SE80−140

電子交換用分析テーブル記述言語

中所武司・鈴木太平
（日立）
峯尾正美
（日立通信システム）

1980年12月8日

社団法人 電子通信学会

SE80-140

電子交換用分析テーブル記述言語

Analysis Table Language for ESS

中所 武司† 鈴木 太平‡ 峯尾 正美‡
Takeshi CHUSHO Taihei SUZUKI Masami MINEO
†日立製作所システム開発研究所 ‡同左戸塚工場 ‡‡日立通信システム
Sys. Dev. Lab. Hitachi, Ltd. Totsuka Works, Hitachi, Ltd. Hitachi Comm. Sys. Inc.

あらまし 電子交換プログラムの生産性，保守性，信頼性向上を目的として，分析翻訳処理で用いる分析テーブルを問題向きに記述する言語を設計した。言語形式は，分析テーブルの記述を空欄記述形式に近いものにする一方，そのデータ構造の記述を高級言語（Pascal）風にして柔軟性をもたせた。従来のアセンブラマクロ方式に対する利点は，木構造を直接表現できるようにノードの概念を導入したり，処理指標（PI）値の自動設定などにより分析過程図に対応した記述（機能記述）ができ，記述量も減ること，保守，デバッグが容易になることなどである。

1. まえがき

近年，電子交換機の普及と共に，電子交換プログラムの生産性，保守性，信頼性の向上が重要な課題となっている。特に最近は，ハードウェアの進歩により実行時間やメモリ使用効率への要求が緩和されたことや，開発用マシンとしてセンターの大型機が用いられるなどのために，記述言語の高級化を中心に各種ツールの充実が図られている。

その際，一般に汎用計算機ソフトウェアの分野では，手続き記述を志向したプログラム作成技法の改良[1]およびその支援ツールの開発が主になっているが，電子交換ソフトウェアの分野では，その適用分野が限定できるため，問題向きを志向した，ツールの専用化の傾向が強い[2,3]。

我々もその一環として，電子交換プログラムの特徴的な機能である分析翻訳処理の方式がかなり定式化されていることに注目し，そこで用いられる分析テーブルを問題向き言語で記述できるようなテーブル生成システムの研究を行ってきた[4]。そして，既存のシステムを幾つか調査した結果，テーブルの値の設定は実際の分析過程図のイメージに近い記述が望ましいが，テーブルのデータ構造の記述は高級言語風にしてある程度の柔軟性を持たせるのが良いとわかった。特に後者については，教育用に設計され最近評価の高いPascal[5]の有するデータ型定義機能やフィールドが可変のレコード型が有用であるため，分析テーブル記述言語ATL（Analysis Table Language）はPascalをベースに設計を行った。現在は，言語設計を終え，処理系の設計開発中である。

2. 分析テーブルの特徴

2.1 分析翻訳処理

分析テーブルが用いられる分析翻訳処理とは，入力処理や出力処理が抽出した交換機内外からのサービス要求を分析し，必要な交換動作を決定するもので，その分析内容によって発信分析，数字分析，着信分析，状態分析，電けん情報分析などに分けられる。これらの各分析プログラムは，入力，出力処理プログラムにより待ち行列に登録された呼対応のトランザクションに基づいて，実行管理プログラムが起動する。

