Title	ファジィ応用のためのプログラミング言語の設計とプログラミング支援環境
Author(s)	上田, 賀一
Citation	茨城大学工学部研究集報(39): 193-201
Issue Date	1991-12
URL	http://hdl.handle.net/10109/7560
Rights	

ファジィ応用のためのプログラミング言語の設計とプログラミング支援環境

上田賀一*

(平成3年8月31日受理)

Design of a Programming Language and Its Environment for Fuzzy Applications

Yoshikazu UEDA*

ABSTRACT—This paper introduces a programming language ALFA for efficiently describing fuzzy applications. The ALFA language is defined as an extension of C language, and it contains a lot of operations and decralations for fuzzy sets and fuzzy inferences. These special functions of the ALFA language are explained. Especially, the design of the symbol table is explained concretely. Futhermore, this paper propose the developoment VETA, which supports the peculiar characteristics of fuzzy theory and the programming by the ALFA language. In development of this environment, four editors, namely, fuzzy set editor, hedge editor, inference rule editor and program chart editor are designed and implemented.

1. はじめに

我々は「曖昧模糊」な言葉を当然のこととして使っている。このような曖昧さを含んだ情報を扱うためにファジィ理論は制御を始めとする様々な分野に応用されている。特に，電化製品に広く利用されたこともあり，社会的な流行語になってしまった。今後も，多くの場面でファジィ理論が活躍すると考えられる。

ファジィ理論を利用する道具として，コンピュータが使われるようになり，このとき，そのソフトウェア開発は重要なものである。しかし，現在多く利用されている一般的なプログラミング言語でファジィ処理を記述すると，例えば「ファジィ集合を定義するときに，そのグレード値をひとつずつ記述する」といった煩わしい作業を必要とする。また，このように表現されたプログラムは，ファジィ集合全体のイメージを捉えにくいため，可読性に欠けてしまう。

本研究では，こうした背景を考え，ファジィ処理をより簡単に実現できるように，ファジィ処理用のプログラミング言語を提案する。これまでにも，ファジィデータを扱えるプログラミング言語あるいはエキスパートシステムがいくつか提案されている[9]-[13]。プログラミング言語の場合，人口知能（AI）用言語として位置づけられており，LISP や Prolog をベースとしたものが多い。また，ファジィの特殊性を考慮した支援環境を持たず，単にプログラミング言語として存在している。一方，エキスパートシステムの場合，ルール記述やファジィデータ入力などのシステム構築のための支援環境は整えられ

*茨城大学工学部情報工学科（日立市中成沢町）
Depratment of Computer and Information Sciences, Faculty of Engineering, Ibaraki University. Hitachi 316, Japan

ている。しかし，エキスパートシステムという性質上，主にファジィ推論ルールだけを記述するプロダクションシステムを採用したものが多く，処理手順をユーザ側で自由に記述することができない。本研究では，知的情報処理に向けての初期段階として，従来の情報処理にファジィデータやファジィ推論能力を加えたものを考え，そのためのプログラミング言語およびその支援環境を構築する。ここで提案するプログラミング言語は，ファジィ処理ができるC言語の拡張型に設計される。従来，様々な情報処理が行われてきたが，C言語が極めて多方面の情報処理に対応できることから，ベースとしてC言語を採用した。また，支援環境はソフトウェア開発を行う際の様々な処理（ファジィ集合の定義，推論ルールの設定など）が並行して行えるように各処理にひとつのウィンドウを割り当て，ウィンドウシステム上で実現する。更に，ソフトウェア開発の効率を善くするための，視覚的側面からの作用も計り，グラフィックスインターフェイスを取り入れ，ユーザが使い易い環境を目指す。

本稿では，まず，ファジィ処理用のソフトウェア開発のためのシステム全体についてその構成を説明する。次に，ファジィ処理用プログラミング言語の特徴について述べる。更に，開発支援環境の仕様と実現について報告する。最後に，本システムのまとめを述べる。なお，本稿では，ファジィ集合論，ファジィ推論を主に取り扱うことになるが，これらを含めファジィ理論全般については説明を省略するので，書籍等[1][2]を参照して戴きたい。

2．システムの概観

本研究で提案するファジィ処理用ソフトウェアの開発支援環境VETA（Visual Environmental Tool for Fuzzy Applications）の構成をFig. 1に示す。この環境はファジィ理論特有の処理を実現するプログラミング言語ALFA（A Language for Fuzzy Applications）を用い，ファジィ処理を行うソフトウェアの作成を簡単に行うための支援環境である。

