ファジィ・プロダクション・システムの拡張

7C-5

－ 作業記憶の分割と複数ルールの実行 －

馬野 元秀 (大阪大学 大型計算機センター)
江澤 義典 (関西大学 工学部)
中嶋 直紀 (関西大学 工学部)

1. はじめに

通常のプロダクション・システムにファジィ集合の概念[1]を取り入れて、あいまいな知識を表現できるようにしたファジィ・プロダクション・システムについては、すでに報告した[2]。本報告では、このシステムをさらに拡張して、作業記憶を分割して扱える機能、指定した数のマッチしたルールを実行できる機能を追加した。

2. 従来のシステムの概要

ファジィ・プロダクション・システムは、データや条件部が持つあいまいさと確信度的なあいまいさを表現することができ、ファジィ・パターン・マッチングにより推論を進めていく。このシステム[2]は、UNIX 4.2 BSD の Franz Lisp 上で実現されていた。

[例1] ファジィ・プロダクション・システムの例

```
(predicates
  (short  (x) (z  x 155 170))
  (middle (x) (pi x 150 170 190))
  (tall   (x) (s  x 160 180)) )
(terms
  (true      {0.3/0.8, 0.7/0.9, 1/1})
  (about150 {0.4/145, 1/150, 0.4/155}))
(working-memory  {true/(John about150)})
(rules rb1
 (r1 if (=x short(=y))
     then (write =x is short(=y) with =match)
          (stop))
 (r2 if (=x middle(=y))
     then (write =x is middle(=y) with =match)
          (stop))
 (r3 if (=x tall(=y))
     then (write =x is tall(=y) with =match)
          (stop)) )
```

<実行結果>

```
John is short (about150) with
                          {0.3/0.8, 0.7/0.9, 1/1}
***** stop *****
```

①ルールの条件部に書くファジィ述語 short、middle、tall は predicates で定義する。ただし、関数 z、pi、s は図1のようなメンバーシップ関数に対応するものと考える。②作業記憶内で用いるファジィ集合は terms

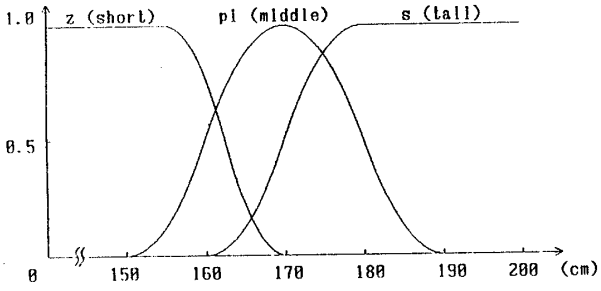

図1. 関数 z、pi、s

で定義する。作業記憶に設定されている true/(John about150) は、ファジィ値 about150 とファジィ真理値 true を持つデータである。③ルールは rules で定義される。ここで、=x と =y は変数であり、作業記憶内のデータにマッチした値が代入される。

述語付き変数 (ルール r1 の例では short(=y)) の場合にはマッチしたデータを用いてその述語の真理値が計算される。また、条件全体としての真理値は予約変数 =match に保存される。例1では、最もよくマッチしたルール r1 の動作部が実行され、その結果が出力されている。

3. 作業記憶の分割化

従来のシステムでは、ルール・ベースの分割機能は実現されていたが、すべてのデータは1つの作業記憶に記述しなければならないという制約があった。そのため、いろいろな違った種類のデータが作業記憶内に混在する場合にはルールの記述が煩雑になるという欠点があった。また、関係のないデータまでマッチングの対象となっていたので実行効率の悪いときがあった。そこで、作業記憶をいくつかのブロックに分けて取り扱えるように拡張した。

この拡張により関連のあるデータごとに作業記憶を設定でき、データのモジュール化が実現できたうえに、マッチングの効率が良くなった。

Extentions of Fuzzy Production System Modularized Working Memory and Multiple Rules Firing
Motohide UMANO[1], Yoshinori EZAWA[2], Naoki NAKAJIMA[2]
[1]Osaka University [2]Kansai University