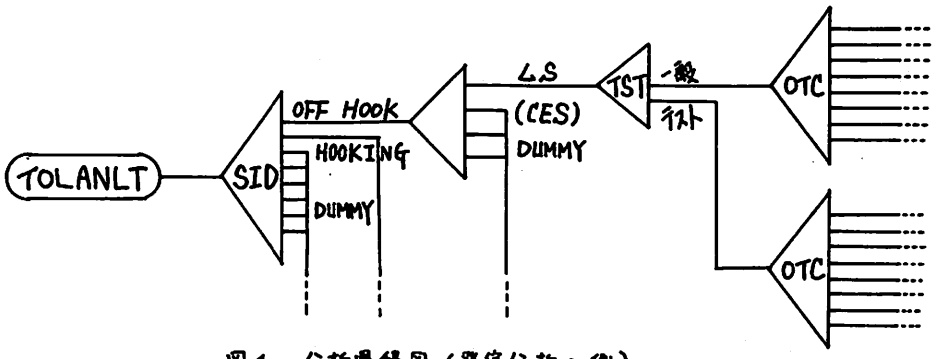

図1　分析過程図（発信分析の例）

このようにして起動された各分析プログラムは、トランザクション内の分析処理識別情報（SID：Sub Identification）およびその他の情報を用いて分析処理を行う。この時、分析テーブルが用いられるが、その構造は、分析方法が展開法か索引法かによって異なる。前者の場合は木構造で、発信、着信、着信分析に用いられている。後者は通常の表構造で、状態分析に用いられる。

2．2　分析テーブルの構造

木構造の分析テーブルは、図1に示すような分析過程図を表現したものである。これを用いた展開法の分析処理を説明する。まずトランザクション内のSID情報でルートノードの次のノードを選択、その後、適当な情報を用いて順に次のノードの選択を繰返し、分析終了条件に到着するとタスク番号、次状態番号、トランク群番号などを入手、出力して終る。

各ノードの1エントリのデータ構造は、例えば表1のようになっている。ここで、PI（Process Indication）は各エントリの種別を表し、分析プログラムの処理内容を指示するものである。この例では、出力情報テーブルを別に持ち、エントリではそのテーブルへのポインタを持つものであるが、システムによっては、分析終了時の出力は1エントリ内に含まれているものもある。一般に1語長が32ビットの大型機では後者の方法がとられる。

一方、索引法による分析で用いられる表構造テーブルは、同じ型のエントリが複数つらなったものである。

表1　PI種別とその機能（発信分析の例）

PI	データ形式	機能概要
1	PI | TBLIDX	発信エリアを抽出し、次段テーブルを索引する。
2	PI | TBLIDX	発信端子クラスを抽出し、次段テーブルを索引する。
≈	≈	≈
7	PI | TTI	トーントーキー接続タスクを決定する。
8	PI | TSKIDX	分析処理を終了し、タスク情報を得る。
9	PI | TBLIDX	試験呼表示を抽出し次段テーブルを索引する。
0 10~15	PI	イリーガル

TBLIDX：　次段テーブル相対アドレス
TSKIDX：　トランク群番号などの分析終了情報用テーブル内の相対アドレス

3．設計の基本方針

3．1　目的

本システムの狙いは、分析テーブルの保守性と生産性の向上にあるが、特に前者は、交換サービス機能の追加、変更で永続的に毎年行われることから非常に重要である。そのため、本システムでは、分析テーブルの記述をできるだけ分析過程図のイメージに近い形式で行えるようにする。

