2S－1

SGML文書を対象とする文書構造操作言語の提案

高橋 亨　松本 正義
(株) 日立製作所 システム開発研究所

1. はじめに

文書記述言語の国際規格SGML(ISO 8879)[1]の普及に伴い、作成した文書データを単に美しくレイアウトして印刷するだけでなく、文書の構造を利用したさまざまな処理を行うことに関心が集まりつつある。印刷・表示以外の応用としては、例えば文書データの持つ論理構造を組替えて別種の文書を生成する、文書データ中から必要部分を抽出してデータベースに自動投入する、等の処理がある。

SGML自身は文書データの内容と構造を厳密に記述することのみを目的としており、その応用に関しては何も規定していない。特定の応用のみを目的とするシステムでは特にデータ変換を意識する必要はないが、自由度の高い処理系を構築しようとする場合には、SGML文書を応用目的に合わせたさまざまな形態に変換するための手段が必要になる。本稿では、SGML文書を入力として、その構造と表現形式を操作・変換するための言語として報告者らが開発しつつある文書構造操作言語について述べる。(以下、本稿ではこの言語を仮称のÆsopで示す。)

2. 従来の変換処理方式

SGML文書を応用目的に適した形式のデータに変換する方法としてこれまでに実用化されている方式には次のようなものがある。

(1) SGMLパーサが解析結果として出力するデータ(フルタグ形式データ等)をAWK[2]等の文字列処理言語で直接処理する。
(2) タグ(文書を構成する各論理要素の先頭と末尾を示すマーク)に対応して実行すべき処理を記述した対応表を用意しておき、文書中に現れるタグに応じて対応する処理を行う。

一般には後者の方が、より簡潔な指定で同様な変換ができる。後者の方式はSGMLパーサの付属機能として提供されている場合が多い[3]。図1に、タグと変換処理とを対応させた表の簡単な例を示す(これは特定システムについての具体例ではない)。

Proposal of a Document Structure Operating Language for SGML documents
Toru TAKAHASHI, Masayoshi MATSUMOTO
Systems Development Laboratory, Hitachi, Ltd.

```
タグ <LIST>
  --> "\begin{itemize}\n" を出力
タグ </LIST>
  --> "\end{itemize}\n" を出力
```

図 1: 変換処理指定の例

入力SGML文書の構造と出力すべきデータの構造が類似している場合には、上記いずれの方式でも特に問題なく対処できる。しかし、複雑な変換が要求されると、従来方式では次のような問題が生じてくる。

(1) 同一種別の論理要素に対して、要素の持つ属性値や要素の出現文脈（上位要素の種別や同位要素間での並び順など）に応じて処理を変更することが難しい。
(2) 要素の出現順序の入れ換えや不要な部分構造の削除など、文書の持つ構造自体を変更する操作の実現が難しい。

なお、文字列処理言語を用いる方式ではこれらの処理も原理的には可能であるが、そのプログラムは複雑で理解しにくいものになってしまう。

3. Æsopのねらいと特徴

Æsopのねらいは、上記のような問題点を解決し、複雑な変換処理をできるだけ平易に記述できるようにすることにある。このため、Æsopは次のような特徴を備える。

(1) SGMLパーサが生成する解析済み文書データ(構文解析を終了し、SGML文書上では省略されていた情報等がすべて展開された木構造データ)を入力として動作する。
(2) 解析済み文書データの木構造を深さ優先の順序で辿りつつ、その各ノード上で適切な処理手続き(スクリプト)の選択・実行を繰り返すことを最も基本的な動作手順とする。この基本動作を木構造データのトラバースと呼ぶ。トラバースは暗黙のうちに行われるので、明示的に指示する必要はない。図2に文書インスタンスの一例を、図3にそのインスタンスに対応する解析済み文書データの構造とトラバース順序を示す。

```
.....
<A a1="xx">
  <B> ..... </B>
  <C> ..... </C>
</A>
.....
```

図 2: 文書インスタンスの一例 (部分)