[例２] 複数の作業記憶を使った例

```
(predicates
  (short  (x) (z  x 155 170))
  (middle (x) (pi x 150 170 190))
  (tall   (x) (s  x 160 180)) )
(terms
  (true      {0.3/0.8, 0.7/0.9, 1/1})
  (false     {1/0, 0.7/0.1, 0.3/0.2})
  (about60   {0.4/55, 1/60, 0.4/65})
  (about70   {0.4/65, 1/70, 0.4/75})
  (about150 {0.4/145, 1/150, 0.4/155})
  (about170 {0.4/165, 1/170, 0.4/175})
  (about190 {0.4/185, 1/190, 0.4/195}) )
(working-memory people {Japanese})
(working-memory American {Bob, John, Mike})
(working-memory Japanese {Hanako, Taro})
(working-memory Bob {0.9/(height about170),
                true/(weight about60),(age 21)})
(working-memory John {true/(height about150),
                   1/(weight about60),(age 23)})
(working-memory Mike {false/(height about190),
                  0.4/(weight about60),(age 10)})
(working-memory Hanako {0.8/(height about150),
                   1/(weight about60),(age 20)})
(working-memory Taro {true/(height about170),
                  0.9/(weight about70),(age 24)})
(rules rb0
  (r0 if (*absent =) then (stop))
  (r1 if =x
      then (fprint =x)(fterpri)(delete =x 1)
           (push-wm =x)(push-rb rb1)) )
(rules rb1
  (r0 if (*absent =) then (pop-wm)(pop-rb))
  (r1 if =x
      then (fprint =x)(fprinc " ")
           (delete =x 1)
           (push-wm =x)(push-rb rb2)) )
(rules rb2
  (r1 if (height short(=x))
      then (write is short(=x) with =match)
           (pop-wm)(pop-rb))
  (r2 if (height middle(=x))
      then (write is middle(=x) with =match)
           (pop-wm)(pop-rb))
  (r3 if (height tall(=x))
      then (write is tall(=x) with =match)
           (pop-wm)(pop-rb)) )
```

<実行結果>

```
Japanese
Hanako is short (about150) with {0.8}
Taro is middle (about170) with
                {0.3/0.8, 0.4/0.875, 0.7/0.9, 1/1}
***** stop *****
```

作業記憶は名前を付けて区別できる。例えば、作業記憶 people にはデータ Japanese が入っている。ルール・ブロック rb0 のルール r1 では people に入っているデータ Japanese を取りだし、push-wm によってカレントの作業記憶名 people を保存し、作業記憶 Japanese に移り、 push-rb でルール・ブロック名 rb0 を保存した後、ルール・ブロック rb1 に移る。また、ルール r0 では作業記憶のデータの有無を調べている。ルール・ブロック rb1 では作業記憶のデータがなくなった時、 保存した直前の作業記憶 people に戻り、ルール・ブロックも保存した直前のルール・ブロック rb0 に戻る。 ルール・ブロック rb2 ではデータ中の height のファジィ値とファジィ述語 short、middle、tall と最もよくマッチしたものを表示している。

4. 複数個のルールの実行

従来のシステムでは、マッチング後のルールの実行は最もよくマッチするものを１つ選んでいた。そのためファジィ制御の場合のようにそれぞれのルールの推論結果を結合することができなかった。そこで、複数ルールの実行を可能とし、それぞれのルールの推論結果を結合できるようにした。

実行するルールの指定方法は２通りあり、マッチ度の順に個数で指定する方法とマッチ度の下限値を指定する方法とがある。

[例３] 例１で実行ルール数 3 と指定した場合

```
John is short (about150) with
                {0.3/0.8, 0.4/0.875, 0.7/0.9, 1/1}
John is middle (about150) with {1/0, 0.4/0.125}
***** stop *****
```

例１ではマッチ度が最大のルール r1 のみを実行していたが、例３では実行ルール数を指定したので、よりマッチ度の低いルール r2 の結果も出力されている（ルール r3 にはまったくマッチしない)。 この例では推論結果を結合していないが、ルールの動作部で関数 update を使うことにより推論結果を結合できるのでファジィ制御のシミュレートも可能である。

5. おわりに

今後は、本システムを用いて実際にエキスパート・システムを作成し、その経験からのフィードバックによりさらにシステムの充実を図りたい。

[謝辞]

本研究を行なうに当たり、いろいろとご配慮を頂いている関西大学の伊藤郁男教授、久保井不二男教授に深く感謝致します。

[参考文献]

[1] L.A. Zadeh:"Fuzzy Sets," Information and Control, Vol.8, pp.338-353 (1965).

[2] 馬野:「ファジィ・プロダクション・システムとその実現」、SICE 関西支部シンポジウム「あいまい情報処理と知的システム制御」、pp.75-80 (1988).

[3] 中嶋:「ファジィ・プロダクション・システムの拡張」、関西大学・工学部・卒業論文(1989).