3．2　問題の定式化

本システムは基本的には汎用のテーブルジェネレータとして開発することもできる。しかし、

```
table TOLANL ; /TSWF ORIGINATING ANALYSIS TABLE

 const /TONE_TALKIE INDEX
          BTRQ=0   ;
          TRMV=8   ;
          NPAY=9   ;
          OSTP=10  ;
          TONL=12  ;
            :
            :
 type ENTRY1=record *PI : bit(4) ;
                  case *PI of 0 : ( * : bit(12)) ;
                              1 : ( TBLIDX : bit(12) ptr(TOLANLT) of LNTLR_OLC ) ;
                              2 : ( TBLIDX : bit(12) ptr(TOLANLT) of LNTLR_OTC ) ;
                              3 : ( TBLIDX : bit(12) ptr(TOLANLT) of LNTLR_OMD ) ;
                              4 : ( TBLIDX : bit(12) ptr(TOLANLT) of LNTLR_HOT ) ;
                              5 : ( TBLIDX : bit(12) ptr(TOLANLT) of LNTLR_RSG ) ;
                              6 : ( TBLIDX : bit(12) ptr(TOLANLT) of OLEN_MN   ) ;
                              9 : ( TBLIDX : bit(12) ptr(TOLANLT) of TCB_TST   ) ;
                              8 : ( TSKIDX : bit(12) ptr(TOLTSKT)) ;
                              7 : ( * : bit(6) ; TTI : bit(6))
           end ;

     SUBTBL=array(*) of ENTRY1 ;

     TOL_SID    = SUBTBL(8) label ( OFF_HOOK = 0 ;
                                    HOOKING  = 1 ;
                                    SPARE    = 2,3,4,5,6,7 ) ;
     LNTLR_OLC = SUBTBL(4) label ( LS       = 0 ;
                                    CES      = 1 ;
                                    SPARE    = 2,3 ) ;
             :
             :
 tree TOLANLT : ENTRY1 ; node TOLANLT : TOL_SID    ;
                                         OFF_HOOK    : NO2OLC0
                                         HOOKING     :
                                         SPARE       :

                         node NO2OLC0 : LNTLR_OLC ;
                                         LS          : NO2OBT
                                         CES         :
                                         SPARE       :

                         node NO2OBT  : TCB_TST    ;
                                         GENERAL     : NO2OTC0
                                         TEST        : NO2OTC1

                         node NO2OTC0 : LNTLR_OTC ;
                                         VCT_TERM    : BTRQ
                                         GEN_SUB     : NO2OMD0
                                         TERM_ONLY   : TONL
                                         OTL         : NO2OMD0
                                         TTL         : BTRQ
                                         SPARE       :
                                             :
                                             :
 list TOLTSKT (64) : ENTRY2 ;    /--TSKIDX-----TGN-----TASK_NO--
                                   NO2TSK00: DPORTM1,  TT10000
                                   NO2TSK01: DPORTM2,  TT10000
                                   NO2TSK02: DPORTM3,  TT10000
                                   NO2TSK03: DPORTM4,  TT10000
                                   NO2TSK04: DPORTM1,  TT09600
                                             :
end /END OF TABLE (TOLANL)                   :
```

定数定義
型定義
定義ユニット

木構造テーブル定義
表構造テーブル定義
データユニット

図2　本言語による分析テーブル記述例

対象範囲が広くなるほど、記述形式が一般的になり、記述量も増加する。我々は、前節でも述べたように問題向き記述を志向しているので、本システムの機能を分析テーブルの記述に必要十分なものにする。

3．3　言語形式

既存の分析テーブルを幾つか調査した結果、2章でも少し述べたが、分析テーブルのデータ構造は幾種類か存在することがわかった。したがって、各テーブル毎に異なる部分を吸収するために、本システムでは、分析過程図に対応するデータの記述の他に、データ構造の記述機能が必要である。この後者の記述機能にはある程度の柔軟性が要求されるため、言語形式は問題向きにするよりも高級言語のデータ宣言機能のようなものが適している。そこで、本システムでは、データ記述は空欄記述方式に近い問題向き形式とする一方、データ構造の記述は高級言語風にすることにした。

特に後者については、Pascalの有するデータ型定義機能やフィールドが可変のレコード型が有用であることが分析テーブルの調査から判明したので、Pascal風の構文にすることにした。

4．分析テーブル記述言語

4．1　プログラム構造

まずプログラム構造の概略を図2に示す。これは、図1の分析図を記述した例である。全体は定義ユニットとデータユニットから成り、前者は定数定義と型定義を用いて分析テーブルの構成要素であるノードや各エントリのデータ構造を定義する。データユニットは分析テーブルに対応する部分で処理系がオブジェクトを出力する。ここでは、木構造、表構造テーブルの記述が可能で、特に前者は図1で示したような分析過程図を表現し、後者がタスク決定テーブルを表現する。

4．2　定義ユニット

ここでは定数定義と型定義が行われ、後でデータユニットで記述される分析過程図のオブジェクト生成に必要な情報を提供する。定数定義の方法はPascalと同じである。型定義はほぼPascalと同じにしたが、配列型の添字の大きさについては型識別子の引用時に引数として指定できるようにした。これは、分析テーブルの各ノードのエントリ数が一定でなくとも、ノードのデータ型としては1つにまとめて定義できるようにするためのものである。図2のSUBTBLがこの例に当り、型定義の右辺の配列型の添字がnの所に、その型を引用した時の実引数が対応する。

