エキスパートシステムにおける
あいまい性の取扱いに関する研究

平成６年

三好　力

内容梗概

この論文は、知識表現、推論方法、開発環境の観点からファジィエキスパートシステム開発支援ツール LIFE FEShell に関する研究を行ない、知識獲得の観点からプロダクションルールの結合演算子の自動チューニングに関する研究を行なうことによって、エキスパートシステムにおけるあいまい性の取扱いに関して得られた知見をまとめたものである。

LIFE FEShell は、ファジィプロダクションシステム、ファジィフレームシステム、オブジェクトエディタの 3 つから構成される。ファジィプロダクションシステムは、ルールの条件部内におけるファジィな条件、ルールの条件部と結論部の間の関係のあいまい性およびデータのあいまい性が同時に出現する状況を取扱うことができる。ルールの条件部内の条件の間の関係として、ファジィ測度で定義される複雑な関係も取扱える。これにより、ファジィプロダクションシステムは広範囲な悪構造問題にも適用可能である。ファジィフレームシステムは、フレーム間の関係を利用してあいまい性の定義の継承、階層的管理、推論などの枠組みを提供している。これにより、あいまい性を言語的に表現する時に、同一の言葉に対して複数の定義を可能にし、対象に応じて適切な定義を採用することができる。ファジィプロダクションシステムとファジィフレームシステムは相互に呼び合うことができ、相互に協調しながら問題解決を行なうエキスパートシステムを構築することができる。これによって、より自然な形で知識表現や推論を実現することができる。オブジェクトエディタは LIFE FEShell の開発環境として作成されたもので、X ウインドウによる GUI (Graphical User Interface) を実現している。

ファジィルールの結合演算子の自動チューニングについては、プラントネットワークとコントローラネットワークからなるシステムを用いて、ファジィコントローラの結合演算子に、パラメータ付き T オペレータを採用し、定義した誤差をもとに、

バックプロパゲーションを利用してこのパラメータの自動チューニングを行なう方法を提案する。さらに、計算機によってシミュレーションを行ない、提案したシステムを用いて自動チューニングが可能であることを示す。パラメータ付きTオペレータによって典型的なTオペレータの強さの中間にある最適な演算を得ることができる。

目　次

第1章　序論

1.1 研究の背景

知識工学[1]は、専門知識が重要な役割を果たす分野へ人工知能を応用する技術として、Stanford大学のE. A. Feigenbaumによって提唱された。その中心となる技術には知識表現、知識利用、知識獲得がある[2]。知識表現は、特定領域の専門家の持つ知識を計算機上に表現する方法に関する技術である。知識利用は、計算機上に表現された専門知識を利用して問題を解決するための技術である。知識獲得は、専門家の持つ知識を整理・編集してシステムに取り込む技術である。エキスパートシステムを実際に構築する場合には、以上の3つの技術の他に、プログラム言語や開発環境も重要である。

知識表現の方法としては、これまでにプロダクションルール、フレーム、意味ネットワーク、述語論理などによる方法が研究されている。また、人間の持つ知識やデータを表現するために、あいまい性の取扱いが避けがたいという認識に基づいて、確率、ファジィ論理[3-6]、可能性理論[7-9]などを利用した知識表現も可能となっている。

知識利用は推論の研究が中心であり、前向き推論や後向き推論が研究されている。あいまいな知識やデータの取扱いが可能な推論方法の拡張も提案されている。

知識獲得は、現段階では、知識工学の技術者が専門家にインタビューすることによって行なっているが、これには多大の労力が必要である。知識獲得が困難である理由の一つとして、専門家の知識に含まれるあいまい性に起因するものがある。このあいまい性は、人間の知的活動、すなわち、対象の認識、意味理解、概念形成、意味や概念の表現などと深くかかわっており、今後、さらに研究が行なわれるべき

であると考えられる。知識の自動獲得のアプローチとしては、ニューラルネットワーク [10, 11] や遺伝的アルゴリズム [12-14] などを用いる方法が代表的である。この方法はデータから自動的に近似関数を得られる可能性があるが、得られた結果に対する解釈や理解が非常に難しいという欠点がある。

開発環境については、ワークステーション上で開発されたシステムには、ウインドウベースの GUI (Graphical User Interface) を持っているものが多い。

このような状況において、知識表現、推論方法、開発環境の観点からファジィエキスパートシステム開発支援ツールに関する研究を行ない、知識獲得の観点からプロダクションルールの自動チューニングに関する研究を行なった。この論文は、これらの研究を通して得られたエキスパートシステムにおけるあいまい性の取扱いに関する新しい知見をまとめたものである。

1.2 研究の目的

エキスパートシステムを構築する場合に、あいまいな知識やデータを取扱うことが避けられないという認識に基づいて、あいまいな知識やデータを取扱うことのできるファジィエキスパートシステム構築支援ツール(以下「ファジィシェル」と呼ぶ)の研究開発が各所で進められている [15-25]。あいまい性の取扱いを許すことによって、ルールの数を減少させ、保守性を向上させることが期待できる。さらに、想定されなかった入力があった場合にも、近似推論によって近似解を得ることができる可能性がある。具体的な応用として、心臓の超音波診断 [18]、財務分析 [20] などが報告されている。

しかし、これらのシステムは一般のエキスパートシステム構築支援ツール (以下「従来のシェル」と呼ぶ) に比べて利用できる知識表現の形態や開発環境などの面で不十分なものが多い。具体的には、従来のシェルでは知識表現としてプロダクションルールに加えてフレームなどをサポートしているが、ほとんどのファジィシェルではプロダクションルールのみしか備えていない。開発環境、すなわち、知識編集やデバッグ・チューニングの支援ツールの面においても、従来のシェルの多くがワークステーション上でウインドウベースの GUI を持っているのに対して、ファジィシェルではそのような環境を持っているものはほとんどない。

また、ファジィシェルにおける最も重要な機能であるあいまい性の取扱いに関しては、プロダクションルールの一部を拡張してあいまい性を知識表現に取り込み、それを取扱うことができるように推論方法を拡張しているものがある。しかし、取扱えるあいまい性は一種類で、あいまい性を分析し、性質の異なった複数のあいまい性を同時に取扱えるようにしたファジィシェルはほとんどない。特に、プロダクションルール中のファジィ条件に加えて、例えば「身長が 170cm 位」といったデータのあいまい性を取扱えるものは見当たらない。これは、現在のところどのような種類のあいまい性を扱うのが適当であるかがまだ明確になっていないということにも原因の一つがあると考えられる。

知識獲得に関しては、ファジィプロダクションルールのファジィ条件のメンバーシップ関数を自動的にチューニングする方法が既に多く研究されている [26, 27] 。メンバーシップ関数だけでなく、条件を結合する結合演算子、インプリケーション演算子、コンビネーション演算子を総合的にチューニングすることによって、人間がプロダクションルールを用いて知識表現する場合に、どの部分が不自然になっているかを知ることができる。これにより、プロダクションシステムにおける知識獲得をかなり改善できる可能性がある。