処理の流れを説明する。ファジィ理論特有の処理を行うためのファジィ集合エディタ，言語ヘッジエディタが用意されている。それぞれ，ファジィ集合の定義，言語ヘッジの定義と確認，推論ルールの設定，およびプログラミング構造の設計を行うものであり，ユーザはこれらを利用してソフトウェアを開発する。支援環境システムは各エディタで作成されたデータをひとつにまとめ，線形リスト構造で表現された内部データの形で保持する。ソースコードジェネレータを使って，このデータからALFA言語によりテキスト表現されたソースプログラムを生成する。更に，このソースプログラムは，ALFAトランスレータを起動することにより，Cソースプログラムに変換される。以上が支援環境VETAの処理の流れである。最終的にユーザは標準的に用意されたCコンパイラを使って，先のCソースプログラムをコンパイルして，実行することができる。

エディタ管理部は，各エディタの相互呼び出しを行えるように管理する機能を持つ。これは支援環境VETAを起動中は常時動作しており，ユーザが各エディタを呼び出すためのボタンを有するウィンドウを呈示する。

支援環境VETAは，Xウィンドウシステム上で実現され，環境構築の使用言語はC言語とし，X Tool kit[6]を利用した。

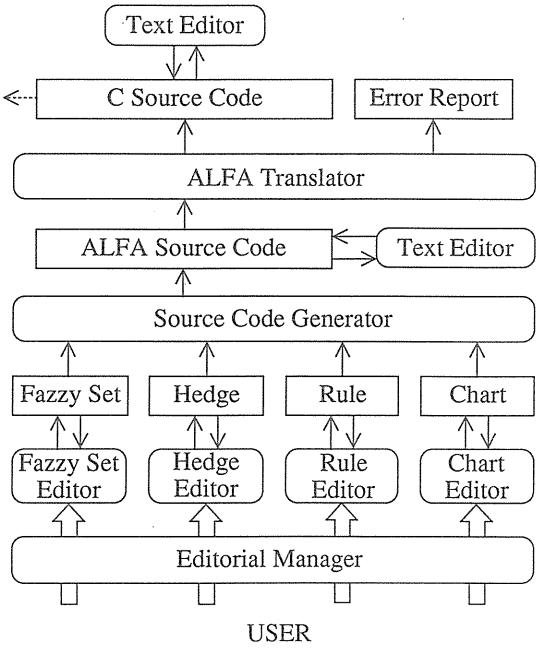

Fig. 1 Outline of development support environment VETA for fuzzy applications.

3．ALFA言語の仕様と実現

3.1 ALFA言語の特徴と仕様

ALFA言語はC言語にファジィ処理のための機能を追加したものである。原語にC言語を取り上げた理由

は，C言語が汎用的であることと，既に広く利用されていることからである。従って，C言語を使うユーザは，ALFA言語の記述の仕方の理解が容易であると言える。ALFA言語を設計する上で，ファジィ処理のために取り入れた機能は，データ構造，ファジィ集合演算およびファジィ集合用の制御構造に大別することができ，それぞれ以下のような特徴を持たせた。

【データ構造】

◇台集合の要素の刻みは一定値でなくてもよい（対数刻みなどをサポートする）。

◇メンバーシップ関数のグレード値は必ずしも，0から1の実数型に限定されず，必要に応じてグレードの型と範囲をユーザ定義できる。

◇グレード値は数値だけでなく，文字型や名前そのものを値として持つことができる。

【ファジィ集合演算】

◇ファジィ集合の演算を，通常の演算のように扱える。また，通常の演算がファジィ集合においても使える。

◇言語ヘッジの定義が行える。

◇予め定義された推論ルールによりファジィ推論が行える。

【ファジィ集合用制御構文】

◇ファジィ集合も通常の集合型データを使った制御構造と同様の記述ができる。

以上のそれぞれの機能を実現するために用意した宣言文は次のようなものである。

【データ構造】

◆台集合定義

suppdef 型名 集合名＝{初期値;最終値;＋＝刻み値};

例：suppdef float age＝{0;100;＋＝1};

◆グレード範囲定義

graddef 型名 {上限，下限} ファジィ集合型名；

例：graddef float{－1，1} newtype;

◆ファジィ集合変数宣言

fuzzy：台集合名 ファジィ集合名＝{グレード，…};