4．3　データユニット

ここで定義される木構造テーブルと表構造テーブルの違いは、ノードの概念の有無である。即ち、図1からもわかるように、最開法で用いる分析過程図は木構造で、その各ノード毎に情報がまとめられているため、ノードの概念を明示的に導入した。個々のノードの記述はエントリを単純に並べた表構造とほぼ同じである。

木構造テーブルのもう1つの特徴は、各ノードのエントリのデータ型はテーブル全体で同一であることである。勿論、表1の例のように各エントリのデータ構造は幾種類かあるが、これらは特定のノードに依存したものではなく、任意のノードの任意のエントリのデータ構造でありえる。これは最開法による分析処理方式の前提となる条件であり、本システムでもこれに呼応してエントリのデータ型はテーブル全体で1つに限るようにした。そして、データ構造の詳細な部分の差異はレコード型の記述の中に可変部分を許すことで表現できるようにした。例えば図2の例では、TOLANLTという名前の木構造テーブル定義ではそのエントリのデータ型がENTRY1であることを明記している。

各エントリの記述は、基本的には定数の並びであるが、先頭にラベルを付けても良い。ラベルとしては識別子と整数が可能で、識別子を用いた場合は通常のラベル機能であるが整数を用いた場合はそのエントリが先頭から何番目に位置するかを示す。この整数型ラベルは、線数指定や領域指定も可能にしているので同一データのエントリの一括記述に便利である。特に分析テーブルは、将来の拡張に備えて各ノードに未使用エントリを確保しており、これらのエントリの記述にも一括記述機能が有用である。

4．4　データ型

本システムでサポートすべきデータ型は，分析テーブルの記述に必要かつ十分なものにするという基本方針に沿って決定した。まずPascalを基本とし，不必要なものを除き，必要なものを加えると共に，Pascalと同じデータ型に対する拡張も行った。その結果，基本データ型としては，整数型，ポインタ型，スカラ型，セレクタ型を備え，その他に，配列型，レコード型，番号型，およびユーザ定義データ型を許した。これらのデータ型の性質はそのまま本システムの分析テーブルジェネレータとしての特徴を表すと思われるので，以下個々に説明する。

4．4．1　整数型

分析テーブルを構成するデータ要素は，基本的には整数型の値とポインターである。このうち前者に当るものは，

(1)　タスク番号（ＴＳＫＮ）
(2)　遷移先状態番号（ＮＳＴＮ）
(3)　トランク群番号（ＴＧＮ）
(4)　ジョブ番号（ＪＯＢ）
(5)　処理指標（ＰＩ）

などがある。(1)～(3)は通常の分析終了時情報、(4)と(5)は分析処理方法に依存して必要となるものである。

4．4．2　ポインタ型

ポインタ型は先の整数型と共に分析テーブルの基本データ要素であり，

pointer　（ID1）　of　ID2

の構文で表す。ID1はポインタ型の値がID1からの相対アドレスであることを示し，出来れば絶対アドレスとする。ID2の指定は，木構造テーブルの場合のみ有効で，ポインタが指す先がノードでかつそのノードのデータ型はID2（型識別子）であることを示す。本機能の有用性はレコード型のＰＩの自動設定の所で述べる。

4．4．3　スカラ型

この型は見かけ上はPascalと全く同じであるが，値に用いる各識別子は定数定義をする必要がある。その理由は，本システムが生成するテーブルを参照する分析プログラムは本システムとは無関係に作成されるため，システム側で適当な値を割当てるのは実用的でないからである。スカラ型の導入理由についてはレコード型のＰＩ自動設定の所で述べる。

4．4．4　セレクタ型

これはレコード型のフィールドのデータ型としてのみ使用可能で，その値としては自分のフィールド識別子だけである。この導入理由もPI自動設定の所で述べる。

4．4．5　配列型

これは通常のものとほぼ同じであるが，宣言文の右辺に用いられた場合は添字の大きさを明記しなくても良い機能を付加している。即ち，添字が＊の場合は左辺の型識別子が実際に引用された時の実引数を対応させる。この用途はすでに４．２で述べた。

