文書の論理構造の変換におけるデータ形式の指定法

3S－4

酒井 乃里子　高須 淳宏　安達 淳
東京大学工学部　学術情報センター研究開発部

1 はじめに

SGML (Standard Generalized Markup Language; ISO 8879) [1] に従って、論理構造を明示的に記述する文書の作成は、科学技術論文ではごく一般的になっている。しかし、SGMLでは論理構造は必要に応じて任意に定める余地があるので、学会・出版社などごとに多様な論理構造が存在し、それらの文書を統合して扱う全文データベースを構築する際には、論理構造の多様性を吸収する機構が求められる。

本研究ではこのような状況を踏まえて、論理構造の変換手法を提案している。本研究の特徴は、HTML (HyperText Markup Language) など特定の論理構造ではなく、任意の論理構造間の変換を目的とする点である。これにより、データベース側も自由に論理構造を設定し、それに基づいた検索や閲覧を実現できる。

本稿では、特に変換処理におけるデータ形式の指定法とその処理に焦点をあてる。以下本稿の構成は、第2節で関連する研究に触れたあと、第3節で提案するシステムの全体像を説明し、そのあと第4節で本システムにおけるデータ形式の指定法ついて述べる。

2 論理構造の変換についての関連研究

SGML文書の論理構造の変換に関しては、論理構造の定義 (DTD; Document Type Definition) に書き込む、あるいはテキスト中のマークアップタグが出現するタイミングに対して、アクションを定義する方法が提案されている[2][3]。しかしこのような手法は理解しやすいものの、要素を組み合わせるといった複雑な変換を行うのは困難である。

一方で、論理構造の変換仕様をプログラムのように記述する言語も提案されており[4]、柔軟で高度な変換が実現している。しかし、記述がプログラムソースと同程度の複雑さを持つために、ユーザには能力や経験が要求され、使用にあたっての敷居は高い。

3 本手法による論理構造の変換法

3.1 論理構造の差異

変換前後の文書[3]の論理構造の差異には、要素の取捨（図1の3.）、要素名の違い（0. とc.）、繰り返しをまとめる（4., 8 とb.）、新旧要素が1対多（1., 2. とa.）、などがある。(図1。左右各論理構造の左端の数字・英数字は説明のための記号)

本手法では、各新要素に、それを構成する旧要素を割り当てることで変換動作の仕様とする（図2、左端の数字は説明のための番号)。仕様は、学会などの論理構造と、データベースの用いる論理構造との間で、予め一度作成する。

3.2 本研究での文書の表現

仕様において要素は、同名でも違う文脈（上位の要素下）で出現するものを区別するために、論理構造木上の「絶対パス」で表現することを原則とし、本研究では、パス表現されたものを要素と呼ぶ。また、パスを構成する、論理構造木上の中間ノードをコンポーネントと呼ぶ。図中でコンポーネントは、名前と識別番号 (ID) で表されている。

3.3 変換仕様書の記述と変換動作

変換仕様の記述は、新旧要素が1対1が1対多かによって二通りある。1対1の場合、各新要素について、新要素自身と、割り当てられた旧要素を、";" で区切って順に記す。たとえば、図2の0. によれば、新要素 "/recordlist/record/subject" は、旧要素 "/Recordlist/Record/Title" で構成される。

一方、1対多の場合（組み合わせのある割り当てと呼ぶ）の仕様は、各新要素について、新要素自身と、割り当てられた旧要素群を、";" で区切って順に記す。さらに旧要素群は、パスを共通部分と残りの部分（複数）に分けて、":" で区切って順に記述する。たとえば図2の5. によれば、新要素 "/recordlist/record/author" を構成する旧要素は、パスの共通部分が "/Recordlist/Record/Author/" で、残りの部分が "Surname" のものと "Firstname" のもの、つまり "/Recordlist/Record/Author/Surname" と "/Recordlist/Record/Author/Firstname" である。

要素名以降は、新要素を構成する旧要素（群）が整形される形式を指定する部分であり、第4節で説明する。

コンポーネント名の前にある "!" は、旧要素の繰り返しや組み合わせの判断や、新要素の論理構造上の位置の決定で用いるが、本稿では詳しくは触れない。

この仕様書を用いて、本手法の論理構造の変換動作は、1. 旧文書から、必要な要素を抽出し、2. 組み合わせや繰り返しなどの整理を行い、3. 新要素を設定して、新文書を受理する、という段階からなる。次節では、2. の段階で行われる、データの整形を説明する。

4 データ形式の指定法

4.1 データ整形の種類

論理構造の変換で必要なデータ整形は、1つは個々の旧要素の形式である。たとえば、「すべて小文字で表現したい（図1の9. とd.)」といったものである。

次に、組み合わせのある場合、つまり1つの新要素を複数の旧要素が構成する場合は、旧要素群の組み合わせ方を定めなくてはならない。たとえば著者名について、「もとの文書では姓と名とが別の要素だったが、これを1つの要素内で、"姓，名前の頭文字" という形式にしたい（12., 13. とf.)」という要求である。

Specification of Data Format in Transforming Logical Structures of Documents
Noriko SAKAI , Atsuhiro TAKASU , Jun ADACHI
Faculty of Engineering, The University of Tokyo
Research & Development Department, National Center for Science Information Systems

