ソフトウェア工学 46－4
（１９８６．２．６）

抽象的順序機械からＡＳＬ／Ｆプログラムへの変換システムについて

安松　一樹　　杉山　裕二　　鳥居　宏次
（大阪大学　基礎工学部　情報工学科）

１．　まえがき

代数的記述言語は意味の定義が簡潔であり、問題の仕様の記述など抽象度の高いものから、抽象度の低い関数型のプログラムまで同じ言語で書くことができ、種々の検証が比較的容易に行なえる等の利点をもつ。

一般の代数的記述を関数型プログラムあるいは手続き型プログラムに機械的に変換する手法はないが、代数的記述のうち抽象的順序機械と呼ばれるものについては系統的な変換手法が知られている[1]。

抽象的順序機械とは、データ型として抽象的状態、関数として状態遷移関数（Ｓ－関数と呼ぶ）、状態から具体的な値を取りだす関数（Ｖ－関数と呼ぶ）を導入し、公理で各Ｓ－関数について遷移後のＶ－関数の値がどう変化するものかを記述するものである。

前述の手法では、まず、抽象的順序機械により、どのような条件が成立すれば、Ｖ－関数がどのように更新されるかを示す決定表を抽出し、その決定表により効率の良い流れ図を導出し、その流れ図に従ってプログラムを生成する。

本システムは、その手法を用いて、抽象的順序機械として代数的に記述されたプログラムを、関数型言語ＡＳＬ／Ｆに変換するものである。本システムの特徴は、利用者との対話形式により、決定表から流れ図の導出を行なうことである。これにより、頻繁に行なわれる条件判定を先にする等、効率の良い流れ図が導出でき、プログラムのこの種の最適化を行なうことができる。

本報告では、システムの、入力として許される抽象的順序機械、変換方法、ならびに特徴と機能について述べる。

２．　抽象的順序機械

ここでは、まず本システムの入力として許される抽象的順序機械の概念とその関数を説明した後、抽象的順序機械による仕様の記述法について述べる。

２．１　抽象的順序機械の概念と関数

抽象的順序機械は、ブール値、整数などの基本データタイプや、それらの上の基本演算が定義されている基底代数を前提とする代数的仕様[2]の一つである。代数的仕様[2,3,4,5]は、仕様を（抽象）代数系として、（その代数系を構成する）いく種類かの集合（ソートと呼ばれ、データタイプに相当する）と、それらソート上に定義された演算（関数）、及びそれら演算の意味を定義する公理集合で表わすものである。基底代数で定義されている基本ソート（データタイプ）や基本関数は、本システムなど各種処理系の組み込みデータタイプや演算に相当し、それらの宣言や公理は、基底代数を前提とする代数的仕様では、陽に書かない。

抽象的順序機械は代数的言語の制限された形である。基本ソート以外のソートがただ一つであるような仕様において、新たに定義する関数及びその公理が次の条件を満たすとき、それを抽象的順序機械と呼ぶことにする。その新しいソートは状態ソートと呼ばれ、抽象的順序機械の状態集合に対応する。

１）引数に状態ソートを二つ以上持つ関数は、if-関数を除いて、存在しない。

２）公理の左辺の形は f(X1,X2, …) 又は f'(g(X1,X2, …), …) である。ただしｇ（Ｓ－関数と呼ぶ）の値域は状態ソートである。ｆの引数の一つ、ｆの値域、ｆ'（Ｖ－関数と呼ぶ）の値域はそれぞれ状態ソートであってもよい。また、変数 X1,X2, … は全て異なる。

３）全てのＳ－関数 ｇ、Ｖ－関数 ｆ' の組について、 f'(g(X1,X2, …), …,Xn) と公理で等価となるような項 t(X1,X2, …,Xn) が存在する。ここで、 t(X1,X2, …,Xn) は、Ｖ－関数、基本演算、及び変数X1,X2, …,Xn のみからなる項である。

4）同じ左辺を持つ公理は複数個存在しない。また左辺が f'(g(X1,X2, …),…) の形の公理があるとき、左辺が f'(X1,X2, …) や g(X1,X2, ...) の形の公理はない。

