6Q−4

文書処理統合環境DIETにおける文書データ処理言語

長島 正明　山川 正
キヤノン(株) 情報システム研究所

1. はじめに

ワープロやDTPの発展により[1]文書の電子化が進み，文書データの利用形態が多様化してきている．我々は，利用形態に応じて文書処理システムを構築できる環境として，文書処理統合環境DIET(Document Integration Environment and Tools) の研究を行なっている[2][3][4]．

DIETでは，SGML[5]を基本にして，文書の論理構造を付加した記述形式(DIET形式)を提唱した[3][4]．今回は，DIET形式で記述された文書データ(DIETary)の加工について述べる．

2. DIETary加工用言語DPLの検討

2.1 背景

DIETでは，構造化文書データであるDIETaryを処理対象とし，文書データの加工を行なう．ここで，DIETaryの加工とは，DIETaryから別な論理構造を持つ文書データ(DIETaryも含む)，あるいはスタイル構造を持つ文書データへの変換のことである．

DIETaryの加工法として，C言語等の純粋なプログラミング言語で，そのコンバータを作成することが考えられる．しかし，この方法は加工ツールの作成が一般的に難しく，「手軽に加工できる」わけではない．また，awk等のテキスト処理用の言語を利用することも考えられるが，しかし，awkは，レコード単位の1次元リストを処理対象としているため，DIETaryのような構造化文書データを取り扱うのに，十分ではない．

他に，構造化文書データを取り扱う処理系として，SGMLで記述した文書データを扱うMARKUPというシステムが提唱されている[6]．これは，SGMLデータをパーザで解析し，マッピング記述に従って加工処理を行なう．マッピング記述には，文書要素の始まりと終りを検出した時に挿入される文字列，あるいは，その時実行される手続きのC関数を記述する．従って，文字列変換のみの単純な加工の時はよいが，文字列に対して何らかの処理を行ないたい場合，C言語で記述することになり，その記述は難しくなる．

A Document Processing Language in the Document Integration Environment and Tools (DIET)
Masaaki NAGASHIMA, Tadashi YAMAKAWA
Information Systems Research Center, Canon Inc.

2.2 目的

従来の文書データを加工するための処理系における構造化文書データを扱う時の問題点として，以下のことが挙げられる．

- 階層レベルに応じた処理の記述が容易でない．
- 上位文書要素に応じた処理の切り替えの記述が容易でない．
- 文書要素処理終了時における上位文書要素処理の再起動の記述が必要である．
- 同じ文書要素における開始タグ，終了タグでの処理記述の場所が離れる．

そこで我々は，上の問題点を解決しつつ，「必要な時に手軽にDIETary加工ツールを作成する」ことを可能にするための言語として，DPL(DIET Processing Language)を考案した．

2.3 DPLの処理モデル

DPLでは，上に述べた問題点を回避し，簡潔に処理記述ができるようにするため，以下のような方針をとった．

- 文書要素の指定に，階層構造を記述できるようにする．
- 文書要素の階層構造による処理の切り替え機構を，処理系に委ねる．
- 文書要素ごとに，開始処理，内容処理，終了処理を記述する．

上に述べた方針に従い，構造化文書データに対し，次のような取り扱いをすることにした．

- 文書要素ごとにその内容の加工方法を記述するものとする．
- 処理単位はデータチャンクである．
- データチャンクは複合文書要素で区切られる文字列とする．
- 複合文書要素の処理結果であるリターン値もデータチャンクとして扱う．

ここで，複合文書要素とは開始タグと終了タグを含む文書データのことである．

例えば，図1の(1)にあるDIETaryの複合文書要素Aを処理することを考える．データチャンクは，複合文書要素Bで区切られた文字列αとγ，そして，複合文書要素Bの処理からのリターン値δである．文書要素がタグを