4．4．6　レコード型

これはＰＬ/Ｉの構造体と同じようなものであるが，可変部分を有するPascalを基本にした。まず，Pascalのレコード型について次の例で説明する。

```
record
  ＮＡＭＥ : array [1..10] of char;
  ＡＧＥ   : integer;
  case   S : SEX of
         ＭＡＬＥ : (θ : OCODE ;SS: integer);
         ＦＥＭＡＬＥ : (B, W, H : real)
end
```

このデータ型では，まず名前と年齢のフィールドがＮＡＭＥ，ＡＧＥというフィールド名で固定的に確保される。そして，それに続くフィールドは男性と女性で異なるため，可変部分が記述されている。選択されるフィールドリストにはＭＡＬＥおよびＦＥＭＡＬＥというケースラベルが付いており，実際の選択はＳと名付けられた，ＳＥＸ型のタグフィールドの値によって行われる。例えば，タグフィールドＳがＭＡＬＥという値を持つ時は，このレコードは名前，年齢，職業，収入の各フィールドから構成される男性用のデータになる。

本言語では，このPascalのレコード型を次のように拡張した。

(1)　フィールド名の代りに＊を書くと，その

フィールドは未使用エリアを表す。

(2) 可変部分のタグフィールドには固定部分に現れたフィールド名を指定する。このデータ型は整数型に限るため、データ型を指定する型識別子は不要である。

(3) ケースラベルとして範囲指定ができる。

(4) 頭に＊を付けたフィールド名をタグフィールドに書いた場合は、そのフィールドの値はシステム側で自動設定する。

これらのうち、(1)はデータのストレージ割当を正確に記述するためのものであるが、その他はいずれも可変部分に関するもので、現在の分析処理方法との関連が深い。即ち、2.2節で少し述べたように、分析プログラムは各ノードのエントリの先頭フィールドのPIの値によって次の処理を選択しており、それに対応して表1の例のようにエントリのデータ構造が変わっている。そこで本システムでは、これらの変化部分をレコード型の可変部分で表現することにより、テーブル全体でエントリのデータ型を1つと見るようにしている。従って、タグフィールドにはPIのフィールド名が指定されると考えて良いので上記の(2)の拡張を行った。(3)は、PIの値が異ってもデータ構造が同じ場合が多いので一括記述できるようにしたものである。

ところで、このPIはあくまでも分析処理方法に依存して導入されたものであり、分析テーブルの論理的な意味を表現している分析過程図上には出ていない。従って、この分析過程図を記述するデータユニットではPIの値を記述しなくて良いような方法が、問題向き記述形式として望ましい。そこでこれを可能にしたのが上記(4)の拡張である。システム側では、可変部分のタグフィールドの値と対応するフィールドリストとが1対1の対応関係にあることを前提にして、データユニットで指定された、各エントリに対応する定数並びからそれに合ったデータ構造のフィールドリストを選び、同時にPIの値も設定しようとするものである。

しかしながら、実際にはこの1対1対応の前提条件は保証されない。例えば図3(a)の例では以下の場合にPIの値を一意に決定できない。

(1) データとしてラベル名がくると、PIが1～5のいずれかはデータ構造からは決定できない。但し、PI＝5の場合はラベルのあるテーブルが他と異なるので識別可能。

(2) データとして整数値が来ると、PIが6か7か決定できない。

そこで本システムではこれらの問題を解決し、PIの自動設定機能を現実的なものにするため、以下の機能を備えた。

(1) ポインタ型がノード選択に用いられる場合、相手ノードのデータ型を指定できる。

(2) セレクタ型の導入。

(3) スカラ型の導入。

上記のPIの値を決定できないケースの最初のものは、次の処理すべきノードを示すエントリのPIの値がそのノードの種類に依存していることから、概ね(1)によって解決する。しかし、全く同じ型のノードを指しながらPIの異なるような実例もあり、その時は(2)のセレクタ型の使用によって解決する。後のケースについては、値が限定されている場合はスカラ型の導入によって解決し、それ以外はセレクタ型を用いる。