5）公理の右辺には、その左辺に現われない変数は存在してはならない。

抽象的順序機械では、記述すべき対象を、図1のように、入力系列により、抽象的状態が初期状態から最終状態まで状態遷移していくものとしてとらえる。上記条件1）～3）は、状態が常に一つであり、遷移後の状態が遷移前の状態で完全に現わされるための十分条件である。S－関数は状態遷移関数に相当し、V－関数は状態から値を取りだす関数である。図1の状態遷移を流れ図で表わしたものを図2に示す。条件4）及び5）は抽象的順序機械が（項書き換え系として）合流性を満たすための十分条件である[2]。このため抽象的順序機械の無矛盾性は保障されている[2]。

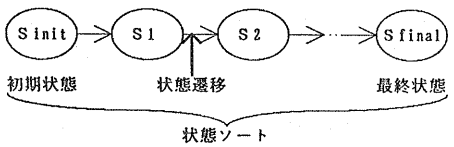


図1　抽象的順序機械の状態遷移

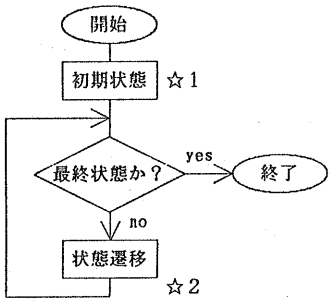


図2　状態遷移の流れ図

```
for j:=n downto 1 do
  for i:=0 to j-1 do
    if a[i] > a[i+1] then
      swap(a, i, i+1);
```

図3　ﾊﾞﾌﾞﾙｿｰﾄのｱﾙｺﾞﾘｽﾞﾑ

```
SPEC      Bubble_sort;

INCLUDE    ARRAY(ary, int, 50, ALOAD, ASAVE);

SORT       state;

OP                        /* target function */
    bubble_sort :  ary, int ->  ary  ;
                          /* loop function */
    bubble      :  state    ->  state;
                          /* initial function */
    Initial     :  ary, int ->  state;
                          /* s_function */
    comp_exch   :  state    ->  state;
                          /* v_function */
    a           :  state    ->  ary  ;
    i, j        :  state    ->  int  ;
                          /* aux function */
    swap        :  state    ->  ary  ;

AXIOM

    bubble_sort(A, n) == a(bubble(Initial(A, n))); {☆1}

    bubble(S) ==                                   {★1, ☆2}
        if   j(S) = 0
        then S
        else bubble(comp_exch(S));

    a(Initial(A, n)) == A;                         {☆3}

    i(Initial(A, n)) == 0;                         {☆3}

    j(Initial(A, n)) == n;                         {☆3}

    a(comp_exch(S)) ==                             {☆4}
        if ALOAD(a(S), i(S)) > ALOAD(a(S), i(S)+1)
        then swap(S)
        else a(S);

    i(comp_exch(S)) ==                             {☆4}
        if i(S) < j(S)-1
        then i(S)+1
        else 0;

    j(comp_exch(S)) ==                             {★2, ☆4}
        if i(S) < j(S)-1
        then j(S)
        else j(S)-1;

    swap(S) ==                                     {☆5}
        ASAVE(
            ASAVE(a(S), i(S), ALOAD(a(S), i(S)+1)),
            i(S)+1,
            ALOAD(a(S), i(S))
        );
END;
```

図4　バブルソートの抽象的順序機械プログラム

2．2　バブルソートの記述例

例として、バブルソート（図3）を記述する。抽象的順序機械として記述したバブルソートプログラムを図4に示す。

S－関数ｃｏｍｐ＿ｅｘｃｈとV－関数ａ，ｉ，ｊを導入する。状態ソートはSで表わす。ａは配列、ｉ，ｊは整数とすると、状態遷移後（図2，☆2）の各V－関数の値は、図4，☆4のように記述できる。（ALOAD は配列の要素を参照する関数。 swap は配列の要素の入れ換えをする関数）

状態遷移の流れを制御するものとして、"ループ関数"を導入し、全体の制御の流れを表わす。bubble をバブルソートのループ関数とすると、図2の制御の流れは、図4，☆2のように記述できる。