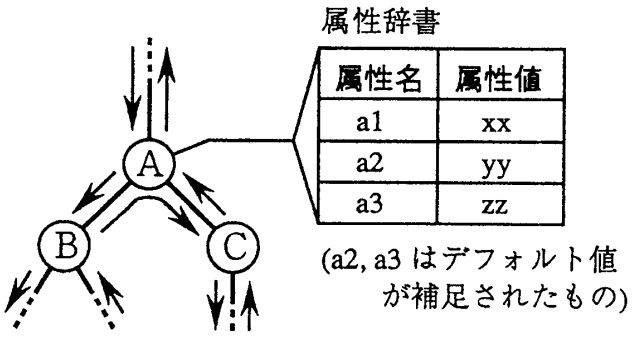

図 3: 解析済み文書データとそのトラバース

(3) スクリプトの選択を行う際には、注目ノードの持つ属性、ノードの木構造中における位置 (文脈) 等の情報を選択条件中に用いることができる。なお、これらの情報はスクリプト本体内部でも利用できる。

(4) 各スクリプト本体には、適用対象ノードの先頭 (開始タグ位置) で実行すべき処理 (プロローグ)、下位ノード群についてトラバースを続行するかどうかの指定 (デフォルトは続行)、対象ノードの末尾 (終了タグ位置) で実行すべき処理 (エピローグ) のそれぞれを記述する。プロローグおよびエピローグの内容は通常のプログラミング言語と同様、手続き的に記述する。

(5) 指定ノードの持つ下位ノード群に関して、その順序の入れ換えや追加・削除を行い、解析済み文書データの木構造を組み替える手段を備える。

(6) 複雑な変換処理をいくつかのステップ (プロセス) に分割し、あるプロセスの結果として得られる木構造データを次のプロセスへの入力とするパイプライン処理を行う手段を備える。

なお、高度な文書構造変換に関する先行例としては [4]、[5] 等がある。[4] では文書構造操作用のオペレータ群を備えた手続き型言語の実装と評価を行っているが、本提案におけるトラバース処理やパイプライン処理は採用されていない。[5] では属性文法を利用した変換方式の検討を行っているが、具体的な処理方式の提案には至っていない。

4. Æsop プログラムの例

上記のような特徴を利用した Æsop プログラムの一例 (部分) を図 4に示す。これは、NOTE 型要素の直下に位置する IT 型要素に対してのみ適用されるスクリプトである。このスクリプトのプロローグでは、上位要素 (NOTE) の属性 REQ の値が "YES" である場合には "備考 n" を含む文字列、それ以外の場合には "参考 n" を含む文字列を出力することを指示している。ここで、n はこの要素の NOTE 内での出現順位を示す数値である。

```
script ELEMENT
    IT (gi(parent(self))=@NOTE)
{
    prolog {
        local n; n:=occ_ord(self);
        if (get_attr(parent(self),@REQ)="YES")
            printf("\\item[備考 %d] ", n);
        else
            printf("\\item[参考 %d] ", n);
    }
    epilog { printf("\\n"); }
}
```

図 4: Æsop プログラムの一例 (部分)

5. おわりに

SGML 文書を構造や表現形式の異なるデータに変換するための文書構造操作言語 (仮称:Æsop) を提案した。

Æsop は文書データの木構造を自動的に辿り、要素の出現文脈や属性に応じて異なる処理を行う機能、木構造の組み替えを行う機能、変換処理を複数のプロセスに分割して段階的に行う機能等を備え、複雑な変換処理を簡潔に記述することができる。Æsop は現在言語仕様の設計をほぼ終了しており、今後は実装と評価を進めていく予定である。

参考文献

[1] ISO 8879, Standard Generalized Markup Language (SGML). ISO, 1986

[2] A.V.Aho, 他. プログラミング言語 AWK. トッパン, 1989.

[3] E.Herwijnen. 実践 SGML. 日本規格協会, 1992.

[4] G.E.Blake, 他. Shortening the OED: Experience with a Grammar-Defined Database. ACM Trans. on Info. Sys., Vol.10, No.3, 1992.

[5] 今郷, 他. SGML インスタンスの変換方式の検討. 情処 47 回全国大会, 1993.