この論文では、ファジィシェルが抱える問題点の解決と知識獲得の改善を目的として、ファジィエキスパートシステム開発支援ツール LIFE FEShellの研究開発 [28-41] と、ファジィプロダクションルールの結合演算子のバックプロパゲーションによる自動チューニングの研究 [42-44] について述べる。

1.3 研究の概要

前節の議論より、ファジィシェルの課題として次のものがあげられる。

(課題 1) 複数の知識表現

(課題 2) 複数のあいまい性の取扱い

(課題 3) 開発環境の充実

(課題 4) 推論エンジンの柔軟性の確保

(課題 5) 知識獲得の改善

それぞれの課題に対して次のような研究を行なった。

課題1に関しては、LIFE FEShellではファジィプロダクションルールとファジィフレームの取扱いが可能である。第3章でファジィプロダクションルールについて、第4章でファジィフレームについて述べる。

課題2に関しては、第3章でファジィプロダクションルールに、第4章でファジィフレームに現れるあいまい性の分析とその取扱いについて述べる。

課題3に関しては、第5章でLIFE FEShellの開発環境について述べる。

課題4に関しては、第5章で推論エンジンの柔軟性を損なわないインプリメント方法について述べる。

課題5に関しては、第6章でファジィプロダクションルールの自動チューニングについて述べる。

LIFE FEShell [30-32, 36, 41] は、ファジィプロダクションシステム (FPS) [28, 39, 40]、ファジィフレームシステム (FFS) [29, 34, 35]、オブジェクトエディタ (OE) [33, 37, 38] から構成される。図1.1にLIFE FEShellの構成を示す。

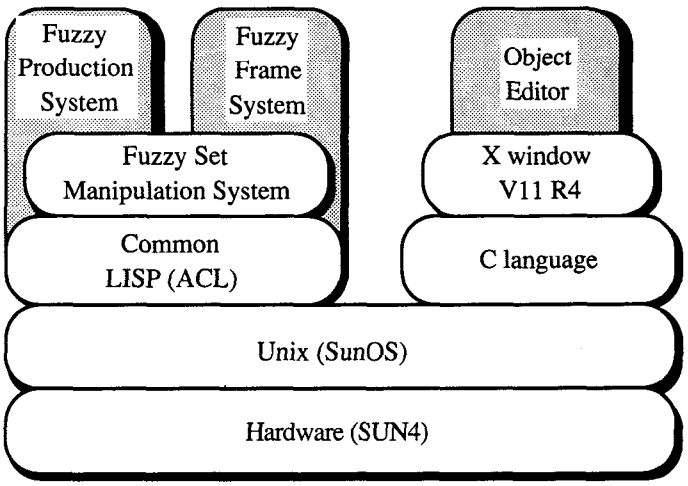

図1.1 : LIFE FEShellの構成

FPSとFFSは一般のプロダクションシステムとフレームシステムにファジィ述語や可能性分布を用いてあいまいな知識の表現ができるように拡張を行なったものである。LIFE FEShellでは、FPSとFFS双方の長所を取り入れるため、これらは相互に呼び合うことができ、FPSとFFSが相互に協調しながら問題解決を行なうエキスパートシステムを構築することができる。これによって、より自然な形で知識表現と推論を実現することができる。FPSとFFSは実験研究用の推論エンジンであるこ

とを考慮し、Allegro Common Lisp 上で開発を行なった。ファジィ集合処理は馬野のファジィ集合処理システム (FSMS) [45] を利用している。

OE は LIFE FEShell の開発環境として開発されたもので、C 言語を用い、X ウインドウを使って GUI を実現している。推論エンジンと OE の間には、双方向の複雑な情報交換が発生するため、Allegro Common Lisp と C 言語との間で通信を行なう。

我々は LIFE FEShell を用いて外国為替取引支援エキスパートシステム FOREX [46-48] を開発した。FOREX の知識の規模を以下に示す。

・ルール数約 5,000 ブロック (ステップ数約 50,000 行)

・内部状態数約 300 フレーム (ステップ数約 6,000 行)

ただし、FOREX では内部状態の表現にフレームを用いている。FOREX はファジィシェルを利用したシステムとしては非常に大きく、このようなシステムのインプリメントができたことからも LIFE FEShell の有効性が確認されたと考える。

この論文の構成は以下のようになっている。

第 2 章では、あいまい性を取扱うための理論として、ファジィ論理、T ノルムと T コノルム、ファジィ集合論、ファジィ測度とファジィ積分、可能性理論を概説する。

第 3 章では、ファジィプロダクションルールに関して、まずプロダクションシステムの枠組みの中で取扱えるあいまい性について整理し、既存のファジィプロダクションシステムでは、これらのあいまい性を同時には取扱うことができないことを指摘する。そして、LIFE FEShell FPS におけるあいまい性の表現と取扱い機能について述べる。

第 4 章では、ファジィフレーム表現に関して、フレームに関する基本的な考え方と、フレームシステムに現れるあいまい性について分析し、LIFE FEShell FFS におけるあいまい性の表現と取扱い機能について述べる。

第 5 章では、まず、従来のシェルとファジィシェルに関して開発環境の観点から分析を行ない、問題点および解決案と LIFE FEShell 開発環境の設計目標を示す。次に推論エンジンが取扱う各オブジェクトの開発環境を用いた編集の実際と開発環境の詳細について述べる。さらに、推論エンジンと開発環境をインプリメントする際に生ずる問題点の分析を行ない、推論エンジンの柔軟性を損なわない有効なインプリメント方法を提案する。

第6章では、ファジィプロダクションルールの結合演算子のチューニングに関して、結合演算子にパラメータによって性質が変化するTノルム、Tコノルムを採用し、結合演算子の自動チューニングに関する手法を提案し、シミュレーションによって評価する。

第7章では、各章で得られた結果と今後の展望を述べる。

第２章　あいまい性を扱う理論

2.1 はじめに

この章では、あいまい性を扱うための理論として、ファジィ論理[3-6]、Ｔノルムとコノルム[5]、ファジィ集合論[49, 5]、ファジィ測度とファジィ積分[50-52]、可能性理論[7-9]について概説する。

2.2 ファジィ論理

「ファジィ論理」という言葉は特定の論理体系を指す言葉ではなく、真理値空間として区間 [0, 1] あるいはその上におけるファジィ集合の族をとる論理の総称として使われている。ここではその内の一つで、次に示す真理値計算体系を「ファジィ論理」[3-6]と呼ぶ。