状態を初期化するものとして、"状態初期化関数"を導入し、Ｖ－関数の初期値を設定する。initial をバブルソートの状態初期化関数とし、入力系列を配列Ａ，整数ｎとすると、初期状態（図２，☆１）の各Ｖ－関数の値は、図４，☆３のように記述できる。

抽象的順序機械の入出力を表わすものとして、"目的関数"を導入する。bubble_sort をバブルソートの目的関数とし、出力をＶ－関数ａ（配列）とすると、図４，☆１のように記述できる。

ｓｗａｐは"補助関数"と呼ばれ、抽象的順序機械の補助的な記述をする。バブルソートにおけるｓｗａｐは、図４，☆５のように記述できる。（ASAVE は配列に要素を割り当てる関数）

以上六つの関数により、抽象的順序機械を記述する。

２．３　抽象的順序機械の構文

本システムでは前提とするソートとして、整数、実数、ブール、文字列、ヒープ、ポインタ、及び配列、ファイル、タプル（並び）がある。また、その上の演算として約８０の関数がある。

抽象的順序機械のソースプログラムは次の六個の部分からなる。

（１）ＳＰＥＣ部
　プログラム名の記述。
（２）ＩＮＣＬＵＤＥ部
　配列の記述。
（３）ＳＯＲＴ部
　抽象的状態のソート名、及びタプルのソート名の記述。
（４）ＯＰ部
　定義関数の種類と引数のソート、結果のソートの記述。
（５）ＡＸＩＯＭ部
　定義関数の定義本体の記述。
（６）ＥＮＤ部
　プログラムの終了。

ＡＸＩＯＭ部では、各関数を公理の形で定義する。

公理定義部は、

　目的関数名(引数) == 項；
　ループ関数名(引数) == 項；
　Ｖ－関数名(状態初期化関数名(引数))
　　== 項；
　Ｖ－関数名(Ｓ－関数名(引数)) == 項；
　補助関数名(引数) == 項；

のいずれかの形をしている。

また、｛　と　｝で囲まれた部分は注釈である。

３．　抽象的順序機械からＡＳＬ／Ｆプログラムへの変換方法

抽象的順序機械には、関数型プログラムあるいは手続き方プログラムに変換する方法が知られている。

その変換方法とは、まず、図２，☆２の抽象的順序機械の状態遷移の部分を決定表で表わす。そして、その決定表から効率のよい流れ図を導出し、導出された流れ図に従ってプログラムを生成するものである。

本システムでは、この変換方法に基づいて、抽象的順序機械からＡＳＬ／Ｆを対話的に導出するものである。まず、変換の方法を決定表の抽出、決定表から流れ図への変換、流れ図からＡＳＬ／Ｆプログラムの生成、の三つの部分に分け、図４のバブルソートプログラムを例に取り、本システムの変換の様子を示していく。

３．１　決定表の抽出

抽象的順序機械では、各Ｓ－関数がどのような条件のもとで適用されるかがループ関数の定義（２．２）で表わされており、Ｓ－関数による遷移後のＶ－関数の値が

　Ｖ－関数名(Ｓ－関数名(引数)) == 項；

の形の公理により定義されている。

よってＶ－関数の値の変化を知るためには、二つの条件、Ｓ－関数の適用条件とそのＳ－関数による状態遷移後のＶ－関数の更新条件との論理積が決定表におけるＶ－関数の更新条件となる。

図４のプログラムのＶ－関数ｊの更新を例として考える。図４，★２によりｊはＳ－関数 comp_exch による状態遷移後、条件

<cond1> := ! (i(S) < j(S) - 1)

("!"は、論理否定を表わす。)

のとき、j(S) − 1 となることがわかる。また、図４，★１により comp_exch は

<cond2> := ! (j(S) = 0)

