SGML文書の変換・再利用のための言語"AEsop"

高橋 亨
東野 純一

"AEsop" -- A Language for Conversion and Reuse of SGML Documents

Toru Takahashi
Jun-ichi Higashino

To construct a general purpose SGML-based document processing system, it is required to translate the document structure and/or the data representation, from a source document written in SGML, to the format required by the application. In this paper, We propose a language to do this translation, whose processor executes translating procedures while traversing the tree structure of the parsed document produced by an SGML parser. This language (currently called "AEsop") is an general-purpose document structure operating language, which can handle any DTD and any output data format. For example, AEsop has the ability to: (a) select a procedure to be applied to a node, according to the conditions of the node (the element type name, attribute values, the position in the document structure, etc.), (b) transform the tree structure of the document. We finished the language design, and now implementing a prototype processor.

1．はじめに

文書記述言語の国際規格SGML[1]が普及しつつある。文書記述にSGMLを用いることによって得られる最大の利点は、文書の持つべき論理的構造(DTD: Document Type Definition)を明確に定義し、その構造に従って厳密に文書内容を記述できる点にある。これにより、文書の原データを処理系や機種に依存しない形で作成し、これを異なるシステム間での交換や使用システムの変遷に左右されない永続的な保存のために用いることができる。その反面、SGML自身は作成した文書の応用については何も規定していないので、SGML文書を入力として何らかの具体的な処理（組み版・印刷、CD-ROMやネットワークを介した電子出版、文書データのデータベースへの投入等）を行おうとした場合、処理系側が必要とする形式に合わせてデータ変換を行う必要がある。

このデータ変換には単にデータの表現形式を変更するだけで済む単純な場合もあれば、文書の構造自体を大幅に組み替えなければならない場合もある。特定の応用のみを目的とするシステムであれば処理系内部で目的に合わせた専用の変換処理を行えばよいが、作成したSGML文書をさまざまな目的に再利用するためには、自由度の高い汎用の変換ツールを提供することが重要となる。

本稿では、任意のSGML文書を入力とし、その構造と表現形式を操作・変換するための言語である文書構造操作言語AEsop (仮称) について、その機能と特長を報告する。

2．従来の変換処理方式とその問題点

入力SGML文書を変換し、応用目的に適した形式のデータを生成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) 解析結果を一定の形式（フルタグSGML形式等）で出力する機能を備えたSGMLパーサと、AWK[2]等の文字列処理言語とを組み合わせて変換を行う。
(2) SGML文書中のタグ（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宣言的に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。

前者の方式は原始的ではあるが簡単な変換処理ならば十分に可能である。後者の方式はSGMLパーサの附属機能として提供されている場合が多く、こちらのほうが一般的にはより簡潔な指定で変換ができる[3]。図１に、後者の方式における変換処理指定の簡単な例を示す。（注：これは特定システムについての具体例ではない。）

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

図１　簡単な変換処理指定の例

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ついて、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）に応じて処理を切り分けることが難しい。
(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、パーサ出力を文字列処理言語で処理する前者の方式ではこうした処理も原理的には可能であるが、そのプログラムはかなり複雑となり、作成・保守ともに困難なものとなってしまう。

3．AEsopのねらいと特長

AEsopの設計に際しては、上記のような問題点を解決し、複雑な変換処理であっても平

易に記述できるようにすることを目標とした。このため、AEsopには次のような特長を持たせた。

(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。すなわち、AEsopの言語プロセッサはSGMLパーサの後処理系として働く。

(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、プログラム上で指示する必要はない。図２に文書インスタンスの一例(部分)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)

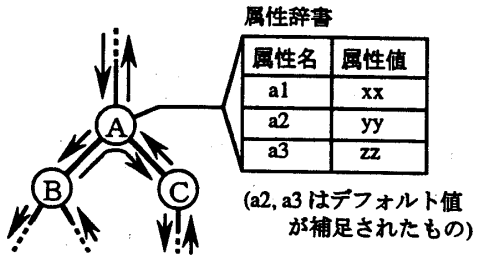

図３ 解析済み文書データとそのトラバース

(3) トラバース処理の過程において注目ノードに対して適用すべきスクリプトを選択する際には、そのノードの種別(要素、文字列データ、処理命令、他)、ノードが持つ属性、木構造中におけるノードの位置(文脈)等の情報を組み合わせて選択条件を構成することができる。なお、これらの情報はスクリプト本体の内部からも利用できる。