本稿では、変換前後の文書などをそれぞれ「旧」「新」と呼ぶ。

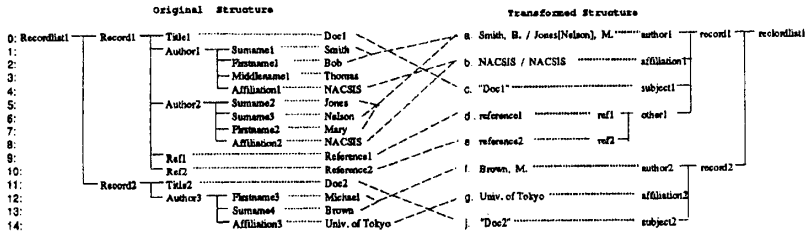

図 1: 論理構造の差異と変換の例

```
   # 変換仕様書
   #1. 新旧要素が 1 対 1 の場合
0. /recordlist /record !/subject; /Recordlist /Record !/Title; ''$1''; $$ ($1)
1. /recordlist /record !/affiliation; /Recordlist /Record !/Author !/Affiliation; $1; $$ / $1
2. /recordlist /record !/other /ref; /Recordlist /Record /Ref; &lowercase($1); $$, $1
3. /recordlist /record /para; /Recordlist /Record !/Body /Para; $1; $$ $1
4. /recordlist /record /para /fig; /Recordlist /Record !/Body /Para /Fig; $1; $1

   #2. 新旧要素が 1 対多の場合
5. /recordlist /record !/author; /Recordlist /Record !/Author/: Surname: Firstname; $1: &initial($1);
                                    $$ [$1]: $$ [$1]; $1, $2; $$ / $1
```

図 2: 論理構造の変換仕様の例

また、割り当ては 1 対 1 であっても、その旧要素が複数回現れる場合（繰り返し）で、新文書では複数個にできない場合、つまり論理構造の定義上繰り返しが許されていなければ、1 つの新要素にまとめなくてはならない。たとえば、「もとの文書で複数回現れるたら、"/" で区切ってつなげて 1 つの要素にしたい（4., 8. と b.)」という要求である。繰り返しは、1 対 1 の場合以外に、組み合わせたものにも起こり得る。

繰り返しは、組み合わせる要素にも起こる。たとえば、新要素 "/recordlist/record/author" は 2 つの旧要素群 "/Recordlist/Record/Author/Sirname" と "/Recordlist/Record/Author/Firstname" で構成されるが、これらは複数回現れ得る（5., 6.)。これらは、組み合わせる前にまとめてしまう。

以上から、必要な形式の指定は次の通りになる。

- 1 対 1 の場合
  - 個々の旧要素の形式
  - 繰り返される場合の形式
- 組み合わせのある場合
  - 個々の旧要素の形式
  - 個々の旧要素が繰り返される場合の形式
  - 組み合わせる形式
  - 組み合わせたものが繰り返される場合の形式

4.2 データ形式の仕様記述と動作

仕様書における形式の指定で、データを表す変数には yacc と同様の記法を採用する。つまり、それまでに作られたデータを "$$"、新しいデータを "$1, $2" などと表す。また "&" から始まる記述、たとえば "&initial()" は、予め用意した処理関数にデータを引数として渡し、その結果を用いることを意味する。これにより、「頭文字のみ」などの処理を実現できる。

1 対 1 の場合には、上記の 2 つの項目を、";" で区切って順に記述する。たとえば図 2 の 0. では、個々の旧要素の形式としては二重引用符 '' で挟み、繰り返された場合には、新しく加わった方を括弧に入れて後ろに並べる、という指定を表している。

組み合わせがある場合には、上記の 4 つの項目を ";" で区切って順に記述し、さらに個々の旧要素とその繰り返しの形式の項目中では、個々の旧要素に関する指定を ":" で区切って記述する。このときの記述順や記述中のデータの変数（"$1" など）は、旧要素名を記述した部分の出現順に準拠する。たとえば、5. では個々の旧要素の形式は "Surname" はそのまま、"Firstname" は関数 "initial()" にデータを与えた結果とし、個々の旧要素が繰り返される場合には、両方とも、新しいデータの個々の整形結果を "[]'" で挟んでつなげる形式で、組み合わせ方は "Surname のデータ, Firstname のデータ" という形式、さらに組み合わせたものが繰り返される場合には、"/" で区切って後ろに並べる、という指定を表している。

以上の指定を用いる整形処理は、記述に従って、"$" から始まる部分は該当するデータに置き換え、"&" から始まる部分は該当関数を起動した結果に置き換え、それ以外は文字列置換で、データを作成すればよい。

5 まとめ

本研究では、多様な論理構造を持つ文書を統合するデータベースなどを実現するための、論理構造の変換手法を検討しており、本稿では変換における、データ形式の指定法とその処理法を提案した。本手法を用いることで、多種かつ多量な文献データを扱い、しかも管理者などの主義に適合する全文データベースが比較的容易に構築できるものと期待できる。

データの整形部は現在実装されており、システム全体を作業中である。また今後、より実際に近いデータによる実験を行いたい。

参考文献

[1] van Herwijnen, E. 著, SGML 懇談会実用化 WG 訳: "実践 SGML," 日本規格協会 (1992).

[2] Price, L. A., and Schneider, J.: "Evolution of an SGML Application Generator," ACM Conference on Document Processing Systems (1988), pp. 51–60.

[3] Warmer, J., and van Vliet, H.: "Processing SGML Documents," Electronic Publishing, Vol.4/No.1 (March 1991), pp. 3–26.

[4] 高橋, 東野: "SGML 文書の変換・再利用のための言語 'AESOP'," 情報知識学会第 3 回研究報告会講演論文集 (May 1995).