体は等価であるが、共通性の高いエレメントタイプを用いてその属性でエレメントの内容の種類を特定するか、細かくエレメントタイプを分けてエレメントの内容の種類を特定するかの違いがある。

3.2 概要

図 5に示すように、DTD導出体系では次の3種類のDTDを扱う。

中核ベースDTD 一般的な文書を記述するための共通性の高いDTD。

拡張ベースDTD 中核ベースDTDの各エレメントタイプ (の属性定義リスト宣言) に、DTD導出体系で使用するいくつかの属性を加えたDTD。中核ベースDTDとの差異は小さい。

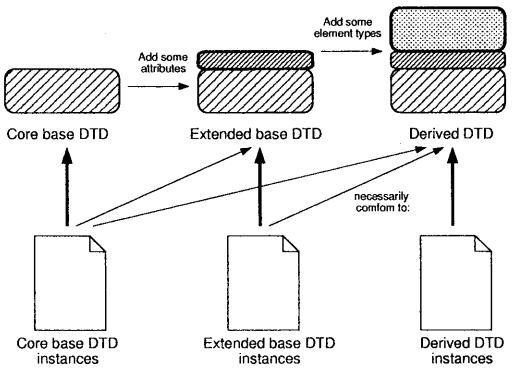

図 5: ベース DTD と導出 DTD との関係

導出 DTD 拡張ベース DTD にユーザが指定したデータから生成した新たなエレメントタイプ (エレメント宣言および 属性定義リスト宣言) を加えた DTD。エレメントタイプを生成して加えることを導出と呼ぶ。

中核ベース DTD と拡張ベース DTD とを区別する必要がない場合は、両者を合わせてベース DTD と呼ぶ。

図 5に示すように、中核ベース DTD のインスタンスは、拡張ベース DTD および導出 DTD にも適合する。拡張ベース DTD のインスタンスは導出 DTD にも適合する。

導出 DTD のインスタンスは、ベース DTD のインスタンスに必ず変換することができる。拡張ベース DTD のインスタンスに変換した場合には、その逆の変換、つまり導出 DTD のインスタンスに戻すことが可能であるが、中核ベース DTD のインスタンスに変換した場合には、逆変換はできない。

新たな文書を作成する場合、ユーザは導出 DTD を用いる。

3.3 詳細

図 6に DTD 導出体系の構成を示す。

DTD 導出体系で使用するデータには、個々の文書型から独立したデータと、個々の文書型に依存するデータの 2 種類がある。個々の文書型から独立したデータは、拡張ベース DTD と DTD 特性指定、個々の文書型に依存したデータは文書構造指定である。

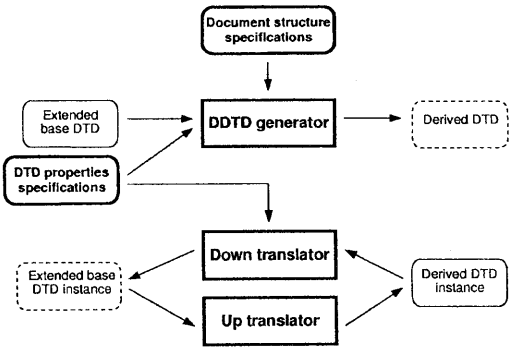

図 6: DTD 導出体系の構成

拡張ベース DTD と DTD 特性指定をいったん作成しておけば、新たな文書型を導出するために作成しなければならないデータは文書構造指定だけである。

3.3.1 拡張ベース DTD

中核ベース DTD としては、階層的な章・節構造をサポートした汎用的な DTD を使用する。たとえば HTML[6], gdoc[3] などの DTD が使用できる。

拡張ベース DTD は、中核ベース DTD に次の事項を追加した DTD である。

- DTD の先頭に "ddtdattr" という名前で内容が空のパラメータエンティティ宣言を追加する。
- 後述する DTD 特性指定で指定するエレメントタイプの属性定義リスト宣言にパラメータエンティティ"ddtdattr" への参照を追加する。すべてのエレメントタイプに対して追加を行なっても構わない。

3.3.2 DTD 特性指定

これはベース DTD の様々な特性を記録したファイルで、行頭のトークンによって、指定する特性を特定する。

RootGI 文書エレメントの要素型名。

DivGI 章・節に相当するエレメントタイプの要素型名。コンマで区切って複数の要素型名を指定できる。

表 1: DTD 特性指定の例

特性名	内容
RootGI	HTML
DivGI	DIV
DivTitleGI	H1,H2,H3,H4,H5,H6
ParagraphGI	P
ParagraphTitleGI	STRONG
ListGI	OL
BodyStartString	<TITLE DRVDGI="BASEINTR"> </TITLE>
BodyEndString	

DivTitleGI 章・節のタイトルに相当するエレメントタイプの要素型名。章・節に相当するエレメントの最初の内容として使用できると仮定する。コンマで区切って複数の要素型名を指定できる。

ParagraphGI 段落に相当するエレメントタイプの要素型名。

ParagraphTitleGI 段落のタイトルに相当するエレメントタイプの要素型名。

ListGI リストに相当するエレメントタイプの要素型名。

BodyStartString 文書エレメントの開始タグと、本文に相当するエレメントの開始タグとの間に置かねばならない内容。

BodyEndString 本文に相当するエレメントの終了タグと、文書エレメントの終了タグとの間に置かねばならない内容。

HTML 3.2 に対する DTD 特性指定の例を表 1に示す。

3.3.3 文書構造指定

特定の文書型の文書構造をインデントを利用して記述したファイルである。文書構造指定は EasyDTD という DTD 作成ツール [7] によく似た構文を持っている。