例：fuzzy:age young＝{1.0, 0.9, 0.8, 0.7, …};

グレードが0から1の範囲の場合，予約語fuzzyを使うが，グレード範囲定義のされたファジィ集合型名があればそれを使ってもよい。また，グレード値の初期化部分はなくてもよい。

【ファジィ集合演算】

◆言語ヘッジ定義

hedgedef ヘッジ名；

これを大域宣言し，更に次のような関数定義を行う。

```
hedge ヘッジ名
element;
{
  エレメントに対する処理；
  return（処理結果の値）；
}
```

elementは予約語で，ファジィ集合のひとつの要素を表し，演算は全ての要素に施される。

```
例：hedgedef very;
hedge very
element;
{
  return (element＄2)/*'＄'はべき乗演算子*/
}
```

◆推論ルール定義

ruledef ルール名：モード if ファジィ集合名
then ファジィ集合名；

例：ruledef sample_rule : RM
if large then young;

モードはルールの前件部と後件部からファジィ関係を作る際の演算方式を選択するものである。現在のところ，8種類の算出方式[7]が用意されている。

◆ファジィ推論文

infer ファジィ集合＝
ルール名 {モード，ファジィ集合}；

例：infer fset＝sample_rule {DIRECT, fsetdata};

ここでのモードはルールとファジィ集合の演算を指定するものであり，推論ルール定義のモードとは異なる。現在，推論のために5種類の算出方式[7]と直接法[8]の6モードが用意されている。

【ファジィ集合用制御構文】

◆ファジィ処理用繰返し文

for（ファジィ集合｜＞要素番号i）{…}；

ファジィ集合の全要素において，グレード値が0.5以上のものを属している要素として処理するものである。属するというグレード値の設定を変更する場合は，

for（acut（ファジィ集合，0.7）｜＞要素番号i）{…}

のようにαカット関数を利用する。

3.2 ALFA言語トランスレータの実現

トランスレータはFig. 2に示す構成を取る。トランスレータは，コンパイラ[3][4]と同様に字句解析からコー

ド生成の過程を持つが，コンパイラで行われる最適化の過程は省かれている。本研究では，字句解析部は字句解析生成系の代表的なシステムであるLexを，構文解析部は構文解析生成系の代表的なシステムであるYaccを用いて作成した[5]。これらを使用するにあたっては，言語仕様を明確に記述しなければならず，機能豊富な言語であればあるほどその記述量は膨大になるので，容易ではない作業である。意味解析部およびコード生成部は確立された方法がなく，ファジィ処理に応じた言語処理を施した。各機能に対する意味解析やコード生成の詳細は省略し，全般を通して利用されるシンボルテーブルについて説明する。

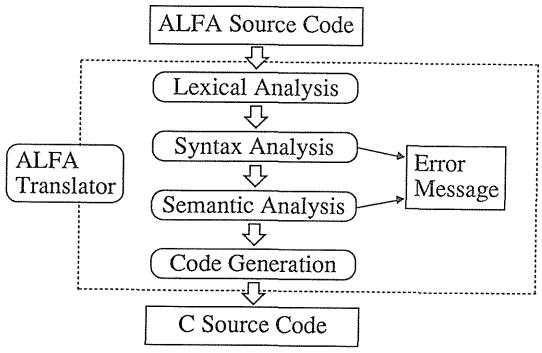

Fig. 2 Construction of ALFA translator.

【シンボルテーブルの構造】

プログラム中に現われる予約語以外のユーザが定義する関数，変数，配列，定数などの様々な名前がシンボルとしてシンボルテーブルに記憶される。プログラム中に出てくるこれらの名前がどの種類のものであるかといったような情報を素早く検索するために，シンボルテーブルが用いられ，シンボルテーブルの適切な設計が後の作業に影響を及ぼすので，充分配慮しなければならない。ALFA言語におけるシンボルテーブルが管理すべき情報には以下のものがある。

・変数名（単純変数，配列変巣，ポインタ変数の区別，配列番号）
・関数名（関数宣言の了／未了，引数）
・台集合名（台集合要素の値）
・ファジィ集合（ファジィ集合のグレード，台集合，ファジィ集合の配列）
・言語ヘッジ関数（ヘッジ定義の有無）
・ファジィ推論ルール（推論ルールを構成するファジィ集合）