先の図3(a)の例をこれらの機能を用いて書き直すと同図(b)のようになる。ここで、同図(c)に示したデータユニットのノード記述例のPIの値の決定方法について述べる。

(1) 第0番目のエントリはOLC型のノード名なのでPI＝1とする。

(2) 第1番目のエントリはLSC型のノード名なのでPIは3か4であるが、フィールド名TSTSETが指定されているのでPI＝4とする。

(3) 第2番目のエントリはタスク決定テーブルTOLTSKT内のラベルなのでPI＝5とする。

(4) 第3番目のエントリはスカラ型TTIDXの値なのでPI＝7とする。

(5) 第4番目のエントリは整数値なのでPI＝6とする。

(6) 第5～7番目のエントリはデータ無しなのでPI＝0とする。

4.4.7 番号型

今回調査した既存プログラムでは、分析終了情報の1つであるタスク番号（TSKN）が実際には遷移先状態番号（NSTN）とサブ番号（SN）から構成されているものがあった。そこでは、このようなデータの記述は、アセンブ

```
type ENTRY=record
            *PI:bit(4);
            case *PI of
                  0:(*:bit(12));
                  1:(TBLIDX:bit(12) ptr(TOLANLT));
                  2:(TBLIDX:bit(12) ptr(TOLANLT));
                  3:(TBLIDX:bit(12) ptr(TOLANLT));
                  4:(TBLIDX:bit(12) ptr(TOLANLT));
                  5:(TSKIDX:bit(12) ptr(TOLTSKT));
                  6:(SPI:bit(8),*:bit(4));
                  7:(*:bit(6),TTI:bit(6))
            end;
```

(a) PIの自動設定が不可の例

```
type ENTRY=record
            *PI:bit(4);
            case *PI of
                  0:(*:bit(12));
                  1:(TBLIDX:bit(12) ptr(TOLANLT) of OLC);
                  2:(TBLIDX:bit(12) ptr(TOLANLT) of OTC);
                  3:(TBLIDX:bit(12) ptr(TOLANLT) of LSC);
                  4:(TSTSET:select,
                     TBLIDX:bit(12) ptr(TOLANLT) of LSC);
                  5:(TSKIDX:bit(12) ptr(TOLTSKT));
                  6:(SPI:bit(8),*:bit(4));
                  7:(*:bit(6),TTI:bit(6):TTIDX)
            end;

type TTIDX=(BTRQ,TRMV,NPAY);

const BTRQ=0;
      TRMV=8;
      NPAY=9;
```

(b) PIの自動設定が可能な例

```
node N02OMD1:OMD;                    PI値
              0:N02OLC3        ---> 1
              1:TSTSET,N02LSC2 ---> 4
              2:N02TSK13       ---> 5
              3:TRMV           ---> 7
              4:8              ---> 6
            5-7:               ---> 0
```

(c) データユニットのノード記述例

図3 PI（処理指標）の自動設定機能

うのマクロ機能を用い、ユーザが例えばタスク番号としてTT12345を指定すると、123を遷移先状態番号、45をサブ番号として設定するという方法が採られている。こうした方法が本質的か否かは別にして、ユーザに現在の方式に対する変更要求は避けるという本システム設計の基本方針に沿って、このようなデータ設定法を可能にするために番号型を導入した。

この例の場合、データ型は、

TSKN: number (AA11122) of
(NSTN: bit(10), SN: bit(6))

と表現される。ここでAA11122はこの番号型の定数のパターンを表し、3文字目から5文字目までを整数値とみて第1フィールドNSTNに対応させ、6文字目と7文字目を同じく整数値とみて第2フィールドSNに対応づけることを意味する。

4.4.8 ユーザ定義データ型

本システムの型定義機能の1つの特徴は配列の添字の大きさをそのデータ型の引用時に指定できることである。その主な用途については4.2節で既に述べた。

もう1つの特徴としては、ノードのデータ型を定義する時に、そのデータ型のノード内のエントリの整数型のラベルとして用いる識別子を定義する機能がある。既ち、4.4節で述べたように、データユニットの各エントリの先頭には整数型ラベルを付けて良いが、ノードは、そのデータ型が決まれば各エントリの意味づけもその位置によって決まる。そこで、この意味づけを表現する識別子を整数型ラベルの代りに用いることにより、ドキュメント性を向上させるために本機能を導入した。図2のデータ型TOL_SIDではこの機能が用いられている。