真理値空間：区間 [0, 1] とする。

論理演算：命題 p、q に対し区間 [0, 1] の値をとる真理値を $v(p)$、$v(q)$ で表すとすると、and ($\wedge$)、or ($\vee$)、not ($\neg$) は次のように定義される。

$$v(p \wedge q) = T\big(v(p), v(q)\big)$$

$$v(p \vee q) = S\big(v(p), v(q)\big)$$

$$v(\neg p) = 1 - v(p)$$

ここで、T はＴノルムを、S はＴコノルムを表す。

2.3 T ノルムと T コノルム

T ノルム (T-norm) とは、通常の「and」演算子の拡張で、次の条件を満たす演算 $T(x, y)$ の総称である。

$$T(x,1)=x,\quad T(x,0)=0$$

$$x_1 \le x_2, y_1 \le y_2 \quad \Rightarrow \quad T(x_1,y_1) \le T(x_2,y_2)$$

$$T(x,y)=T(y,x)$$

$$T(x,T(y,z))=T(T(x,y),z)$$

T コノルム (T-conorm) は T ノルムとは双対な演算であり、通常の「or」演算子の拡張で、次のように定義される。

$$S(x,y)=1-T(1-x,1-y)$$

T ノルムと T コノルム (以下 T ノルムと T コノルムを総称して「T オペレータ」と呼ぶ) は既に色々なものが提案され、その性質が研究されている [5]。典型的な T オペレータを表 2.1 に示す。

表 2.1: 典型的な T オペレータ

N	$T_N(x,y)$	$S_N(x,y)$
1	$\min(x,y)$	$\max(x,y)$
2	xy	$x+y-xy$
3	$\max(x+y-1,0)$	$\min(x+y,1)$
4	$xy/(x+y-xy)$	$(x+y-2xy)/(x-xy)$
5	$\begin{cases} x, & \text{if } y=1 \\ y, & \text{if } x=1 \\ 0, & \text{otherwise} \end{cases}$	$\begin{cases} x, & \text{if } y=0 \\ y, & \text{if } x=0 \\ 1, & \text{otherwise} \end{cases}$

ここで、T_1 は論理積、S_1 は論理和、T_2 は代数積、S_2 は代数和、T_3 は限界積、S_3 は限界和、T_4 は Hamachar 積、S_4 は Hamachar 和、T_5 は激烈積、S_5 は激烈和と呼ばれる。これらの T オペレータの間には演算結果に次のような関係がある。

$$T_5 < T_3 < T_2 < T_4 < T_1,\quad S_1 < S_4 < S_2 < S_3 < S_5$$

また、パラメータによって性質が変化するTオペレータ(以下パラメータ付きTオペレータと呼ぶ)も数多く提案されている[5]。以下にパラメータ付きTオペレータの例を2つ示す。

(1) Dombi のTオペレータ

Dombi のTオペレータは次のように定義される。

$$T_D(p,x,y)=\frac{1}{1+\sqrt[p]{\left(\frac{1-x}{x}\right)^p+\left(\frac{1-y}{y}\right)^p}}$$

$$S_D(p,x,y)=\frac{1}{1+\sqrt[p]{\left(\frac{x}{1-x}\right)^p+\left(\frac{y}{1-y}\right)^p}}$$

パラメータ p に対して以下の4つの性質が知られている。

1) $p>0$

2) $p\to 0$ の極限では、 $T_D=T_5, S_D=S_5$

3) $p=1$ では、 $T_D=T_4, S_D=S_4$

4) $p\to\infty$ では、 $T_D=T_1, S_D=S_1$

従って、Dombi のTノルムは、パラメータによって論理積(T_1)から激烈積(T_5)まで、Tコノルムは論理和(S_1)から激烈和(S_5)まで性質が変化する。

(2) Hamachar のTオペレータ

Hamachar のTオペレータは次のように定義される。

$$T_H(p,x,y)=\frac{xy}{p+(1-p)(x+y-xy)}$$

$$S_H(p,x,y)=\frac{x+y-xy-(1-p)xy}{1-(1-p)xy}$$

パラメータ p に対して以下の4つの性質が知られている。

1) $p\geq 0$

2) $p=0$ では、 $T_H=T_4, S_H=S_4$

3) $p=1$ では、 $T_H=T_2, S_H=S_2$

4) $p\to\infty$ では、 $T_H=T_5, S_H=S_5$

従って、HamacharのTノルムは、パラメータによってHamachar積(T_4)から激烈積(T_5)まで、Tコノルムは Hamachar和(S_4)から激烈和(S_5)まで性質が変化する。

2.4 ファジィ集合論

ファジィ集合[49]はファジィ論理を基にした集合論であり、任意のファジィ述語に対応するファジィ集合を定義することができる。例えば、P を $X_1 \times X_2 \times \cdots \times X_n$ 上で定義されたファジィ述語とすると、対応するファジィ集合 A は次のように表現できる。

$$A = \left\{ \mu_A(x_1, \cdots, x_n)/(x_1, \cdots, x_n) \middle| \right.$$
$$\left. \mu_A(x_1, \cdots, x_n) = v\big(P(x_1, \cdots, x_n)\big), (x_1, \cdots, x_n) \in X_1 \times \cdots \times X_n \right\}$$

ここで、$\mu_A(x_1, \cdots, x_n)$ をファジィ集合 A のメンバーシップ関数という。

拡張原理[3]は、任意の関数 $f: X \to Y$ を X のファジィ集合 A から Y のファジィ集合 B への写像に一般化する。関数 $f: X \to Y$ と次のような X 上のファジィ集合 A が与えられた場合、

$$A = \{\mu_1/x_1, \mu_2/x_2, \cdots, \mu_n/x_n\}$$

拡張原理を用いると、

$$f(A) = f\big(\{\mu_1/x_1, \mu_2/x_2, \cdots, \mu_n/x_n\}\big)$$
$$= \{\mu_1/f(x_1), \mu_2/f(x_2), \cdots, \mu_n/f(x_n)\}$$

となる。

2.5 ファジィ測度とファジィ積分

ファジィ測度について述べる前に通常の測度の定義を示す。測度は集合の大きさなどを表す数量的尺度であり、長さ、面積、体積、質量分布、確率分布などの数学的抽象化と考えることができる。

[定義 : 測度]

Ω : 全体集合 (ここでは有限集合のみを考える)

$\forall A \subseteq \Omega, \quad g(A): 2^{\Omega} \to R \quad (0 \le R)$

$g(\phi) = 0$

$g(\Omega) < \infty$
$\forall A, B \subseteq \Omega, \quad A \cap B = \phi$
$g(A \cup B) = g(A) + g(B)$ 加法性

一般に物理量を対象とした測定はこの測度に基づいて行なえば良い。しかし、人間の判断のようなものを対象とする場合、加法性の制約が厳しすぎてうまくいかないことが多い。

例えば、上下2巻からなる本を古本屋に持っていく場面を考える。このとき、一般には次のような状況が存在しうる。

・上巻1冊は1,000円で買ってくれる。

・下巻1冊は800円で買ってくれる。

・上下そろえて持っていくと1,800円ではなく2,000円で買ってくれる。

この例では、上下そろえた場合に加法性が成立していない。

ファジィ測度[50-52]とは、通常の測度における「加法性」の制約を、これより緩い制約である「単調性」に置き換えたものである。

[定義:ファジィ測度]

Ω:全体集合(ここでは有限集合のみを考える)

$\forall A \subseteq \Omega, \quad g(A): 2\Omega \to R \quad (0 \le R)$

$g(\phi) = 0$

$g(\Omega) < \infty$
$\forall A, B \subseteq \Omega$
$A \subseteq B, \quad g(A) \le g(B)$ 単調性