シンボルテーブルの設計にあたり，原語であるC言語をカバーするためにあらゆる型の組み合わせに対応できることと，汎用性の高いファジィ処理が行えることを目的とした。そのため，シンボルテーブルを補助する形で，型テーブル，台集合テーブル，ファジィ集合テーブルを用意した。

◆シンボルテーブル

シンボルテーブルは宣言についての種々の情報を保持し，以下のような要素からなる。

(1) 宣言子の種類（変数，関数，引数，未定義，台集合，ファジィ集合，ファジィ関係）を表す定数
(2) 型テーブルへのポインタ
(3) 名前テーブルへのポインタ（台集合テーブルとファジィ集合テーブルへのポインタを共用体でもつ）
(4) 次リストへのポインタ
(5) 関連のある次の宣言子へのポインタ（関数の複数の引数の関係付けなど）
(6) 変数宣言のブロックレベル

◆型テーブル

型テーブルはシンボルテーブルの(2)の要素により指される。各型は以下の要素を鎖状連結することで表される。

(1) 型の種類（ポインタ型，配列型，関数型，構造体型，共用体型，列挙型，型再定義型）を表す定数
(2) 型テーブルへのポインタ（シンボルテーブルへのポインタを共用体でもつ）
(3) 配列の個数を表す整数値

◆台集合テーブル

台集合テーブルはシンボルテーブルの(3)の要素により指される。台集合にはその要素の実体を持たず，初期値と最終値と刻み値と刻み方（線形刻み，対数刻み）を示す演算子で台集合を定義する。以下の要素から構成される。

(1) 名前テーブルへのポインタ
(2) 台集合要素の初期値（double型）
(3) 台集合要素の最終値（double型）
(4) 台集合要素の刻み値（double型）
(5) 台集合の刻み方を決める演算子を表す文字
(6) 台集合の要素数

◆ファジィ集合テーブル

ファジィ集合テーブルは以下の要素からなる。

(1) 名前テーブルへのポインタ
(2) 台集合へのポインタ
(3) ファジィ集合のグレードの実体へのポインタ

(4) ファジィ集合のグレードの文字列を納めた木構造へのポインタ
(5) グレードの個数

◆名前テーブル

名前テーブルは以下の要素からなる。

(1) ハッシュテーブルへのポインタ
(2) 文字列

◆ハッシュテーブル

ハッシュテーブルは以下の要素からなる。

(1) 同じハッシュアドレスを持つ変数の線形リストへのポインタ

以上のテーブルを管理するテーブル管理部は，検索，登録，削除，型の基本操作により先のテーブルを操作する。

コード生成において，ファジィ集合の演算はC言語では関数の形で表す必要がある。そして，演算式の優先順位処理のため，解析木を作成して型合わせを行った後，コードを生成した。ファジィ集合間の型合わせは，台集合を合わせなければならない。そのため，両方の台集合を含むような台集合に合わせることとし，関数内において一時的に作成し，ファジィ集合変数への代入後，解放することとした。

3.3 ALFA言語のプログラム例

ALFA言語により記述したプログラム例をFig. 3に示す。この例は，自動車のハンドルの操作制御のシミュレーションであり，次のような推論ルールの下でのファジィ制御を行う。

「もし，自動車の向きが右向きならば，
ハンドルを左に回せ」
「もし，自動車の向きがまっすぐならば，
ハンドルをそのままにせよ」
「もし，自動車の向きが左向きならば，
ハンドルを右に回せ」