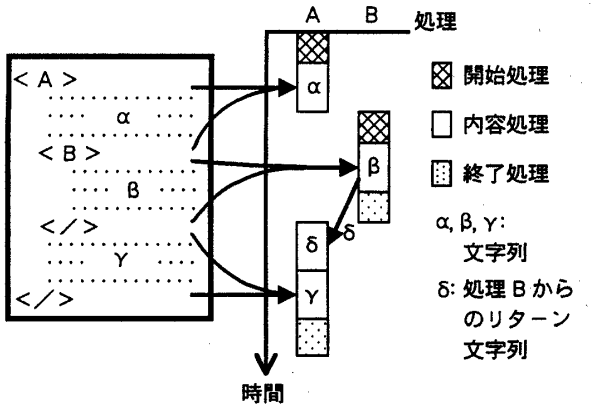

図 1 DPLの処理モデル

含まない基本文書要素の時は，そのまま文字列を処理する(図1(2)のαとγの処理)．複合文書要素の時は，一旦，処理をその複合文書要素の処理に委ね(図1(2)の処理B)，そのリターン値を処理Aで処理する(図1(2)のδの処理)．

2.4 DPLの記述例

DPLでは，各複合文書要素ごとに，その中の文書要素に対する処理を記述する．そこで，一つの複合文書要素に対し，

/文書要素名/{ 開始処理}{ 内容処理}{ 終了処理}

と記述する．文書要素名の記述の箇所は，階層構造も記述できるようになっている．例えば，「YYYの下にあるXXX」，「最上位にあるXXX」等の表現も可能である．また逆に，階層構造を意識しない「すべてのXXX」等の表現も可能である．

図2(3)にDPLの記述例を示す．比較のため，従来の文書データ加工例として，awkで記述した例を図2(4)に示す．これらは，図2(1)のDIeTaryから図2(2)のLaTeXのデータへ変換する例である．上位の文書要素に応じて処理を切り替えるなど，DPLでの記述は，awkでの記述に比べ，文書要素ごとに簡潔に処理記述ができる．

```
<header>...<emphasize>ABC</>...</>
<par>...<emphasize>abc</>...</>
```

(1) 入力DIeTary

```
\title{...\underline{ABC}...}
...{\em abc}...\par
```

(2) LaTeX データ

```
/heder:emphasize/     # headerの下のemphasize
    {write_str("\\underline{");}
    {write_str($0);}
    {write_str("}");}
/par:emphasize/       # parの下のemphasize
    {write_str("{\\em ");}
    {write_str($0);
    {write_str("}");}
/header/
    {..開始処理..}{..内容処理..}{..終了処理..}
/par/
    {..開始処理..}{..内容処理..}{..終了処理..}
```

(3) DPLでの記述

```
BEGIN{level=0
      RS=...
}
/^emphasize/{
  level++
  stack[level]="emphasize"
  if (stack[level-1] == "header")
    printf "\\underline{"
  else if (stack[level-1] == "par")
    printf "{\\em "
  for (i=2; i <= NF; i++)
    printf "%s",$i
  printf "}"
}
/^header/{...contentの処理...}
/^par/{...contentの処理...}
/^\//{
  if (stack[level] == "header")
    printf "}\n"
  else if (stack[level] == "par")
    printf "\\par\n"
  level--
  if (stack[level] == "header")
    ...header contentの処理...
  else if (stack[level] == "par")
    ...par contentの処理...
}
```

(4) awkでの記述

図 2 DPLの記述例

3. おわりに

今回は，DIeTaryを加工するための言語DPLを考案した．DPLでは，文書要素ごとに処理を切り替え，各処理の取り扱うデータを文字列として取り扱うことにより，構造化文書データであるDIeTaryの加工処理を容易に記述することが可能になる．

現在，DPL処理系のプロトタイプを作成している．今後は，プロトタイプを用いて，DPLの有効性の実証と，細部仕様の検討を行なう予定である．

参 考 文 献

1) “ミニ特集: デスクトップパブリッシング”, 計測と制御, Vol.28, No.3 (1989).
2) 山川,川端,田村: “キヤノンEoDs計画とその統合文書処理環境”, 情報処理学会卓上出版シンポジウム報告集, pp.223-232 (1988.7).
3) 川端,山川,出井,田村: “文書処理ワークステーションと文書アーキテクチャ”, 電子通信学会ワークショップ — 電子出版の現状と課題, pp.53-59 (1989.4).
4) 山川,川端,田村: “OA業界から見たDTP”, 情報処理, Vol.31, No.11, pp.1508-1517 (1990).
5) ISO 8879: Information Processing - Text and Office Systems - Standard Generalized Markup Language(SGML) (1986).
6) Lynne A. Price, Joe Schneider: “Evolution of SGML Application Generator”, ACM CONFERENCE ON DOCUMENT PROCESSING SYSTEMS, pp.51-60 (1988).