加法性が成り立てば単調性も成立するから、ファジィ測度は通常の測度を含むより一般的な概念である。なお、単調性は次のように2つの場合に分けられる。

優加法性 $g(A \cup B) \ge g(A) + g(B)$

劣加法性 $g(A \cup B) \le g(A) + g(B)$

ファジィ積分[50]とはファジィ測度に基づく積分である。ファジィ積分には菅野積分[51]、ショケ(Choquet)積分[50]等があるが、ここではショケ積分について簡単に述べる。

まず次のような単関数 h を考える(図2.1参照)。

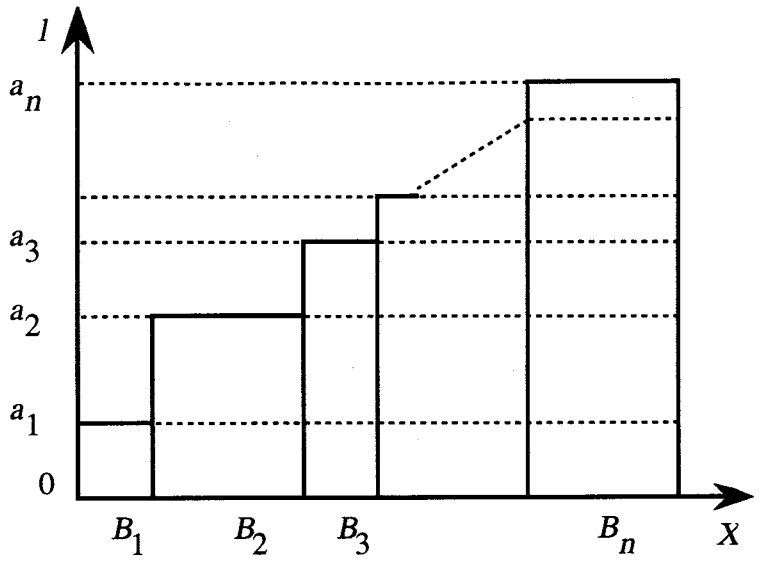

図 2.1 : 単関数 h

$$h: X \to [0,1]$$

$$l_{Ai}(x) = \begin{cases} 1, & x \in A_i \\ 0, & \text{else} \end{cases}$$

$$0 \le a_1 \le a_2 \le \cdots \le a_n \le 1$$

ファジィ測度 g による次の形式をショケ積分と呼ぶ。

$$(c)\int h dg = \sum_{i=1}^{n} (a_i - a_{i-1}) g(A_i)$$

ここで g が通常の測度である場合、この積分はルベーク積分に一致する。

例えば、上・中・下の 3 巻からなる本を古本屋へ持っていくといくらで買ってくれるかについて考えてみよう。

まず、買取価格の定義を行なう。

Ω = {上巻、中巻、下巻}

$g(\{\phi\})$ = 0 円

$g(\{$上巻$\})$ = 800 円

$g(\{$中巻$\})$ = 800 円

$g(\{$下巻$\})$ = 1,000 円

$g(\{$上巻、中巻$\})$ = 1,600 円

$g(\{$上巻、下巻$\})$ = 2,000 円

$g(\{$中間、下巻$\})$ = 2,000 円

$g(\{$上巻、中巻、下巻$\})$ = 3,000 円

これは次に示すように加法性が満たされないので、ファジィ測度となっている。

$g(\{$上巻、中巻、下巻$\})$ > $g(\{$上巻、中巻$\})$ + $g(\{$下巻$\})$

ここで上巻 7 冊、中巻 5 冊、下巻 3 冊をこの古本屋へ持っていくといくらで買ってくれるかについて考える。買取価格の合計は次のショケ積分によって得られる (図 2.2 参照)。

価格合計 = $(c)\int hdg$

= 2 × g({上巻}) + 2 × g({上巻、中巻}) + 2 × g({上巻、中巻、下巻})

= 2 × 800 円 + 2 × 1,600 円 + 2 × 3,000 円

= 13,800 円

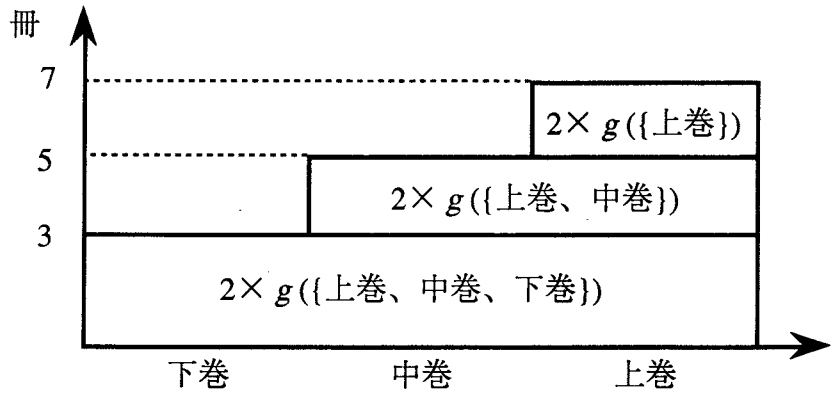

図 2.2 : ショケ積分による古本価格計算の例

2.6 可能性理論

可能性理論とはファジィ集合の提唱者である L. A. Zadeh [9] によって 1978 年に提案され、その後 D.Dubois および H.Prade [7, 8] 等によって整備された理論であり、ファジィ測度の一種である可能性測度と必然性測度によってあいまい性を取扱うものである。

2.6.1 可能性理論による不確実性の取扱い

$p, q, \cdots$ を通常の命題 (真か偽いずれかの真理値をとる) とする。また、 P は次を満たす命題の集合とする。

$p \in P$ ならば $\neg p \notin P$

$p \in P \wedge q \in P$ ならば $p \wedge q \in P$

$0 \in P$

$1 \in P$

ここで、0 は例えば $p \wedge \neg p = 0$ といった恒偽命題を表し、1 は例えば $p \vee \neg p = 1$ といった恒等命題を表す。

また、 P 上の可能性測度 ($Pos(p)$) と必然性測度 ($Nec(p)$) は次のように定義される。

[定義 : P 上の可能性測度 ($Pos(p)$)]

$\forall p \in P, \quad Pos(p) \in [0,1]$

$Pos(0) = 0$

$Pos(1) = 1$

$p \to q$ のとき $Pos(p) \le Pos(q)$

$Pos(p \vee q) = \max(Pos(p), Pos(q))$

[定義 : P 上の必然性測度 ($Nec(p)$)]

