SGMLのエンティティ機構を利用した漢字表現方法

5S－7

今郷　詔，　奥井康弘

(社) 日本事務機械工業会

1　はじめに

我々が日常使う文字の多くは符号化文字集合の要素であり、対応する符号化表現を用いることで様々なシステム間で文書を交換できる。しかし必要な文字が常に符号化文字集合に含まれているとは限らない。ある特定のシステムでなら、私的な文字を定義して外字という形で表現することは可能であるが、他のシステムと交換することはできない。

また符号化文字集合は字形を定義している訳ではないので、例え符号化文字集合に含まれている文字であっても、特定の字形を交換する必要がある場合にも支障がある。

本稿では、SGML[1] を用いて外字や字形の交換を行なう方法を提案する。

2　SGMLでの文字表現

SGMLでは、文字を表現する手段として符号化表現だけでなく、entityという機構も提供している。この機構を利用して、符号化文字集合に含まれていない文字の交換や、特定の字形の交換が可能である。

2.1　Character entity

文字を表現したentityがcharacter entityであり、character entityの宣言を集めたものがcharacter entity setである。様々なラテン文字やギリシア文字・キリル文字・数学記号・組版記号などがcharacter entity setとしてISOから提供されている[1]。

通常のentityは文字の符号化表現しか参照できないが、特殊なentityであるSDATA entityを用いれば、符号化文字集合に含まれていない文字でも参照できる。

KANJI shape representation method using SGML entity feature.

IMAGO Satosi and OKUI Yasuhiro

Japan Business Machine Makers Association.

2.2　SDATA entity

SDATA entityは例えば次のように宣言する。

```
<!ENTITY agr SDATA "[agr   ]"
          --=small alpha, Greek-->
```

ここで、“agr”がこのentityの名前、“[agr]”が内容、“--”で挟まれた部分がコメントである。

SDATA entityでは通常のentityとは異なり、“[agr]”の部分がシステム依存データとして解釈される。つまり、インスタンス中で“&agr;”としてこのentityが参照された時、その部分が“[agr]”に置換されるのではなく、システムに依存した解釈が行われ、その結果と置換されることになる。この例の場合、“&agr;”に対して“α”という文字を表示するシステムもあれば、“ギリシア文字のアルファ(小文字)”という文字列へのリンクボタンを表示するシステムもあるかもしれない。

3　SGMLを用いた外字・字形の交換

外字といっても、個人や小さなグループが必要に応じて生成する外字と、コンピュータメーカなどがあらかじめ提供している外字の２種類がある。

前者は必要に応じて追加されていくので固定したcharacter entity setを用意することはできない。個々のインスタンスにその文書で使われるentityの宣言を含める必要がある。

後者は対象文字が固定されているので、character entity setとしてあらかじめ用意しておくことができる。どちらの場合も、ビットマップで記述した字形を符号化してcharacter entityの定義に用いればよい。

この手段としてSDATA entityが利用できる。ビットマップをentity宣言に埋め込むには、符号化したビットマップを次のようにリテラルに記述すればよい。

```
<!ENTITY kanji1 SDATA
    "符号化したビットマップデータ">
```

インスタンスでは "&kanji1;" としてこの文字を参照する。本稿で提案するこの記法をサポートするアプリケーションであれば、参照箇所でビットマップを基に字形を再現できる。

ビットマップをリテラルに埋め込むには以下の事項を考慮しなければならない。

3.1 使用できる文字

リテラルの中では、SGML character しか使えないのはもちろんであるが、"%" の後ろに名前開始文字か GRPO("(")、または "&#" の後ろに名前開始文字か数字があると、delimiter と認識されてしまう。また LIT "'" や LITA "" も delimiter として認識されてしまうことがある。したがって、リテラル内では "%&"'" は使えない。

ほとんどの SGML 文書では ISO 646 が使えるような SGML 宣言を使っているので、ビットマップデータを base-64 エンコーディングすれば、使用する文字は

"0-9", "a-z", "A-Z", ".", "/"

だけとなるので上記の制約を満たす。

ただし、元のビットマップデータの 6 ビットを 8 ビットで表現することになるので、サイズは 8/6 倍になる。

3.2 リテラルの長さ

リテラルの長さの上限は concrete syntax の LITLEN で規定されており、reference quantity set では 240 である。この長さを超えなければどの SGML システムでも処理は可能であるが、これを超えると処理できないシステムもある。

32 × 32 のビットマップデータなら 128 バイトなので、base-64 エンコーディングを行なっても 171 バイトで表現できる。

特に複雑な字形を交換する場合を除き、32 × 32 のビットマップを使用することが望ましい。

3.2.1 記法の特定

リテラルの内容から、そのリテラルが本稿の記法に従っていることがわからないと、他の記法の entity との区別が付けられない。そこで、リテラルの先頭に何らかの identifier を入れる必要がある。

これらの delimiter は原理的には concrete syntax で変えられるが、そこまで考慮しなくても実用上は問題ない。

このような場合の記法として formal system identifier が提案されており [3]、それに従っておくことが望ましい。formal system identifier は、リテラルの先頭に記法名を開始タグと同様の構文で書くことになっているので、記法名を "SGSEM" として、次のように記述すればよい。

```
"<SGSEM>符号化したビットマップデータ"
```

3.2.2 Entity 名

SGML は名前空間を 1 つしか持たないので、複数の character entity set を同時に用いると、entity 名が衝突する可能性がある。reference quantity set では名前の長さは 8 文字以下であるが、これではかなりの数になると思われる外字の名前空間をわかりやすく分離することはできない。

名前の長さを 32 文字以内とし、次のような構造とすることが望ましい。

```
owner-identifier.character-name
```

4 まとめ

SGML を用いて外字や字形を交換する方法をまとめると次のようになる。

- SDATA entity として、それぞれの外字や字形を宣言する。
- リテラルの先頭は "<SGSEM>" とし、その後に続けて、字形を 32 × 32 のビットマップで表現して base 64 エンコーディングした文字列を書く。
- entity 名は、owner-identifier.name とし、最大長さ 32 文字とする。

なお本提案は (社) 日本事務機械工業会・実装規約小委員会で検討されてきた方法である。

参考文献

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

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

[3] C. F. Goldfarb. ftp://ftp.ornl.gov/pub/sgml/WG8/HyTime/TC/peanxd3.sgm

"SPREAD glyph shape encoding method" の頭字語。なお SPREAD は本手法を検討しているグループの名前 (Standardization Project Regarding East Asian Documents) である。