のとき、状態遷移を行うことがわかる。（ここでは j(S) = 0 のときプログラムが終了することもわかる。）

従って条件 <cond1> & <cond2> のとき関数 j は　j → j − 1　と値が更新される。

以上のようにして、図４のプログラムより抽出された決定表を図５に示す。（図５の動作部の <halt> はプログラムの終了を示す。）

３．２　決定表から流れ図への変換

決定表の条件部分は述語（例えば j ＝０など）の論理式として考えられる。そこで任意の述語 p に注目すると、各条件についてｐが真の場合に依存しているか、偽の場合に依存しているか、それ

Conditions	Actions
j=0	<halt>
ALOAD(a,i)>ALOAD(a,i + 1) & ! (j=0)	a -> swap(a,i)
i<j - 1 & ! (j=0)	i -> i + 1
! (i<j - 1) & ! (j=0)	i -> 0
! (i<j - 1) & ! (j=0)	j -> j - 1

図５　バブルソートの決定表

	Conditions	Actions
t	true	<halt>
f	ALOAD(a,i)>ALOAD(a,i + 1)	a -> swap(a,i)
	i<j - 1	i -> i + 1
	! (i<j - 1)	i -> 0
	! (i<j - 1)	j -> j - 1

図５の決定表を　j=0 で分割

	Conditions	Actions
t	true	i -> i + 1
f	true	i -> 0
	true	j -> j - 1
	ALOAD(a,i)>ALOAD(a,i + 1)	a -> swap(a,i)

上の決定表の　f の部分を　i<j - 1 で分割

	Conditions	Actions
t	true	a -> swap(a,i)
f		

上の決定表の　i の部分を　ALOAD(a,i)>ALOAD(a,i + 1) で分割

図６　決定表の分割

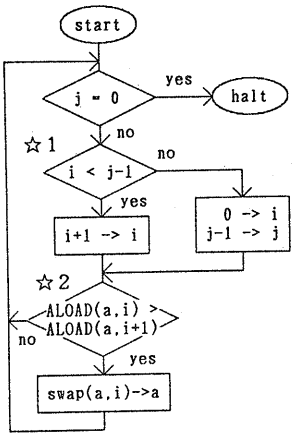


図７　バブルソートの流れ図

ともｐには依存していないかの三つの場合に決定表を分割できる。以下、三つの場合をそれぞれＴ，Ｆ，Ｉで表わす。

図５の決定表を述語ｊ＝０に注目して分割したとする。条件 <ｊ＝０> の部分はＴに分割され、そのとき無条件で動作の <ｈａｌｔ> は実行されるので新たに条件は <ｔｒｕｅ> となる。また条件 <ｉ＜ｊ－１ & !(ｊ＝０)> の部分はＦに分割され、新たに条件は<ｉ＜ｊ－１> となる。

分割された三つの部分決定表もそれぞれ異なる述語に注目して分割することができ、全ての条件が <ｔｒｕｅ> になるまで繰り返し分割する。決定表の分割の様子を図６に示す。

各述語での分割を流れ図での条件の分岐としてとらえることにより、決定表から流れ図を導出することができる。ここで効率のよい流れ図を導出するためには、出現頻度の高い述語から順に注目して分割する、などの手法をとればよいことがわかる。決定表の分割から導出した流れ図を図７に示す。

３．３　流れ図からＡＳＬ／Ｆプログラムの生成

関数型言語ＡＳＬ／Ｆは代数的言語のサブクラスであり、既にその最適化コンパイラがＵＮＩＸ上に作成されている[6]。

ＡＳＬ／Ｆはソートとして状態ソートを持たないが、組み込みソート及びその上の関数を持つ。抽象的順序機械の組み込みソート及びその上の関

数は、ＡＳＬ／Ｆを参考にしている。

ＡＳＬ／Ｆの公理は、f(X1,X2, …) == 項；の形をしており、引数の個数に制限はなく、また定義文左辺の中には互いに異なる変数しか許されない。さらに定義文右辺には、その左辺に現われない変数は存在してはならない。

流れ図からＡＳＬ／Ｆプログラムを生成する手法としては、各Ｖ－関数をＡＳＬ／Ｆの定義関数の変数とし、ループ部分を関数の再帰呼び出しにより実現する。プログラム全体の入出力は抽象的順序機械の目的関数を用いて記述する。流れ図の分岐の部分はＡＳＬ／Ｆの if_then_else 表現（if 関数）を用いて実現できるが、流れ図の合流後（図７，☆２）の記述が問題となる。合流後の部分を if_then_else 表現の then 部と else 部にそれぞれ含めて記述することもできるが（図８）、これではＡＳＬ／Ｆの関数定義が冗長なものとなる。そこで次のような手法を取る。

