マイクロコンピュータ 42－2
（1986・11・21）

事務文書体系の文字内容体系向けの一解釈系の考察

若鳥陸夫
（日本ユニバック）

「開放型システム間相互接続（ＯＳＩ）」の応用層に位置付けされる「メッセージ指向文章交換系（ＭＯＴＩＳ）」や，その封筒部に入れて交換される「事務文書体系（ＯＤＡ）」の国際規格の開発が盛んである。本稿は，事務文書体系の中の「文字内容体系（character content architecture）」の機能符号群の解釈系の高速化についての検討報告である。

A high speed analysis of function codes in a content architecure used on office document architecture.

Rikuo WAKATORI
Nippon Univac Kaisha Ltd.

An experiment has been done on functional analysis of content architecture which is used for office document interchange.
Three methods were compared a speed point of view. These are
a) distributed computation during receiving,
b) distributed computation using a stack,
c) transfer from concatenated string to a value,
The result shows that distributed computation during receiving is the best selection.

1　課題の背景

事務文書体系(Office document architecture)は，大別して，論理構造と割付け構造がある。論理構造は，「文書論理根」－「複合論理対象体」－「論理対象体」の構造となっており，論理対象体から文字内容体系・幾何学的図形内容体系・ラスタ図形内容体系などにポインタが付けられている。割付け構造は，「文書割付け根」－「ページ集合」－「ページ」－「枠」－「区画」－「内容部」の構成をしており，割付け処理の際に内容部が割付け規則群により自動割付けされる(図１)。

事務文書体系による文書交換は，「事務文書交換様式(ＯＤＩＦ)」により行なわれる。その符号化法は，「抽象構文記法１(略称ＡＳＮ．１)」により行なわれ，「識別子」・「長さ記述子」・「内容」となり，内容はまた抽象構文記法１の入れ子構造となる。この符号化法により，文書の論理構造や割付け構造を交換する。

しかし，内容部の符号化法は，「内容体系」の種類により変化するので，事務文書交換様式内では抽象構文記法１による符号化法の他に，文字内容体系の符号化法も存在する。この文字内容体系を，受信中に監視するためには，その符号化法をできるだけ速く解釈・実行することが望ましい。

文字内容の最上位水準の機能符号群は，約３０種類あり，このうち，値を付属している機能符号群の解釈時間をできるだけ少なくしたいため，その変換方法を考察した(図２)。

2　文字列から数値への変換方式

文字列$P_1..Pn$を，数値に変換する手法を考える。
方法を大別すると，受信中に並行解析する方法と受信後に一括して変換する方法がある。変換速度が速ければ前者をとることができる。この事前判定のため３つの方法を検討してみた。

2.1　分配計算法(仮称)

文字列$P_1..Pn$を，１字読み込む毎に数値に変換して貯えておき，終端符号を検出した時の値が，その文字列の値とする。この方法は，解釈に要する時間が，文字列の到来時間間隔よりも，明らかに少ない場合に適用できる。
この場合のｎ行の計算式は，次のようになる。

$$10(\cdots(10(10P_1+P_2)+P_3)\cdots)+Pn \quad \cdots\cdots\cdots\cdots(1)$$

ここでは$P_1\cdots Pn$は，最上位桁数値をP_1，最下位桁数値をPnとするｎ桁の数値とする。

2.2 スタックによる分配計算（仮称）

後入れ先出しのスタックを考え，式１の積・和ごとにスタックに記憶させ，文字列P_1‥Pnが読み取られた直後に，その文字列は数値に変換されているようにする。この方法は，金物イメージのプログラムとなるので高水準言語で記述しても，アセンブラ言語によるプログラムの感じがする。

2.3 文字列貯蔵による一括変換法（仮称）

各々の文字P_1‥Pnを，受信時に，一つの文字列に結合させ，全ての文字を結合した後で，数値変換関数により数値に変換する。この方法は，処理系組込関数の有無により，実現が容易か少しやっかいになるか分かれる。