$\forall p \in P, \quad Nec(p) \in [0,1]$

$Nec(0) = 0$

$Nec(1) = 1$

$p \to q$ のとき $Nec(p) \le Nec(q)$

$Nec(p \wedge q) = \min(Nec(p), Nec(q))$

可能性理論に基づいて不確実性を扱う場合、 $Pos(p)$ と $Nec(p)$ の組み合わせで命題の確からしさを表現する。例えば、次のようになる。

$Pos(p) = 0, Nec(p) = 0$: p は確実に偽である

$Pos(p) = a < 1, Nec(p) = 0$: p が真である可能性は偽である可能性よりも低い

$Pos(p) = 1, Nec(p) = 0$: p の真偽は全く不明

$Pos(p) = 1, Nec(p) = b < 1$: p が真である可能性は偽である可能性よりも高い

$Pos(p)=1, Nec(p)=1$: p は確実に真である

なお、上の例からもわかるように可能性測度 $Pos(p)$ の値が 1 に近いからといって、確率とは異なり、必ずしも p が真である度合い (可能性) が高いことを意味しないことに注意すべきである。

可能性測度と必然性測度の主な性質を挙げると次の通りである。

$Pos(p \wedge q) \leq \min(Pos(p), Pos(q))$ ただし、等号は p と q が独立な場合

$\max(Pos(p), Pos(\neg p)) = 1$

$Pos(p) = 1 - Nec(\neg p)$

$Pos(\neg p) = 1 - Nec(p)$

$\min(Nec(p), Nec(\neg p)) = 0$

$Nec(p) > 0 \quad \Rightarrow \quad Pos(p) = 1$

なお、上記関係式より、 $Nec(p) = 1 - Pos(\neg p)$ が成立するため、 $Pos(p)$ と $Pos(\neg p)$ の組み合わせで確からしさを表現することもできる。

2.6.2 可能性理論による不完全なデータの取扱い

全体集合 Ω 上における可能性測度と必然性測度について考える。これは本質的には前述の $Pos(p)$ 、 $Nec(p)$ と同じものであるが、ここでは慣用に従って $P(A)$ 、 $N(A)$ で表現する。

[定義 : Ω 上の可能性測度]

Ω : 全体集合 (ここでは有限集合のみを考える)

$\forall A \subseteq \Omega, \quad P(A): 2^{\Omega} \rightarrow [0,1]$

$P(\phi) = 0$

$P(\Omega) = 1$

$\forall A, B \subseteq \Omega, \quad A \subseteq B$ ならば $P(A) \leq P(B)$

$\forall A, B \subseteq \Omega, \quad P(A \cup B) = \max(P(A), P(B))$

可能性測度 $P(A)$ は次の可能性分布関数 $\pi(\omega)$ で定義できる。

$\forall A \subseteq \Omega, \quad P(A) = \sup(\pi(\omega) | \omega \in A)$

ただし、

$\pi(\omega) = P(\{\omega\})$

$\pi(\omega)$: $\Omega \to [0,1]$

$\exists\omega \quad \pi(\omega)=1 \quad (\because \quad P(\Omega)=1)$

可能性分布関数 $\pi(\omega)$ の直観的な意味は次の通りである。例えばある人の身長を観測した結果「だいたい 170cm ぐらい」との情報が得られたとする。この情報は例えば次のような可能性分布で表現できる。

{0.3/167, 0.8/168, 1/169, 1/170, 1/171, 0.7/172, 0.4/173}

ここで「0.3/167」は $\pi(167)=0.3$ を示している。

[定義 : Ω 上の必然性測度]

Ω : 全体集合 (ここでは有限集合のみを考える)

$\forall A \subseteq \Omega, \quad N(A): 2\Omega \to [0,1]$

$N(\phi)=0$

$N(\Omega)=1$

$\forall A, B \subseteq \Omega, \quad A \subseteq B$ ならば $N(A) \le N(B)$

$\forall A, B \subseteq \Omega, \quad N(A \cap B) = \min(N(A), N(B))$

必然性測度は可能性分布関数から次のようにして求められる。

$\forall A, \quad P(A) = 1 - N(\neg A)$

$N(A) = \inf\{1-\pi(w) \quad w \in \neg A\}$

可能性理論に基づいて不完全性を扱う場合、不確実性と同様に、 $P(A)$ 、 $N(A)$ の組み合わせで集合に属する度合を表現する。

2.7 おわりに

この章では、あいまい性を取扱うための理論として、ファジィ論理、T ノルムと T コノルム、ファジィ集合論、ファジィ測度とファジィ積分、可能性理論の概略を説明した。

第3章　ファジィプロダクションシステム

3.1 はじめに

この章では、まずプロダクションシステムの枠組みの中で取扱えるあいまい性について整理した上で、既存のファジィプロダクションシステムではこれらあいまい性を同時には取扱うことができないことを指摘する。次に、ファジィ論理[3-6]、可能性理論[7-9]およびファジィ測度・ファジィ積分[50-52]の各理論を使用して実現したLIFE FEShellファジィプロダクションシステム(FPS)[28, 39, 40]のあいまい性取扱い機能、特にパターンマッチングの詳細について述べる。

3.2 プロダクションシステムに現れるあいまい性

プロダクションシステムとは、一般に「IF 条件部 THEN 結論部」という形式のルールに基づいて推論を行なうものであり、前向き推論タイプと後ろ向き推論タイプに大別できる。

前向き推論タイプは、ワーキングメモリ、プロダクションルールメモリ、およびインタプリタから構成され、次の認知・実行サイクルを繰り返して結論に至るシステムである。

1) ルール条件部の条件を満たすワーキングメモリが存在するルールを見つける。この操作をルールとワーキングメモリとのマッチングと呼ぶ。一般に条件が満たされるルールは複数存在し、これらのルールの集合を競合集合と呼ぶ。
2) 競合集合の中から何らかの方法でひとつのルールを選ぶ。この操作を競合解消と呼ぶ。

3) 選ばれたルールの結論部を実行する。通常はこの実行処理の中でワーキングメモリに対する何らかの操作 (書き込み、消去、更新等) が行なわれる。

一方、後ろ向き推論タイプは、ワーキングメモリ、プロダクションルールメモリ、およびインタプリタから構成される点は前向き推論タイプと同様であるが、次の処理を繰り返して結論に至る推論機構を持つシステムである。

1) 求めたいゴールを満たすデータがあれば、今の質問は yes となる。

2) データが見つからなければ、質問を満たす結論部を持つルールを探す。

3) ルールがあれば、このルールの条件部を新しいゴールとし、再帰的に検証する。

4) ルールがなければ、元の質問は no となる。

一般にプロダクションシステムの枠組みの中で取扱い可能なあいまい性を分類すると以下のようになろう。

3.2.1　ルールの条件部と結論部の関係のあいまい性 (不確実性)

これは完全には正しくないルール、すなわち条件部が満たされたからといって結論部が常に成り立つとは限らないルールで、例えば次のようなルールである。

IF　利益率 > 0.15 and 資本化率 < 0.8　THEN　長期投資は妥当 (0.7)

これはこの条件が満たされたからといって、長期投資が確実に妥当とはいい切れず妥当性に関する確信の度合いが 0.7 であることを示している。なお、ここでこのルールの条件部と結論部はいずれもクリスプな命題である点に注意されたい。

この種のあいまい性を扱う場合、一般にルールの適用によって得られる結論には確信度が付加されることから、ワーキングメモリにも確信度が付加されることになる。すなわち、このタイプのシステムはルールの条件部と結論部の関係の不確実性に加えてデータの信頼性も同時に取扱う必要が生じる。