１）流れ図の合流後の部分にＶ－関数の更新が来る場合（図９（ａ））、Ｖ－関数の更新部分を then 部と else 部に含める。

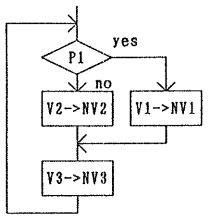
２）流れ図の合流後の部分に判定（述語の評価）が来る場合（図９（ｂ））、新たな関数を定義する。

以上　１），２）の手法を再帰的に適用することにより冗長性を排除する。（実際、冗長性を排除しなければ関数定義が巨大になり、コンパイラの制限によりコンパイルできない。）

この手法を取ると図１０のような場合、☆１と☆２の部分で同じ関数を２度定義することになるが、システムではそれを判断し、一つの関数として定義する。

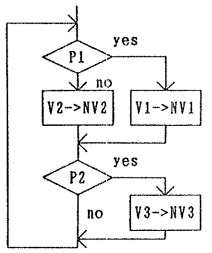
```
.....
if i+1 < j
then
  if ALOAD(a, i) > ALOAD(a, i+1)
  then .....
  else .....
else
  if ALOAD(a, i) > ALOAD(a, i+1)
  then .....
  else .....
```

図８　冗長な関数定義




```
AX1(V1, V2, V3) --
if P1
then AX1(NV1, V2, NV3)
else AX1(V1, NV2, NV3);
```

(a)




```
AX1(V1, V2, V3) --
if P1
then AX2(NV1, V2, V3)
else AX2(V1, NV2, V3);

AX2(V1, V2, NV3) --
if P2
then AX1(V1, V2, NV3)
else AX1(V1, V2, V3);
```

(b)

図９　冗長さを排除する関数定義

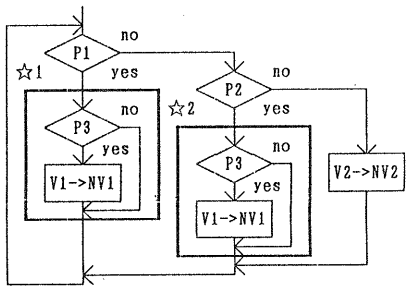


図１０　同じ部分を持つ流れ図

冗長性を排除するためＡＳＬ／Ｆで複数の関数を定義するが、それにより次のような問題が生じる。抽象的順序機械ではＳ－関数により状態遷移が行なわれる場合、そのＳ－関数により値が更新されるＶ－関数は、全て同時に評価されなければならない。しかし図７の流れ図では☆１の部分と☆２の部分では評価されるｉの値が異なる。生成するＡＳＬ／ＦプログラムではＶ－関数を変数とし、更新後の値を他の定義関数に変数として受け渡すので、（関数型言語の並列リダクションが使えず、）評価される値が異なる。従って同時性を実現するため、状態遷移前のＶ－関数の値と遷移後のＶ－関数の値を別々の変数とし、述語の評価と値の更新を遷移前の変数の値で行ない、更新された値を遷移後の変数に代入することにする。これは他の手続き型言語を生成する場合にも生じる問題であり、同じ解決方法を取ることができる。