走行中の自動車のある時刻の進行方向に対するずれ(−15〜+15)を入力として受け取る。この値よりルール毎に推論を行い，3つの推論結果から合成したファジィ集合の重心を取ることで最適なハンドルの回転角を求める。このプログラムをトランスレータによりC言語ソースコードに変換したプログラムをFig. 4に示す。トランスレータが作ったコメントを見ることができ，どの部分が変換されたかが理解できる。

```
/* Definition of supports */
suppdef int direct_sup = {0; 30; +=5};
suppdef int handle_sup = {0; 90; +=15};
/* Declaration of fuzzy sets */
fuzzy:direct_sup right    = {0,0,0,0,0.5,1,1};
fuzzy:direct_sup straight = {0,0,0.5,1,0.5,0,0};
fuzzy:direct_sup left     = {1,1,0.5,0,0,0,0};
fuzzy:handle_sup r_turn   = {0,0,0,0,0.5,1,1};
fuzzy:handle_sup mid_turn = {0,0,0.5,1,0.5,0,0};
fuzzy:handle_sup l_turn   = {1,1,0.5,0,0,0,0};
/* Definition of inference rules */
ruledef rule1:RM if right then l_turn;
ruledef rule2:RM if straight then mid_turn;
ruledef rule3:RM if left then r_turn;

main()
{
  int   accel,direction,i;
  float handle;
  /* Declaration of fuzzy set type variables */
  fuzzy:handle_sup hand1;
  fuzzy:handle_sup hand2;
  fuzzy:handle_sup hand3;
  fuzzy:handle_sup fuz_hand;
  accel = 1;              /* Accelerator on */
  direction = input();   /* Get direction to go */
  while(accel) {
    direction += 15;
    if(dirction<0 || 30<dirction) {
      puts("Emergency situation\n");
    }
    /* Inference statements */
    infer hand1 = rule1(DIRECT,direction);
    infer hand2 = rule2(DIRECT,direction);
    infer hand3 = rule3(DIRECT,direction);
    /* Compound of inference results */
    fuz_hand = hand1 \/ hand2 \/ hand3;
    /* Calculation of decision value */
    handle = center(fuz_hand);
    handle -= 45;
    if(handle>=0) {
      printf("Turn right %.0f degree\n",handle);
    }
    else {
      handle = 0-handle;
      printf("Turn left $.0f degree\n",handle);
    }
    accel = go();          /* Get acceleration */
    direction = input();  /* Get direction to go */
  }
}
```

Fig. 3 Sample program by ALFA language

```
#include "fzylib.h"
/*[supportset]
  direct_supp={0,5,10,15,20,25,30};*/
static SUPP_SET direct_supp={'I',0,30,5,'+',7,0};
/*[support set]
  handle_supp={0,15,30,45,60,75,90};*/
static SUPP_SET handle_supp={'I',0,90,15,'+',7,0};
float right_grade[7];
FZY_SET right;
float straight_grade[7];
FZY_SET straight;
float left_grade[7];
FZY_SET left;
float r_turn_grade[7];
FZY_SET r_turn;
float mid_turn_grade[7];
FZY_SET mid_turn;
float l_turn_grade[7];
FZY_SET l_turn;
/*[RULE] rule1 RM IF right THEN l_turn*/
static float rule1_grade[7][7] =
{
  { 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00 },
  { 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00 },
  { 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00 },
  { 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00 },
  { 0.50, 0.50, 0.50, 0.00, 0.00, 0.00, 0.00 },
  { 1.00, 1.00, 0.50, 0.00, 0.00, 0.00, 0.00 },
  { 1.00, 1.00, 0.50, 0.00, 0.00, 0.00, 0.00 }
};
static FZY_SET2 rule1
          ={'F',&direct_supp,&handle_supp,0,0};
/*[RULE] rule2 RM IF straight THEN mid_turn*/
static float rule2_grade[7][7] =
{
  { 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00 },
  { 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00 },
  { 0.00, 0.00, 0.50, 0.50, 0.50, 0.00, 0.00 },
  { 0.00, 0.00, 0.50, 1.00, 0.50, 0.00, 0.00 },
  { 0.00, 0.00, 0.50, 0.50, 0.50, 0.00, 0.00 },
  { 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00 },
  { 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00 }
};
static FZY_SET2 rule2
          ={'F',&direct_supp,&handle_supp,0,0};
/*[RULE] rule3 RM IF left THEN r_turn*/
static float rule3_grade[7][7] =
{
  { 0.00, 0.00, 0.00, 0.00, 0.50, 1.00, 1.00 },
  { 0.00, 0.00, 0.00, 0.00, 0.50, 1.00, 1.00 },
  { 0.00, 0.00, 0.00, 0.00, 0.50, 0.50, 0.50 },
  { 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00 },
  { 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00 },
  { 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00 },
  { 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00 }
};
static FZY_SET2 rule3
          ={'F',&direct_supp,&handle_supp,0,0};
#include "finit.c"
int main()
{
  int accel, i, direction;
  float handle;
  float hand1_grade[7];
  FZY_SET hand1;
  float hand2_grade[7];
  FZY_SET hand2;
  float hand3_grade[7];
  FZY_SET hand3;
  float fuz_hand_grade[7];
  FZY_SET fuz_hand;
/*[fuzzy set] hand1*/
  hand1.f_type = 'F';
  hand1.f_supp1 = &handle_supp;
  hand1.f_grade = hand1_grade;
/*[fuzzy set] hand2*/
  hand1.f_type = 'F';
  hand1.f_supp2 = &handle_supp;
  hand1.f_grade = hand2_grade;
/*[fuzzy set] hand3*/
  hand1.f_type = 'F';
  hand1.f_supp3 = &handle_supp;
  hand1.f_grade = hand3_grade;
/*[fuzzy set] fuz_hand*/
  hand1.f_type = 'F';
  hand1.f_supp1 = &handle_supp;
  hand1.f_grade = fuz_hand_grade;

  Initialize();
  accel=1;
  direction=input();
  while (accel) {
    direction+=15;
    if (direction<0||30<direction) {
      puts("Abnormal situation\n");
      exit(1);
    }
    f_f(Assign,&hand1,d_infer(&rule1,(double)direction));
    f_f(Assign,&hand2,d_infer(&rule2,(double)direction));
    f_f(Assign,&hand3,d_infer(&rule3,(double)direction));
   f_f(Assign,&fuz_hand,f_f(Max,f_f(Max,&hand1,&hand2),&hand3));
    handle=center(&fuz_hand);
    handle-=45;
    if (handle>0) {
      printf("Turn right %.0f degree\n",handle);
    }
    else {
      handle=0-handle;
      printf("Turn left %.0f degree\n",handle);
    }
    accel=go();
    direction=input();
  }
}
```