(4) スクリプトは大別して要素ノードを適用対象とする要素スクリプトと、文字列データを適用対象とするデータスクリプトに分けられる。要素スクリプトの本体には、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。データスクリプトの本体には対象文字列データの置換や出力等の処理を記述する。プロローグ、エピローグおよびデータスクリプト本体の内部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) 指定した条件を満たすノードを木構造中で検索する機能、および指定ノードの持つ下位ノード群について順序の入れ換えや追加、削除を行い、木構造そのものを組み替える機能を提供する。

(6) 複雑な構造変換が必要となる場合、これをいくつかのステップ（プロセス）に分割し、あるプロセスを実行した結果として得られる木構造データを次のプロセスに入力として与えるパイプライン処理を行う機能を提供する。

なお、構造変換を伴う高度な変換処理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、この言語では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て変換仕様を指定する方式の検討を行っているが、具体的な方式提案には至っていない。この他、SGML文書の構造変換と組み版スタイルを指定するための言語であるDSSSL[6]がISO規格(案)として提案されている。しかし、DSSSLの構造変換ではSGMLから(他のDTDに従う)SGMLへの変換しか扱わず、パイプライン処理も直接にはサポートされていない。また変換結果として生成されるSGML文書についてもDTDを与えなければならないことなどから、変換仕様の記述はかなり煩雑なものとなる。

4．AEsopプログラムの例

上記のような特長を利用したAEsopプログラムの一例を図4に示す。これはSGML文書をLaTeX[7]に変換して印刷出力を得るためのAEsopプログラムから、脚注の処理に関係する部分のみを抜き出したものである。

LaTeXの標準的なスタイルでは、脚注は「\footnote{..脚注の内容..}」というように脚注生成のためのコマンドと脚注の内容とを組み合わせて記述し、この\footnoteコマンド自体を文章中（脚注を参照している位置）に埋め込んで使用する。これに対して、変換元のSGML文書では、脚注本体と脚注への参照とを分離し、文章中には内容を持たない脚注参照（例えば「<fnref refid=fn1>」）だけを埋め込み、文書内の別の場所にある脚注本体（例えば「<fnbody id=fn1>..脚注の内容..</fnbody>」）をSGMLのID/IDREF機能を用いて参照するというアプローチをとっているものとする。

この場合、生成すべきデータの構造は元のSGML文書とは大きく異なるので、構造変換を伴わずに直接出力データを生成するのは困難である。図4に示したAEsopプログラムでは、変換処理を二つのプロセスに分割し、最初のプロセスで文章中に埋めこまれているすべての脚注参照をそれぞれ参照先の脚注本体によって置き換えるという構造変換を行っている。この結果、文書データの木構造は目的とするLaTeXデータのそれと実質的に等価なものとなるため、次のプロセスでは単純に表現形式を変換して出力データを生成する処理を行うだけで済む。このようにして、簡潔なプログラム記述により複雑な変換処理が記述できる。

5．おわりに

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

```
process 構造変換
{
  script ELEMENT FNREF          # 脚注参照の置き換え
  {
    prolog {
      var fn_body;
      # 参照先の脚注本体を検索
      fn_body := id_search(get_attr(self, @REFID));
      # 脚注本体を親から切り離す
      del_child(parent(fn_body), fn_body);
      # 脚注参照を脚注本体で置き換える
      repla_child(parent(self), self, fn_body);
    }
  }
}

process LaTeXデータ出力
{
  .....
  # (文章中に埋め込まれた)脚注本体の処理
  script ELEMENT FNBODY
  {
    # "\footnote{..}"コマンドの出力
    prolog { puts("\\footnote{"); }
    epilog { puts("} "); }
  }
  .....
  # 文字データ内容の変換と出力
  script CDATA
  {
    # 特殊文字の置き換えを行いつつ文字データ内容を出力する
    puts( conv_string(self, latex_conv_table) );
  }
}
```

図 4 AEsop プログラムの一例（部分）

AEsop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変換を行う機能等を備え、複雑な変換処理を簡潔に記述することができる。AEsopは既に言語仕様の設計を終了しており、今後は処理系の実装と実際的な問題を対象とした試用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. 情報処理学会第47回全国大会, 1993.
[6] ISO/IEC 10179, Document Style Semantics and Specification Language (DSSSL). ISO, 1995.
[7] 奥村. LaTeX美文書作成入門. 技術評論社, 1991.

(株)日立製作所 情報・通信開発本部　　Information Systems and R&D Division, Hitachi,Ltd.