3.2.2　ルール条件部のあいまい (ファジィ) な条件

これは例えば次のようにルールの条件部にファジィな条件を許すものである。

IF　X は同種の品物に比べて価格が高い　THEN　X は品質が良い

ここで条件部の「同種の品物に比べて価格が高い」はファジイな条件であり、一般にある品物の価格が例えば 1,000 円であることがわかっていてもそれを「高い」、

「高くない」のいずれかに類別することはできず、ルールとデータがマッチしたかしないかの判断を0か1かに割り切って行なうことができない。

通常この種のあいまい性はファジィ論理に基づいて取扱われている。

3.2.3 ルールの条件部を構成する複数の条件間の複雑な関係

ファジィな条件を許すと、1つのルールの条件部で記述される複合条件が、この中に出現する各条件のand (Tノルム)、or (Tコノルム) およびnotによる結合だけでは表現しきれない場合のあることが加藤等の研究[53-56]によって明らかになっている。例えば、条件部の各条件が独立ではなく相互依存性を持つ場合などであり、このような複合条件に対するマッチ度を計算するためにはand (Tノルム)、or (Tコノルム) 以外の結合演算が必要となる。

3.2.4 あいまいなデータの取扱い

例えば「Tomの身長はだいたい170cmである」といったデータ (以下不完全なデータと呼ぶ) を許すものである。この場合、ルールの条件部はクリスプな条件しか許さないものであっても、ルールとワーキングメモリとのマッチング結果は一般に0 (マッチしない) か1 (マッチする) にはならない。例えば、このデータと次のルールとのマッチングを考えてみる。

IF Xの身長 ≧ 170cm THEN ・・・

ここでデータの「Tomの身長はだいたい170cmである」は、Tomは実際には171cmかもしれず、逆に169.5cmかもしれないといったことを表しており、条件「Xの身長 ≧ 170cm」を満たしているかもしれないし、満たしていないかもしれない。すなわち、不完全なデータの存在を許すと、データの不完全性を表現する機構に加え、クリスプな条件を満たしているかいないかがはっきりしないといったあいまい性を取扱う機構が必要になる。

3.2.5 信頼性の高くないデータの取扱い

ワーキングメモリの中に完全には信頼できないデータの存在を許すものである。例えば、「Tomの身長はたぶん170cm以上である」を意味するようなデータがこれに該当する。これは上記の4つのあいまい性を許すと必然的に出現するものである。

3.2.6　ファジィプロダクションシステムにおけるあいまい性

一般にこのようなあいまい性を取扱うプロダクションシステムを実現するに際しては、次の4つの点について考慮する必要がある。

(1)　マッチ度の計算方法

ルールとワーキングメモリ中のデータを比較照合し、現在存在するワーキングメモリ中のデータとマッチするルールを求める操作において、マッチする度合いを計算する必要がある。この方法は、どのようなあいまい性を許容するか、およびそのあいまい性をどのような理論に基づいて表現するかに依存する。

(2)　競合解消

あいまい性を含むルールとデータを扱う場合には、多くのルールが競合することになり、競合集合の中から実行するルールを選択する必要がある。この際、マッチ度の情報をどのように利用するかという問題が生じる。

(3)　結論部へのマッチ度の引き継ぎ

ルールの条件部とワーキングメモリ中のデータのマッチ度をそのルールの結論部においてどのように使用するかを決定する必要がある。前向き推論の場合、一般に結論部には手続きが記述できるため、この手続き内でマッチ度を自由に使用可能とする場合が多い。すなわち、使用方法はユーザまかせとする訳である。

(4)　結論の統合

一般にあいまい性を扱うプロダクションシステムにおいては、複数の異なったルールから類似の結論もしくは矛盾する結論が得られることがある。類似の結論とは、例えば「だいたい170cm位」と「173cm位」のように類似した不完全な結論、あるいは、確信度の異なる同じ結論等を指す。このような場合、これらの結論をどう統合するかが問題となる。いずれにしてもあいまい性を扱うプロダクションシステムでは、この問題にどう対処するかを、採用したあいまい性を扱う理論に基づいて明確にする必要がある。なお、前向き推論のシステムではこの部分もユーザまかせとすることも可能である。

3.2.7 あいまい性を取扱うプロダクションシステムの現状

前節で述べた各あいまい性を取扱う方法・理論としては種々のものが提案されている。例えば、エキスパートシステムとして有名なMYCIN [57, 58] では、確信度(Certainty Factor: CF)によって前節で述べたあいまい性を取扱っている。FMUFL [16] では、ファジィ論理によってルール条件部のファジィ条件を取扱っている。また、Dempster & Shafer 理論 [59] によって前節で述べた5つのあいまい性を取扱っているシステムも存在する。さらに、TAIGER [20] では、可能性理論によってルールの条件部と結論部の関係のあいまい性と信頼性の高くないデータのあいまい性を取扱っている。しかし、5つすべてのあいまい性を同時に取扱おうとすると、いずれの理論でも単独では不可能であり、いくつかの理論を組み合わせて使用せざるを得ない。実際、例えばSYSTEM Z-II [21-23] では、確信度とファジィ集合を組み合わせることによってルール条件部を構成する複数の条件間の複雑な関係以外のあいまい性を取扱っている。このように複数の理論を組み合わせて使用する場合、どのような理論を組み合わせるかが問題となる。

3.3 LIFE FEShell ファジィプロダクションシステムの概要

LIFE FEShell ファジィプロダクションシステム (FPS) では、システムとしての統一性およびファジィ論理の枠組みの中に存在する各種理論を実証するという観点より、ファジィ論理の枠組みの中に存在する理論の組み合わせで、5つすべてのあいまい性を同時に取扱うことにする。

LIFE FEShell FPS は馬野のファジィプロダクションシステム [60, 61] を基に拡張を行なう。LIFE FEShell FPS は、標準的な前向き推論のプロダクションシステムの上で、あいまいな知識を取扱うことができる。

このシステムは知識および事実データ等を定義する部分と推論を実行する部分とデバッガによって構成されている。このシステムは Allegro Common Lisp によって記述され、ファジィ集合の処理に関してはファジィ集合処理システム (FSMS) [45] を使用している。

知識表現に関連する特徴を次に示す。

・ワーキングメモリにデータ自身のあいまい性を表す可能性分布を利用できる。

・ルールの条件部に可能性分布とあいまいな条件を表すファジィ述語を利用できる。

・ルールの結論部に任意の関数が記述できる。

・ルールとワーキングメモリをブロック単位に管理できる。

・ルールの条件部に関数が記述できる。

・競合解消規則はルールブロック単位に設定できる。

・ルールブロック単位に上位ブロックを指定できる。

上述の拡張のため、LIFE FEShell FPS はルールとワーキングメモリに加えて、可能性分布の定義であるターム (term) とファジィ述語の定義であるプレディケート (predicate) を利用する。