Fig. 4 Sample program translated to C language

4．支援環境 VETA の仕様と実現

支援環境 VETA では，プログラム構造の設計を行うチャートエディタ以外に，ファジィ処理を行う際に必要となるファジィ集合の定義，言語ヘッジのテスト，推論ルールの設定を行う専用のエディタを設けた。

［1］ファジィ集合エディタ

ファジィ集合エディタはファジィ集合を定義する。Fig. 5 にファジィ集合エディタのウィンドウを示す。これは，メニューボタン，グレード設定領域，台集合設定領域から成る。グレードの定義方法には次の 2 つがある。ひとつは，ファジィ集合をグラフィックスで表し，それを直接マウスで操作することによりグレード値を決定する方法である。もうひとつは，よく利用されるπ，s，z 型のファジィ集合を予め用意しておき，そのパラメータを変える（グラフ上の主要ポイントを移動する）ことにより新しいファジィ集合を実現する方法である。台集合の定義には数値と文字列を使うことができる。数値で設定するときは，初期値，最終値，刻み（変化方法）を指定し，文字列で設定するときは，台集合の各要素をキーボードから入力する。また，新しく定義したグレード値と台集合は別々に登録でき，それらを再利用することができる。

［2］言語ヘッジエディタ

言語ヘッジエディタは，言語ヘッジの編集を行うもので，言語ヘッジは名称となる言語表現とグレード値を加工する計算式から成る。以下の機能が望まれる。

・新しい言語ヘッジを作成できる。
・視覚的な編集ができる。
・ライブラリ登録し，再利用可能とする。
・言語ヘッジの効果を試すため，既存のファジィ集合に作用させ，変化を確認する機能を有する。

名称及び計算式を文字列として入力し，登録することは容易だが，この計算式を解釈することは容易ではない。今回は，既存の言語ヘッジを登録済み扱いとして効率を試すためのヘッジテスターの実現に限った。Fig. 6 に言語ヘッジテスターのウィンドウを示す。図はファジィ集合（点線）に言語ヘッジ very（3．1 節内の言語ヘッジ定義例を参照）を作用させたときの結果（実線）を示している。

［3］推論ルールエディタ

推論ルールエディタでは，if-then 形式の推論ルールを作成する。詳細には以下の機能が望まれる。

・関係行列の作成
これまでに報告された代表的な算出方法でルールの前件部と後件部からファジィ関係を作る。
・推論ルールの試験
作成した推論ルールが入力に対しどのような推論結果を出力するか，プログラムの実行に先だって試験することができる。
・推論ルールの登録
再利用のため，一度作成したルールをライブラリ登録できる。

［4］プログラムチャートエディタ

プログラムチャートエディタでは，他のエディタで作成したデータを受けて，実際のプログラミングを行う。プログラムチャートエディタは以下の複数のウィンドウで構成される。