3 実験方法

文字から数値への変換部分を抽出して，図３のPascalプログラムにより，Univac UP10E50で実行し各方式の処理時間を計測した。

各ケースは，前述の変換方式に対応する。

試験データは，100桁とし，その各桁に0-9までの数字を入れた配列Cを準備した。

また，時間倍率は1000とした。

4 実験結果と評価

実験結果を図４に示す。その結果，ケース１（分配法）が35秒（1文字当たり0.35ミリ秒）で，ケース２及びケース３の３倍の速度があった。この結果，ケース１の方式によれば，１文字当たり0.35ミリ秒であり，2800字／秒の解釈速度となる。すなわち，ケース１の解釈方式によれば，最大22.4KB/Sの解釈速度である。しかし，ここでの試験は，CPU8086(5MHzｸﾛｯｸ) + 8087数値演算プロセッサ付の金物（Univac UP10E50）に処理系Turbo-Pascal Version3.01Aにより行なった。環境が変化すれば，絶対時間は変化する。

この実験で言えることは，「8086搭載のパーソナルコンピュータ水準の資源で，9600BPSの速度で文字内容体系を解釈できる。ただし，解釈した後の機能実行は考慮していない」ということである。"Simple is best"，ありふれたプログラムが一番良効果を得た。

図５に解析見本コードを示す。

参考文献

[1] 国際標準化機構；ISO/DP8613/6, Information Processing Text and Office systems - Character Content Architeture

[2] 日本規格協会情報センター；"システムソフトウェアの標準化に関する調査研究（テキスト）報告書，昭和61年3月

[3] 日本ユニバック（株）；"パーソナルコンピュータUP10Eモデル50/60操作解説書
[4] ボーランドインターナショナル編，MSA訳；"Turbo-Pascalプログラミング・マニュアル，ＪＩＣＣ出版局