同時性を考慮し、抽象的順序機械の入出力を加えた流れ図を図１１に示す。

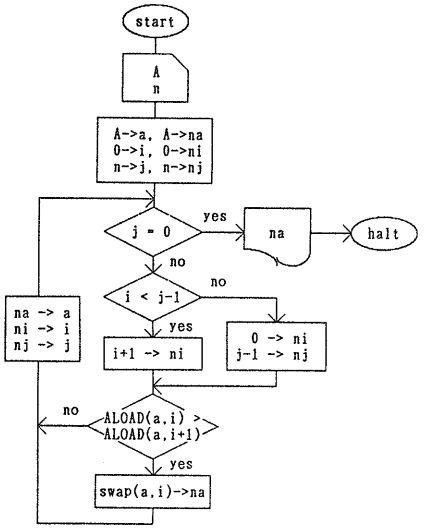


図１１　同時性を考慮したバブルソートの流れ図

４．　変換システム

ここでは、システムの概要と、システムに用意されているコマンドとその機能について述べる。

４．１　システムの概要

本システムは抽象的順序機械として記述されたプログラムから、決定表を抽出し、利用者との対話形式により流れ図を導出し、ＡＳＬ／Ｆプログラムを生成するものである。システムの構成を図１２に示す。

本システムは大きく四つに分かれており、以下それぞれの部分について述べる。

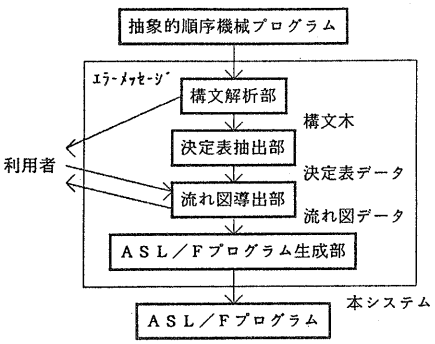


図１２　システムの構成

構文解析部では入力プログラムの文法、及び抽象的順序機械としての正当性を検査しながら構文解析をし、構文木を作る。プログラムに誤りがあればエラーメッセージを表示し、システムは終了する。

決定表抽出部では構文木から決定表のデータ構造を作る。この際、条件の中に補助関数があれば述語に展開する。

流れ図導出部では対話形式により決定表のデータ構造から流れ図のデータ構造を作る。得られるデータ構造は階層構造を成しており、この階層構造は対話形式の概念として用いられる。以下、簡単にデータ構造について述べる。

決定表のデータ構造には決定表の各条件、動作、等の他に、分割の際注目した述語へのポインタ、分割された後のＴ，Ｆ，Ｉの部分の決定表へのポインタ、分割される以前の決定表へのポインタがある。決定表抽出部で得られた決定表には分割される以前の決定表は無いので、分割前のポインタは自分自身を指す。決定表を分割する度にこのデータ構造を作り、ポインタでつなぐ。述語Ｐで分割された後のデータ構造を図１３に示す。

分割により得られたデータ構造は階層構造を成している。また、分割の際注目した述語が真の場合Ｔの部分のポインタをたどり、偽の場合Ｆのポインタをたどり、その後Ｉの部分のポインタをたどることにより流れ図が得られる。

ＡＳＬ／Ｆプログラム生成部では、流れ図合流部の処理、同じ関数定義をする部分の判断、等を行ないＡＳＬ／Ｆプログラムを生成する。

システムの対話形式については次に述べる。

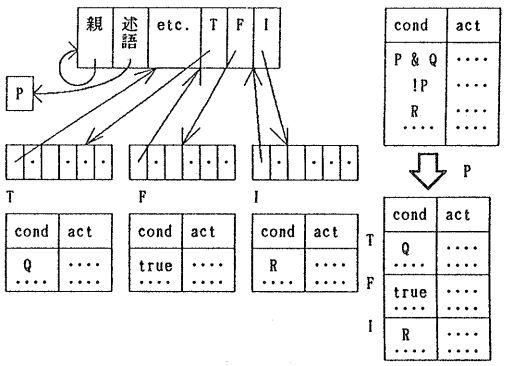


図１３　データ構造（決定表）の階層構造

４．２　コマンド

システムの対話形式の概念として、先に述べた決定表の階層構造がある。以下ではシステムにより抽出された最初の決定表をルート決定表、現在操作している決定表を現決定表、分割されたＴ，Ｆ，Ｉのそれぞれの決定表を子決定表、その逆つまり分割前の決定表を親決定表と呼ぶことにする。システムの対話形式の初期状態はルート決定表のみで、現決定表はルート決定表である。