3.3.1 あいまい性の取扱い

LIFE FEShell FPS では、ファジィ論理、可能性理論およびファジィ測度・ファジィ積分 [50-52] に基づいてあいまい性を取扱っている。以下各々のあいまい性について具体的にどのような理論で取扱っているかを述べる。

(1) ルールの条件部と結論部間の関係

LIFE FEShell FPS ではこのタイプのあいまい性を TAIGER と同様可能性理論に基づいて取扱う。具体的には、ルール条件部が満たされた時に結論部が成立する可能性測度の値および結論部が成立しない可能性測度の値を各ルールに設定可能としている。すなわち、真理値空間 [0, 1] 上における可能性分布関数 $\pi_r(u)$ が各ルールに対して設定可能で、$\pi_r(1)$ の値はルール条件部が満たされた時に結論部が成立する可能性測度の値を、$\pi_r(0)$ の値は結論部が成立しない可能性測度の値を表す。

(2) ルール条件部内のファジィな条件

ルール条件部内のファジィ条件は、ファジィ論理に基づくファジィ述語、具体的にはファジィ真理値関数によって記述する。

(3) ルール条件部内の各条件間の関係

ルール条件部は、各条件間を「and」、「or」、「not」で結合した形式の複合条件に加え、各条件を別途定義したファジィ測度でショケ積分 (Choquet Integral) する形

式の複合条件も記述できる。これにより、相互に独立ではない条件からなる複合条件を表現することができる。

(4) あいまいなデータ

例えば「Tom の身長はだいたい 170cm である」といった漠然としたデータ (不完全なデータ) は、このデータの定義域上の可能性分布関数によって表現する。これらのデータとルールとのマッチングに加え、ルール実行部におけるこのようなデータの処理が可能である。

(5) 信頼性の高くないデータ

LIFE FEShell FPS ではデータの信頼性も可能性理論に基づいて取扱う。これはあいまいなデータと同様、真理値空間 [0, 1] 上における可能性分布関数 $\pi_r(u)$ を各データに付与することによって表現する。具体的には、 $\pi_r(1)$ の値は当該データが信頼できる可能性測度の値を、 $\pi_r(0)$ の値は信頼できない可能性測度の値を表す。

3.3.2 知識宣言部

知識宣言部では推論規則、事実データ、あいまいな単語等を定義する。

(1) 推論規則の定義

推論規則 (プロダクションルール) はブロック単位に管理することが可能である。ルールブロックの名前はユーザが定義する。推論規則は一般のプロダクションルールと同様に "IF 条件 THEN 結論" という形式で記述する。そして、各ルール単位にルールの確からしさ及び優先順位を記述することができる。

条件部には次のものが記述できる。

- アトム (LISP のアトム)
- 可能性分布
- 変数
- ファジィ述語 (メンバーシップ関数)
- 関数 (LISP によって記述された関数)
- 修飾語 (very 、 not 等)
- フレームに関するコマンド

この他に上記項目のリストが記述できる。関数には2種類あり、関数の返却値をマッチ度とする関数と、関数の返却値とワーキングメモリのデータをマッチングする関数とが存在する。

条件項の結合演算子として and 、 or が記述できる。ただし、 and 、 or が省略された時は and とする。また、 and (T ノルム)、 or (T コノルム) 以外の演算子として平均演算子のひとつであるファジィ測度を用いたファジィ積分ができる。これは、ルールの条件部に評価項目を記述し各項目のマッチ度を積分要素として演算が行なわれる。

ルールの具体的な例を次に示す。

```
(rule-block rb1 (&parent-rb p-block) (&conflict c-rule)
  (r1 (&uncertainty 1.0) (&priority {0.5/0.0 1.0/1.0})
        if (d1 small(=x)) (d2 large(=y)) then (deposit (d3 (*+ =x =y)) =match))))
```

この例では、上位のルールブロックが p-block で、競合解消ルールが c-rule であるルールブロック rb1 にルール r1 が入っていることを示している。

(2) 事実データ (ワーキングメモリ) の定義

ワーキングメモリは事実データや中間仮説を格納するメモリであり、ブロック単位に管理されている。ワーキングメモリはクリスプなデータに加えてあいまいなデータも記述できる。あいまいなデータを格納する際には、直接ファジィ集合を記述するか、あるいはファジィ集合によって定義したあいまいな言葉 (term) を用いる。

ワーキングメモリの具体的な例を次に示す。

```
(working-memory taro {(height about170) {0.5/0.0 1.0/1.0}/(weight 70)})
```

この例では、ワーキングメモリブロック taro の中に、 height と weight のデータが入っていることを示している。

(3) ファジィ述語 (メンバーシップ関数) の定義

ファジィ述語はいわゆるメンバーシップ関数であり、ルールの条件部にあいまいな述語として記述できる。ルール条件部で使用する際にはここでファジィ述語を定義する必要がある。また、ファジィ述語は離散表現あるいは連続表現の両方の表現を許している。

ファジィ述語の具体的な例を次に示す。

```
(predicate  (tall(u) (getgrade {0.25/165, 0.5/170, 0.75/175, 1.0/180, 1.0/185, 1.0/190} u))
            (small(u) (s-func (u 0.5 0.75 1))))
```

この例では、ファジィ述語として、 tall(u) を離散表現で、 small(u) を連続表現で定義している。

(4) あいまいな言葉 (可能性分布) の定義

あいまいな言葉をタームとして可能性分布によって定義する。なお、その時に可能性分布は離散表現によって定義する必要がある。

可能性分布の具体的な例を次に示す。

```
(term (about170 {0.5/165, 1.0/170, 0.5/175}))
```

この例では、タームとして、 about170 を定義している。これは例えば、 165 に対するグレードが 0.5 といった形で表現されている。

(5) ファジィ測度の定義

ファジィ積分によって複数項目の総合評価が可能である。評価項目の集合の族をファジィ測度として表現する。この時のファジィ測度は評価項目の重み付けを表現しているのでユーザが決定する必要がある。また、ファジィ測度は評価する集合の全ての族について定義する必要がある。なお、記述されていないものは測度 0 として解釈する。

ファジィ測度の具体的な例を次に示す。

```
(measure (measure1 {0.5/(a) 0.4/(b) 0.6/(c) 0.6/(a c) 0.9/(b c) 1.0/(a b c)}))
```

この例では、ファジィ測度として measure1 を定義している。これは例えば、 (a) に対する値が 0.5 、 (a c) に対する値が 0.6 といった形で表現されている。

3.3.3 推論部

推論部は大きく分けて推論機能、推論環境設定機能、内部状態の保存、読込機能の 3 つを持っている。

(1) 推論機能

これは知識宣言部によって定義された推論規則や事実データ等をもとに推論を進める機能であり、次の 2 つの機能がある。

(i)　モード選択

モードとしてデバッグモードと実行モードが用意されている。この指定をせずに推論を開始するとデフォルトとして実行モードとなる。

(ii)　推論開始

推論を開始する。推論を開始するコマンド(start)のあとに推論の環境設定をするコマンドを付加することも可能である。推論結果の返却値の指定ができる。

(2)　推論環境設定機能

これは推論を進める際の条件や関数を指定する機能である。また、これらの環境設定は推論中に変更することも可能である。

(i)　パターンマッチングの際のマッチ度を算出する関数の指定

次の3つに関して関数を指定できる。デフォルトはそれぞれminとする。

・ひとつの条件とデータのマッチ度　(level 1)