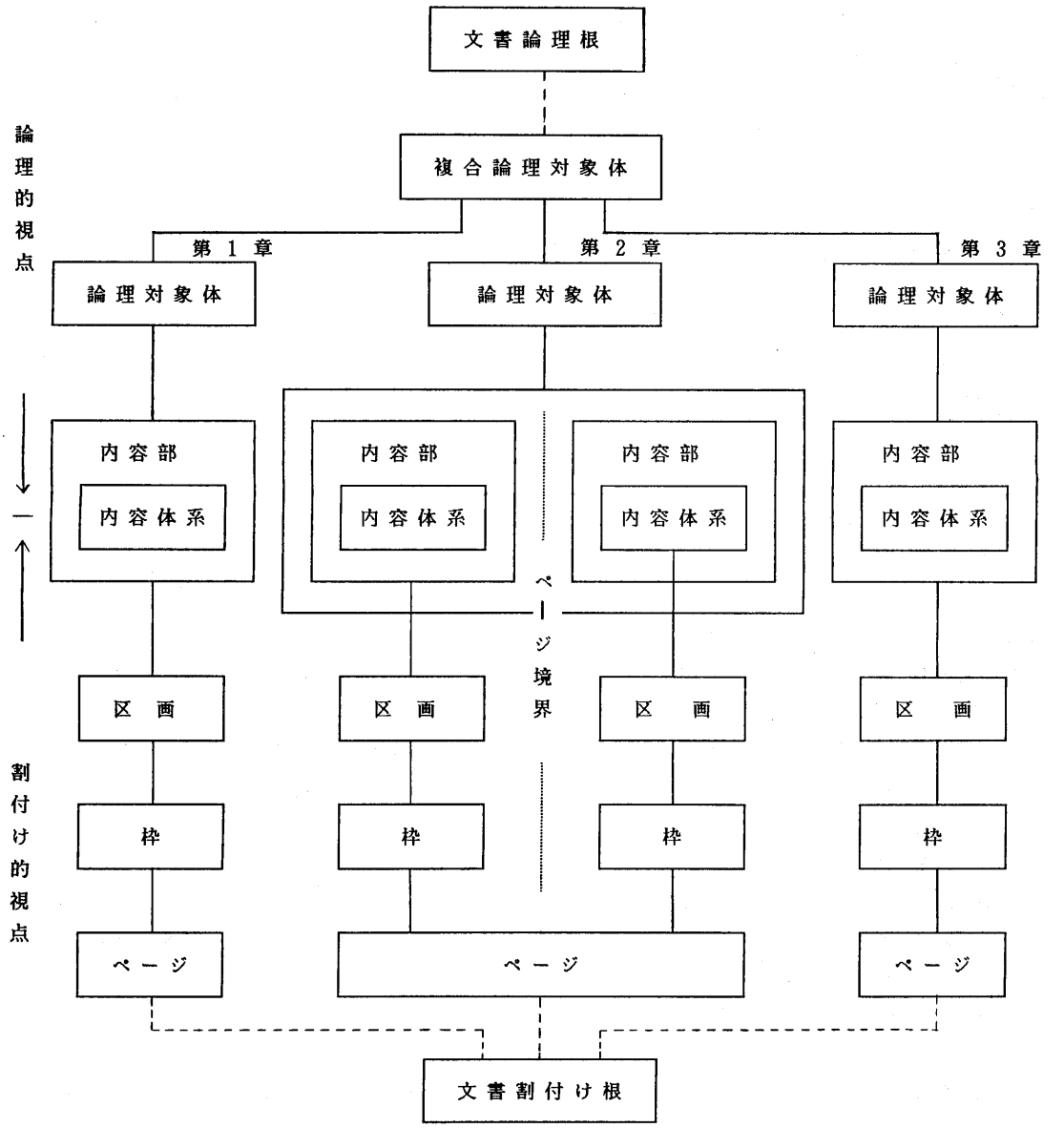