本システムのコマンド名はＵＮＩＸのコマンドを参考にしている。以下、本システムのコマンドとその機能について述べる。

(1)ｃａｔ

各種の情報の表示を行う。コマンドの引数により表示する情報を指定する。引数として以下の四つがある。

ｉ）ｄｔ（又はｄ）

現決定表を表示する。

ii）ｔｆｉ（又はｔ）

現決定表が分割されていれば、分割の際注目した述語及び分割された状態の各子決定表を表示する。

iii）ｐｒｅｄ（又はｐ）

現決定表の全述語を番号を付けて表示する。

iv）ｆｌｏｗ（又はｆ）

分割された状態を、現決定表をルート決定表とする階層構造として表示する。一種の流れ図としてとらえることができる。

(2)ｍａｘｐ

現決定表の中で最も出現頻度が高い述語に番号を付けて表示する。出現頻度の高い述語から分割するなど、効率のよい流れ図を導出するのに有効である。

(3)ｐｒｅｄ

引数として述語の番号を指定することにより、その述語に注目して現決定表を分割する。

(4)ｃｄ

引数に指定された決定表を現決定表とする。'..' は親決定表を表わし、't', 'f', 'i' はそれぞれの子決定表を表わし、'/' は区切として使われる。例えば '/' はルート決定表を示し、'../t/f' は現決定表の親決定表のＴの部分の子決定表のＦの部分の子決定表を示す。決定表の指定を省略した場合、未分割の任意の決定表が現決定表になる。

(5)ｐｗｄ

現決定表のルート決定表よりの位置関係を表示する。

(6)ｌｓ

現決定表と子決定表のリストを表示する。

(7)ａｕｔｏ

自動的に未分割の決定表を最も出現頻度の高い述語で分割する。最も出現頻度の高い述語が複数個ある場合は、番号の小さい方の述語で分割する。引数によりオプションを指定できる。オプションとしては次の三つがある。

ｉ）／

ルート決定表から全ての決定表を自動的に分割する。

ii）ｍ

iii）ｓ

自動的に分割する際、最も出現頻度の高い述語が複数個ある場合（ｍ）、又は述語が複数個ある場合（ｓ）、その決定表と述語を表示し、入力促進記号'ｐｒｅｄ？'を表示する。これに対して述語の番号を入力するとその述語に注目して決定表を分割し、'ｐ'を入力すると再び決定表と述語を表示し、'ｑ'を入力するとａｕｔｏコマンドをその時点で終了する。オプション'ｍ'又は'ｓ'は同時にオプション　'／'も指定でき、その際オプションは'／ｍ'，'ｓ／'などと指定する。

(8)ｍａｎ

全コマンドの操作方法を表示する。

(9)ｅｎｄ（又はｅ）

対話状態の終了を行う。未分割の決定表がある場合は終了しない。

５．　使用例　　－　バブルソート　－

本システムの使用例を図１４に示す。図４のバブルソートプログラムをそのまま入力として用いている。（ｄｔ）は入力促進記号である。また、利用者の端末からの入力には下線を引いて示してある。

本システムにより生成されたＡＳＬ／Ｆプログラムを図１５に示す。

```
welcome to deriving a flowchart from a decision table
Type 'man' for help ...
======= SPEC Bubble_sort =======
(dt) cat d
-----------------------------------------------------
|             Conditions                |    Actions     |
|---------------------------------------|----------------|
| j=0                                   |  <halt>        |
| ALOAD(a,i)>ALOAD(a,i + 1) & ! (j=0)   | a -> swap(a,i) |
| i<j - 1 & ! (j=0)                     | i -> i + 1     |
| ! (i<j - 1) & ! (j=0)                 | i -> 0         |
| ! (i<j - 1) & ! (j=0)                 | j -> j - 1     |
-----------------------------------------------------
(dt) cat p
*1 j=0
 2 ALOAD(a,i)>ALOAD(a,i + 1)
 3 i<j - 1
(dt) auto
/
/f:j=0
(dt) cat f
start
     t:j=0
          <halt>
     f:j=0
          t:i<j - 1
               i = i + 1
          f:i<j - 1
               i = 0
               j = j - 1
          i:i<j - 1
               t:ALOAD(a,i)>ALOAD(a,i + 1)
                    a = swap(a,i)
(dt) cd f/i/t
(dt) cat d
-----------------------------------------------------
|             Conditions                |    Actions     |
|---------------------------------------|----------------|
| true                                  | a -> swap(a,i) |
-----------------------------------------------------
(dt) pwd
/f:j=0/i:i<j - 1/t:ALOAD(a,i)>ALOAD(a,i + 1)
(dt) end
```