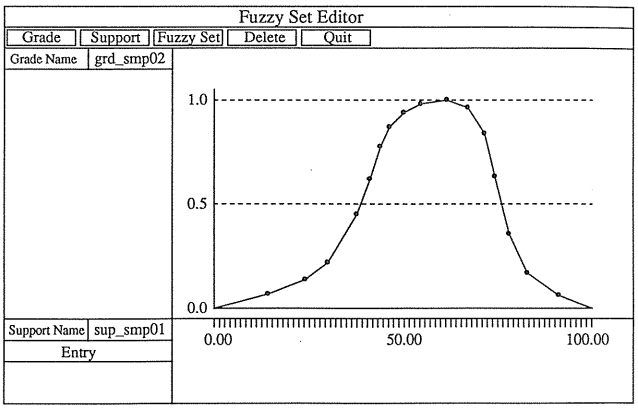

Fig. 5　Window view of fuzzy set editor.

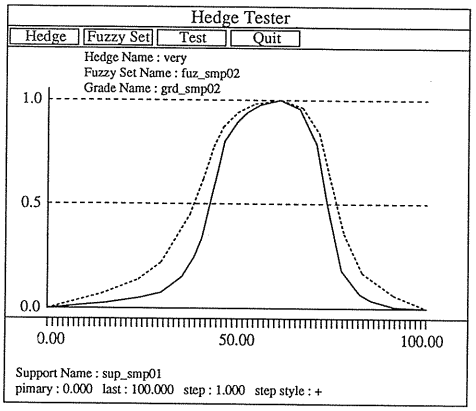

Fig. 6　Window view of hedge tester.

・関数作成ウィンドウ
プログラム構造を理解しやすくするために，プログラムの図式化技法（YACⅡ）の改良したものを用いてプログラム構造を記述する。Fig. 7 に各基本構造の図式を示す。関数毎にウィンドウが割り当てられる。

・外部定義ウィンドウ
プログラム作成時の外部定義となるインクルードファイル設定，マクロ宣言，変数型宣言，大城変数宣言を行うためのウィンドウである。

・変数宣言ウィンドウ
局所変数を宣言するためのウィンドウで，関数毎に割り当てられる。

・コメント編集ウィンドウ
プログラム図式中の注釈を編集するためのウィンドウである。

Fig. 8 にプログラムチャートエディタの主要部分である関数作成ウィンドウを示す。内部にはメニューボタン，チャート図の制御記号と操作ボタン，および作成中のプログラムチャートが示されている。

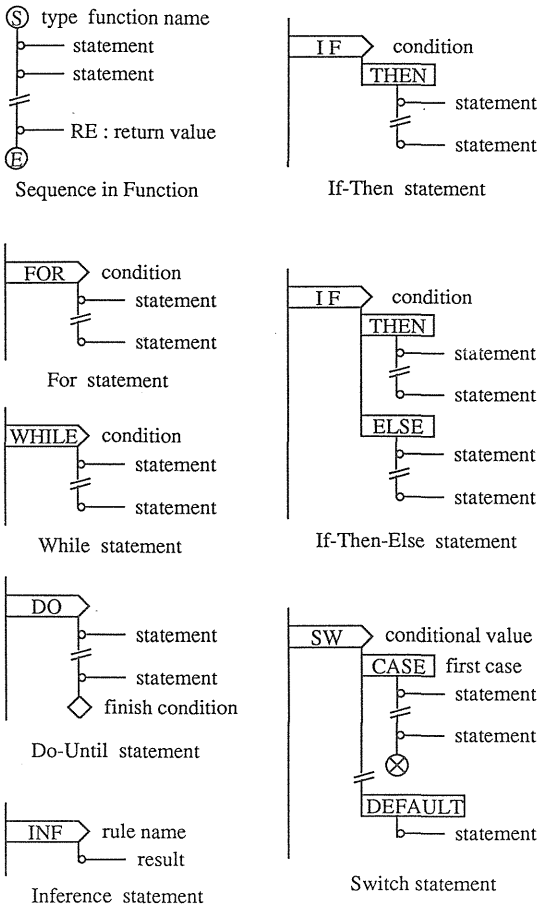

Fig. 7 Primitive structures of programming chart.