4.4.9 その他の機能

本節はこれまで各データ型について述べてきたが、全体的機能として、ビット長指定と初期値指定がある。ビット長指定については既に図2，図3の例で用いているもので、基本型データのうちの整数型、ポインタ型、スカラー型についてはそのストレージのビット長を指定できる。本システムの生成するテーブルを参照するプログラムは本システムとは独立に作成されるため、テーブルのデータ構造はユーザが詳

細に定義する必要があり，本機能は必須である．

5．本システムの特徴

本システムを用いて分析テーブルを記述した場合，従来のアセンブリ言語のマクロ機能を用いた場合に比して，次のような利点がある．

(1) 分析過程図に対応した記述形式なので，分析過程図から直接記述できる．
(2) 宣言的記述形式，各種省略機能などにより記述量が大巾（半分以下）に削減される．
(3) 分析処理方式に依存して生じている処理指標 PI の値が自動設定されるので，プログラム機能記述に近い形式になる．
(4) デバッグ，保守に最適な，分析過程図に機械語を付加したドキュメントリストが出力される．
(5) 処理系で，データ型チェック，木構造チェックなどを行うので，エラーが早期に発見される．

これらの利点のうち，特に(1)，(2)は生産性向上，(3)，(4)は保守性向上，(5)は信頼性向上に寄与する．

6 むすび

電子交換プログラムの生産性，信頼性向上の一環として，分析翻訳処理で用いる分析テーブルを問題向きに記述できるような言語を設計した．

言語設計にあたっては，

(1) 機能は，現在用いられている分析テーブルの記述に必要かつ十分なものとする，
(2) 記述形式は問題向きを志向し，分析過程図のイメージに合わせる．

などを基本方針とした．そして，既存の分析テーブルのデータ構造は幾種類が存在するという調査結果から，分析過程図の記述の他に，そのデータ構造の記述機能も備えることにし，前者は宣言的記述形式に近いものにする一方，後者はある程度の柔軟性が必要なので高級言語（Pascal）風にした．特にベースにする高級言語として Pascal を選んだ理由は，それが有する型定義機能やフィールドが可変のレコード型が必須であるためである．

言語仕様での Pascal に比しての特徴としては，

(1) 型定義による型識別子の引用時に配列の添字の大きさを引数で指定できる．
(2) 木構造テーブルと表構造テーブルの識別および前者についてはノード，エントリとそれらのデータ型を明記させる，
(3) ポインター型の値として絶対アドレスと相対アドレスの選択が可能である，
(4) レコード型の可変部分のタグフィールドの値をシステム側で自動設定する，
(5) 新たにセレクタ型，番号型のデータ型の導入，
(6) ビット長の指定，初期値指定の導入，

などがある．

今後に残された問題として，現在，分析処理プログラムのほとんどはアセンブリ言語で記述されているが，近い将来に高級言語に変更になる可能性がある．その場合の本システムの移行性の問題を検討していく必要がある．

謝辞　終りに，有益な御討論，御意見を頂いた日立製作所システム開発研究所の渡辺担当主任研究員ならびに同社戸塚工場の関係各位に深謝します．

文献

(1) 中所：“プログラムのモジュール化技法”，信学誌，62，1，P.91（昭54－1）．
(2) 加久間，佐藤：“電子交換用支援プログラムシステム”，信学誌，61，6，P.614（昭53－6）．
(3) 木谷他3名：“呼処理遷移向き言語”，信学技報，SE 79－103（1980－1）．
(4) 黒岩他4名：“電子交換機用仕様記述言語－SDL－”，日立評論（昭55－に発行予定）
(5) Jensen, K. and Wirth, N.：“PASCAL－User Manual and Report”，Springer－Verlag，New York (1975)．