図１４　システムの使用例（バブルソート）

```
SPEC Bubble_sort;

INCLUDE
    ARRAY(TY0,int,50,PF0,PF1);

{ ***** Bubble_sort ***** }

OP
    pbubble_sort : TY0,int -> TY0;
    AX0  : TY0,int,int,
           TY0,int,int -> TY0;
    AX1  : TY0,int,int,
           TY0,int,int -> TY0;
    AF0  : TY0,int -> TY0;

AXIOM
    pbubble_sort(I0,I1) ==
        AX0(I0,0,I1,I0,0,I1);

    AX0(V0,V1,V2,nV0,nV1,nV2) ==
        IF V2=0
        THEN nV0
        ELSE
            IF V1<V2 - 1
            THEN AX1(V0,V1,V2,nV0,V1 + 1,nV2)
            ELSE AX1(V0,V1,V2,nV0,0,V2 - 1);

    AX1(V0,V1,V2,nV0,nV1,nV2) ==
        IF PF0(V0,V1)>PF0(V0,V1 + 1)
        THEN AX0(AF0(V0,V1),nV1,nV2,AF0(V0,V1),nV1,nV2)
        ELSE AX0(nV0,nV1,nV2,nV0,nV1,nV2);

    AF0(V0,V1) ==
        PF1(PF1(V0,V1,PF0(V0,V1 + 1)),V1 + 1,PF0(V0,V1));

END;
    TERM : TY0,int -> TY0;
    TERM (I0,I1) == pbubble_sort(I0,I1);
```

図１５　バブルソートのＡＳＬ／Ｆプログラム

６．　あとがき

本システムは、ＵＮＩＸ４．２ＢＳＤ（機種はＶＡＸ－１１／７５０）上に、Ｃ言語を用いて作成された。ただし構文解析部はｙａｃｃ，ｌｅｘを用いた。システムの大きさは、Ｃ，ｙａｃｃ，ｌｅｘのソースプログラムで合計約５０００行である。

本システムで用いた変換方法は、他の手続き型言語に変換する場合にも適用できる。実際、コード生成部の簡単な変更で、ＰＡＳＣＡＬ，Ｃ等の手続き型言語へ変換可能である。その場合の問題点は、更新される値の関数の、引数の同時評価をどのように実現するかである。この問題点は、変数を評価用と更新用の二種類持っていることにより、容易に解決できると思われる。

謝辞　日頃御指導戴く嵩忠雄教授、適切な御助言を戴いた嵩研究室関浩之氏、ならびに本システムの作成に協力して戴いた山下悦司氏(現　クボタ)に感謝致します。

[参考文献]

[1]Torii,K.Morisawa,Y.Sugiyama,Y.and Kasami, T.: Functional Programming and Logical Programming for the Telegram Analysis Probrem, Proc.of IEEE 7th International Conference on Software Engineering, pp.463-472(1984).

[2]杉山，谷口，嵩：基底代数を前提とする代数的仕様，信学論,Vol.J64-D,No.4,pp.324-331 (1981).

[3][4][5]稲垣、坂部：抽象データタイプの代数的仕様記述法の基礎(1)(2)(3)，情報処理，Vol.25,No.1,No.5,No.7(1984).

[6]井上，関，谷口，嵩：関数型言語ＡＳＬ／Ｆとその最適化コンパイラ，信学論,Vol.J67-D, No.4,pp.458-465(1984).

[7]山下：抽象的順序機械からＡＳＬ／Ｆプログラムへの変換システムの作成(1)，大阪大学基礎工学部特別研究報告(1985)

[8]安松：抽象的順序機械からＡＳＬ／Ｆプログラムへの変換システムの作成(2)，大阪大学基礎工学部特別研究報告(1985)