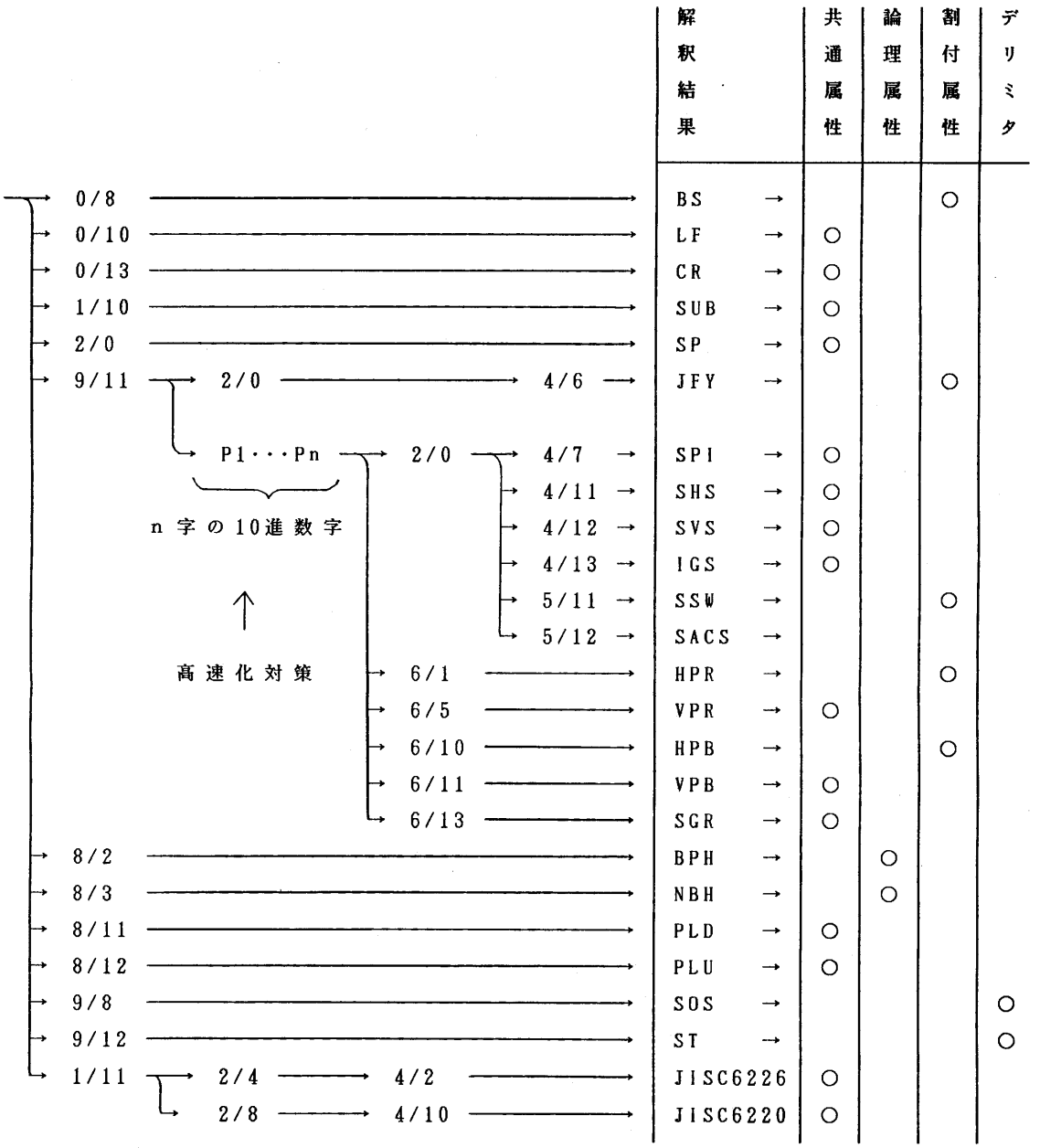