各行が1つのエレメントタイプに対応し、指定された名前がエレメントタイプ名に相当する。インデントはタブで表現し、タブの数がインデントの深さになる。あるエレメントタイプの内容モデルは、1つ深いインデントの行を続けることで表現する。また出現指示子 (“*”・“?”・“+”) と接続子 (“,”・“|”・“&”) も必要に応じて指定できる。

```
研究会開催通知
    研究会名
    日時
    会場
        会場名
        交通
    議題リスト
        議題+
            発表題目
            発表者名
    備考*
```

図 7: 文書構造指定の例

この記述は、ベース DTD を意識することなく行える。文書構造指定の例を図 7に示す。

3.3.4 導出 DTD

導出 DTD は、拡張ベース DTD の内容すべてと、新たに生成した宣言とからなる。したがって、文書型名として拡張ベース DTD にも含まれているエレメントタイプ名を指定すれば、拡張ベース DTD とまったく同じように使用できる。

導出 DTD で新たに生成されたエレメントタイプは、次の特別な属性を持つ。この属性はインスタンスの変換時に使用する。

BASEGI ベース DTD の対応するエレメントタイプの名前を表す。必ず固定値属性となっており、インスタンス中で値を与えられることはない。これは新たに生成されたエレメントタイプすべてが持つ。

BASEPAD ベース DTD のインスタンスに変換した時に、その要素の最初の内容として出現しなければならない文字列を表す。必ず固定値属性となっており、インスタンス中で値を与えられることはない。この値にエレメントが含まれる場合、そのエレメントは必ず “BASEINTR” という値を持つ属性 “DRVDGI” を持つ。

導出 DTD のインスタンスと、ベース DTD のインスタンスとの関係を図 8に示す。図 8では導出 DTD の属性がインスタンス中で指定されるようになっているが、実際には DTD で固定値属性として宣言される。

導出 DTD の例を図 9に示す。ただし複雑になるので属性宣言の一部は省略してある。

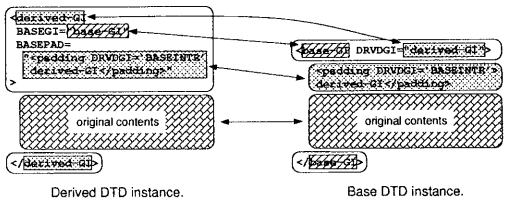

図 8: 導出 DTD インスタンスとベース DTD インスタンスとの関係

```
<!element 研究会開催通知 O O
  (研究会名, 日時, 会場, 議題リスト, 備考*)>
<!attlist 研究会開催通知
  BASEGI NAME #FIXED "HTML">
<!element 研究会名 - O
  (#PCDATA|TT|I|B|U|STRIKE|BIG|...)*>
<!attlist 研究会名
  BASEGI NAME #FIXED "P">
...
```

図 9: 導出 DTD の例

4 評価

DTD 導出体系の特長は、ベース DTD が規定している共通構造の上に、個々の文書型で必要とされる個別の制約を上乗せし、DTD の特殊化を行なえる点にある。ベース DTD を人間が編集して新たな DTD を生成するという従来のやり方に代えて DTD 導出体系を用いることで、次のような利点がある。

- ベース DTD に変更があっても、その変更を個別の DTD に自動的に反映できる。
- DTD 自体の編集を行わなくても済むので、作業時間を短縮できるとともに、作業ミスを防止できる。
- 導出した個別の DTD のインスタンスはベース DTD のインスタンスに自動的に変換できるので、アプリケーションはベース DTD に対する処理パラメータを持つだけで個別の DTD のインスタンスをも処理できる。
- 段落や箇条書き、表などの段落内要素のマーク付け方法はベース DTD と同一であり、文書作成者の習熟が容易である。
- 記入すべき内容がタグとして表示されるので、文書作成者が理解し易い。

DTD 導出体系の拡張を要する点として次の事項が挙げられる。

属性などの宣言 文書構造指定では属性などを定義することができない。ただし属性定義機構を導入する場合、ベース DTD 文書への変換時にエラーを起こす恐れがあるので、対策が必要である。

属性名・属性値の衝突 使用する属性名および属性値は固定しているので、ベース DTD によっては DTD 自身が使用している属性名または属性値と衝突する可能性もある。HyTime[8] のように、使用する属性名および属性値を変更できるような機構が必要である。

5 おわりに

公開されている様々な DTD から、ユーザが必要とする個別の DTD を導出するとともに、導出した DTD のインスタンスを元の DTD に適合するように変換することができる DTD 導出体系を開発した。

今後は DTD 導出体系を実際に利用していきながら、必要なツールの整備と、方式の改良を行なっていく。

References

[1] Charles F. Goldfarb. *The SGML Handbook.* Oxford University Press, New York, 1990.

[2] ISO 12083. *Information and documentation — Electronic manuscript preparation and markup*, 1994.

[3] ISO/IEC TR 9573. *Information Processing — SGML Support Facilities — Techniques for Using SGML*, 1988.

[4] Liora Alschuler. *ABCD... SGML: A User's Guide to Structured Information.* International Thomson Computer Press, Boston, 1995.

[5] Brian E. Travis and Dale C. Waldt. *The SGML Implementation Guide.* Springer-Verlag, New York, 1995.

[6] *Introducing HTML 3.2.* http://www.w3.org/pub/WWW/MarkUp/Wilbur/, 1996.

[7] Norman E. Smith. EasyDTD version 2.3. ftp://ftp.ifi.uio.no/pub/SGML/Demo/easy2-3.tar.z, 1995.

[8] Steven J. DeRose and David G. Durand. *Making Hypermedia Work: A User's Guide to HyTime.* Kluwer Academic Publishers, Boston, 1994.