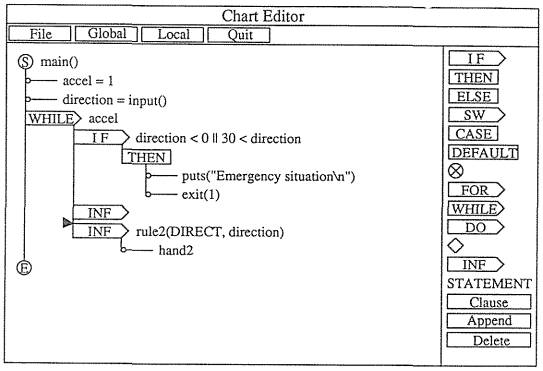

Fig. 8 Window view of chart editor.

5. まとめ

本研究では，ファジィ集合とその演算が扱え，ファジィ推論も行えるファジィ処理用プログラミング言語 ALFA の仕様設計とトランスレータの開発，およびその支援環境 VETA の設計と部分的開発を行った。

本システムは，これまでのファジィプログラミング言語やファジィエキスパートシステムとは異なり，知的情報処理に向けて，従来の情報処理に加えてファジィ情報やファジィ推論の扱えるプログラミング言語とその支援環境をまとめたものとなっている。

言語仕様については，その設計を重視してあり充実したものとなっている。特に，ファジィ集合およびファジィ推論の定義やその利用は容易である。トランスレータの実現においては，様々な型およびそれらの組み合わせに対応できるように，シンボルテーブルの構造を実現した。かなり複雑なものとなったが，ファジィ集合も他のデータ型と組み合わせることができ，ファジィ集合をより一般的なデータとして扱えるように実現することができた。

支援環境については，ファジィ理論特有の処理を設計するために，ファジィ集合の作成，言語ヘッジのテスト，ファジィ推論ルールの設定，プログラム構造の設計の4つのエディタを設計し，作成した。ファジィ集合エディ

タでは，視覚的な確認を行いながらファジィ集合の記述ができるので，作成はかなり簡便になった。また，言語ヘッジテスターにより，言語ヘッジの効果を予め調べることができるようになった。チャートエディタでは，本研究で改良した図式化技法を用い，プログラム構造を容易に作成，理解できるようにした。ルールエディタは作成できなかったが，その設計仕様は，前件部と後件部からの関係行列の作成および推論ルールのテストを取り入れたものである。全般的には，ユーザインタフェースを充分考慮した支援環境を設計したが，更にエディタ間の関係を容易に扱える機能が望まれると考える。

謝　辞

本研究を進めるに当たり御協力頂いた名古屋工業大学電気情報工学科の石井直宏教授，犬塚信博君，中谷浩人君，小川健二君，小山修司君に感謝致します。

参考文献

(1) 水本雅晴：ファジイ集合とその応用，サイエンス社 (1988)

(2) 寺田寿郎，浅居喜代治，菅野道雄：ファジィシステム入門，オーム社 (1987)

(3) A. V. Aho, J. D. Ullman, 土居範久訳：コンパイラ，培風館 (1986)

(4) 疋田輝雄，石畑清：コンパイラの理論と実現，共立出版 (1988)

(5) A. T. Schreiner, H. G. Friendman, 矢吹道郎訳：Cコンパイラ設計 yacc lex の応用，啓学出版 (1987)

(6) D. A. Young, 川手恭輔：X Toolkit プログラミング，トッパン (1990)

(7) 水本雅晴：新しい推論の合成規則の下でのファジィ推論，電子情報通信学会論文誌，Vol. 65-D, No. 11, pp. 1319-1325 (1982)

(8) 廣田薫：ファジィ推論エキスパートシステムの現状動向，情報処理学会誌，Vol. 28, No. 8, pp. 1065-1074 (1987)

(9) 遠藤経一，石井実薫：ファジィエキスパートシステム構築シェル，情報処理学会誌，Vol. 30, No. 8, pp. 948-956 (1989)

(10) J. Bowen, J. Kang : FMUFL : A Fuzzy Multi-Paradigm Language, Fuzzy Sets and Systems, Vol. 34, pp. 263-291 (1990)

(11) Z. A. Sosnowski : FLISP-A Language for Processing Fuzzy Data, Fuzzy Sets and Systems, Vol. 37, pp. 23-32 (1990)

(12) J. J. Buckley, W. Siler, D. Tucker : FLOPS, A Fuzzy logics in Knowledge Engineering, Verlag TUT Rheinland, Koln (1986)

(13) K. S. Leung, W. Lam : Fuzzy Concepts in Expert Systems, Computer, Vol. 21, No. 9, pp. 43-59 (1988)