図１　事務文書体系の論理的視点と割付け的視点
A view of Office document architecture



ただし，HTR･RSPD･RVPD未指定のため，対象外とした．符号の表記法は，JISC6220情報交換用符号系の列・行による。

図2　文字内容体系の機能符号の解析木

変換方式	１００桁×１０００回の変換時間			一字単位の所要時間
	全体（秒）	共通部（秒）	正味時間	
分配計算法	35	0.1ミリ	~ 35	0.35ミリ秒
スタック法	94	0.1ミリ	~ 94	0.94ミリ秒
文字列貯蔵法	100	0.1ミリ	~ 100	1.00ミリ秒

図４　時間測定結果

```
program multitest;     {機能符号解釈速度試験，作者：若鳥陸夫  1986}
                       {言語処理系：TURBO-Pascal V.3.01A with 8087}
const lmtdgt =  100;                              {桁数}
      lmtrpt = 1000;                              {倍率}
var i, j, m, modo, err : integer;
                 p : real;                        {中間値仮置き場   }
                 c : array[1..lmtdgt] of char;    {試験データ置き場}
                 s : string[lmtdgt];              {文字列仮置き場   }
               stk : array[0..20] of real;        {スタック         }

procedure push(d:real);
  begin
    stk[m] := d; m := m + 1
  end; {push}

function pop : real;
  begin
    m := m - 1; pop := stk[m]
  end; {pop}

begin {multitest}
  for i := 1 to lmtdgt do c[i] := chr((i mod 10) + ord('0')); {試験データ生成}
  repeat
    write('変換方式選択：分配法=1, スタック法=2,');
    write('文字列法=3, 終了=4..9 -> ');
    readln(modo);
    for i := 1 to lmtrpt do                       {変換方式別実行時間計測}
    begin p := 0.0;
      case modo of
        1 : for j := 1 to lmtdgt do                          {分配法    35秒}
              p := 10.0 * p + (ord(c[j]) - ord('0'));
        2 : begin m := 0;                                    {スタック法 94秒}
              for j := 1 to lmtdgt do
              begin
                push(ord(c[j])-ord('0'));
                push(10.0 * p);
                p := pop + pop
              end; {for j}
            end;    {case 2}
        3 : begin s := '';                                   {文字列法 100秒}
              for j := 1 to lmtdgt do s := concat(s,c[j]);
              val(s,p,err)
            end;    {case 3}
      end;     {case modo of}
    end;    {for i}
    writeln('結果=',p)                                {結果の出力}
  until ((modo <= 0) or (3 < modo))
end.
```

```
program parser_test;

      {事務文書体系の文字内容体系解析パーザ試験プログラム,一字先読み法}
      {著作権者    :  若鳥睦夫（日本ユニバック）      1986              }
      {処理系      :  TURBO-Pascal V3.01A with 8087 option              }
      {参考文献    :  ISO DIS8613 part 6-1986, 'Content Architecture'   }

type  str50 = string[50];
      byte  = 0..255;
const EOT = $04; BS  = $08; LF  = $0A; CR  = $0D; SUB = $1A; ESC = $1B;
      SP  = $20; JFY = $46; SPI = $47; SHS = $4B; SVS = $4C; IGS = $4D;
      SSW = $5B; SACS= $5C; HPR = $61; VPR = $65; HPB = $6A; VPB = $6B;
      SGR = $6D; BPH = $82; NBH = $83; PLD = $8B; PLU = $8C; SOS = $98;
      SFM = $9B; ST  = $9C;
      HTR = $00; RSPD= $00; RVPD= $00; {未指定機能符号}
var   b   : byte;
      p   : real;

function read_ch:byte;                              {試験データ入力}
  var ch : byte;
  begin {read_ch}
    read(kbd,ch);
    read_ch := ch
  end;  {read_ch}

function parser(var b:byte):str50;                  {第一次パーザ頭部}

  function second_parser(var b:byte):str50;         {第二次パーザ頭部}

    function third_parser(var b:byte):str50;        {第三次パーザ}
      begin {third_parser}
        third_parser := '';
        case b of
          JFY  : third_parser := '*位置調整なし       (JFY)*';
          SPI  : third_parser := '*間隔増分           (SPI)*';
          SHS  : third_parser := '*文字間隔指定       (SHS)*';
          SVS  : third_parser := '*行間隔指定         (SVS)*';
          IGS  : third_parser := '*図形部分集合識別   (IGS)*';
          SSW  : third_parser := '*間隔幅指定         (SSW)*';
          SACS : third_parser := '*追加文字間隔設定   (SACS)*'
        end; {case b},
      end; {third_parser}

    begin {second_parser}                           {第二次パーザ本体}
      p := 0.0; second_parser :='';
      while (0 <= b) and (b <= 9) do
      begin
        p := 10.0 * p + b;
        b := read_ch
      end;  {while}
      case b of
        SP   : begin b := read_ch; second_parser := third_parser(b) end;
        HPR  : second_parser := '*文字位置相対      (HPR)*';
        VPR  : second_parser := '*行位置相対        (VPR)*';
        HPB  : second_parser := '*文字位置後退      (HPB)*';
```

```
        VPB   : second_parser := '*行位置後退          (VPB)*';
        SGR   : second_parser := '*図形表現選択        (SGR)*'
      end; {case b}
    end;   {second_parser}

  function code_designator(var b:byte) : str50;      {符号系指定}
    const DOLLAR = $24; LPAREN = $28; JISK = $42; JISA = $4A;
    begin
      case b of
        DOLLAR : begin b := read_ch; if b=JISK then code_designator:='*漢字*' end;
        LPAREN : begin b := read_ch; if b=JISA then code_designator:='*英字*' end
      end; {case b}
    end; {code_designator}

  begin {parser}                                     {第一次パーザ本体}
    parser := '';
    case b of
      BS    :  parser := '*後退          (BS)*';
      LF    :  parser := '*改行          (LF)*';
      CR    :  parser := '*復帰          (CR)*';
      SUB   :  parser := '*代替          (SUB)*';
      ESC   :  begin b := read_ch; parser := code_designator(b) end;
      SP    :  parser := '*間隔          (SP)*';
      SFM   :  begin b := read_ch; parser := second_parser(b) end;
      BPH   :  parser := '*中断許容      (BPH)*';
      NBH   :  parser := '*中断なし      (NBH)*';
      PLD   :  parser := '*一部行下げ    (PLD)*';
      PLU   :  parser := '*一部行上げ    (PLU)*';
      SOS   :  parser := '*列開始        (SOS)*';
      ST    :  parser := '*列終止符      (ST)*'
    end; {case ch}
  end;

begin {parser_test}                                  {プログラム本体}
  b := 0;
  writeln('機能符号を一符号ごとに数字[0..255]で入力し復改せよ。 終了はCTRL+C');
  while b <> EOT do
  begin
    p := 0.0;
    b := read_ch;
    write('機能符号=',parser(b),   パラメタ=',p:10:1);
    writeln
  end
end.
```

図5　解析見本コード