・データの確からしさを含めたマッチ度　(level 2)

・全ての条件に関するマッチ度　(level 3)

(ii)　しきい値の設定

上記3つのレベルにそれぞれしきい値を設定できる。マッチング結果がしきい値以下の場合は競合集合に加えない。

(iii)　1度の認知 - 実行サイクルで発火するルール数

発火するルール数の指定方法は2つある。

・発火するルール数を指定する方法

・マッチ度がある一定以上のルール全てを発火させる方法

(iv)　バックトラックの指定

マッチするルールがなくなったときにバックトラックするかしないかの指定と、バックトラックする際に指定場所に戻るか、あるいはひとつ上に戻るかの指定を行なう。

(v)　推論開始時のルールブロックおよびワーキングメモリブロックの指定

推論開始時にカレントとなるルールブロックおよびワーキングメモリブロックを指定する。デフォルトは最初に定義されたブロックである。

(vi) マッチング方法の選択

次のマッチング方法の選択が可能である。

・全てのルール、全てのデータを対象に競合集合を作成する方法

・全てのルールと最初にマッチするデータを競合集合とする方法

(vii) 高速推論スイッチ

高速推論アルゴリズムを使用するかどうかの選択をする。使用する場合はどのアルゴリズムを使用するかを選択する。

・一度マッチしたルールとデータは履歴データとして残す。この履歴データを参照して次の認知 - 実行サイクルの際に同一データとルールのマッチングを省略することによって高速化を図る。

・同一ルールブロック内に共通する条件項に着目したリートアルゴリズム風のルール木を作成する。これによって共通する条件項の重複マッチングを防止することが可能である。ただし、条件項が関数の場合にはその対象としない。

(3) 内部状態の保存、読込

推論中の内部状態の保存、読込を行なう機能である。保存、読込の対象となる項目を次に示す。

・サイクル数

・カレントルールブロック名とその時のルール (内部表現)

・カレントワーキングメモリブロック名とその内容

・ファジィ述語の内容

・可能性分布の内容

・ファジィ測度の内容

・推論環境設定 (しきい値、競合解消規則等)

・バックトラック用スタック

・デバッグ環境

3.3.4 デバッガ

デバッガは推論経過の表示や推論のブレイク等を指定できるデバッグ機能であり、推論のトレース、ブレイクポイントの設定、内部状態の表示、指定サイクル後のブレイク、ヒストリ機能、推論サイクルを戻すといった機能を持っている。

(1) 推論のトレース

次の推論経過について表示する。

・カレントワーキングメモリの内容

・競合集合 (マッチしたデータおよびマッチ度)

・競合解消によって発火したルール

・バックトラックの発生

・バックトラック用スタックに積まれているルールおよびデータ

・現在の推論サイクル数

(2) ブレイクポイントの設定

推論を一時中断する場所を指定する。指定できるブレークポイントの場所を図3.1に示す。なお、ブレイクポイントを指定すると毎サイクルブレイクする。また、推論の再開も可能である。

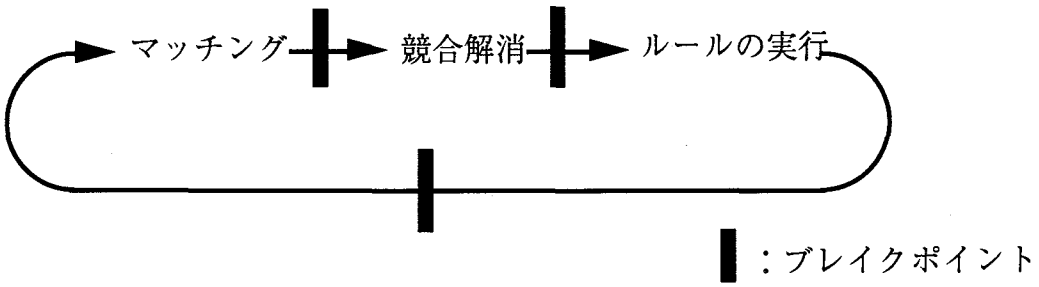

図3.1：ブレイクポイント

(3) 内部状態の表示

推論終了後やブレイク中にワーキングメモリやファジィ述語の内容を表示する機能である。次のものを表示できる。

・ワーキングメモリ

・バックトラック用スタックの内容

・現在のサイクル

・ファジィ述語の内容

・可能性分布の内容

(4) 指定サイクル後のブレイク

推論を開始してから何サイクル目で中断したいかを指定する。中断後、再開も可能である。ブレイクポイントはルールの実行部を実行した後である。ルール指定でこのルールが発火したらブレイクしたいという場合にはルールの実行部にブレイクコマンド(ルール実行部のコマンド)を埋め込めばよい。

(5) ヒストリ機能

推論結果の履歴を保存し、保存した履歴を表示する。保存内容を次に示す。

・サイクル数

・カレントルールブロック名とその時のルール(内部表現)

・カレントワーキングメモリブロック名とその内容

・タイムタグの内容

・ファジィ述語の内容

・可能性分布の内容

・ファジィ測度の内容

・推論環境設定(しきい値、競合解消規則等)

・バックトラック用スタック

(6) 推論サイクルを戻す

指定サイクル分、推論サイクルを戻す機能である。この時に、ヒストリ機能による推論結果の履歴は変更される。サイクルの指定方法は何サイクル目に戻るかの指定と何サイクル前に戻るかの2つがある。

3.4 LIFE FEShell ファジィプロダクションシステムの詳細

LIFE FEShell FPS において最も重要なパターンマッチングについて述べる。

3.4.1　ルールとワーキングメモリ

まず、LIFE FEShell FPS におけるルールとワーキングメモリについて述べる。

(1)　ルール

ルールは次の 2 種類の形式が記述できる。

(i)　形式 1

$(\pi_r(u)$　IF　$A_1 * A_2 * \cdots * A_n$　THEN　$C_1 C_2 \cdots C_m)$

$\pi_r(u)$: 真理値空間 [0, 1] 上における可能性分布であり、ルール条件部と結論部の関係の不確かさを表す

A_i :　条件 (パターンもしくはファジィ述語の列)

パターン : クリスプな値もしくは可能性分布で表現された値

* :　and もしくは or で、and は省略可能

C_i :　システム提供コマンドもしくはユーザ定義の関数

(ii)　形式 2

$(\pi_r(u)$　IF　$(FN)(A_1\ A_2\ \cdots\ A_n)$　THEN　$C_1 C_2 \cdots C_m)$

FN : 別途定義されたファジィ測度 g_{FN} の名前

ただし、

$\Omega = \{A_1, A_2, \cdots, A_n\}$

$g_{FN}(\Omega) = 1$

(2)　ワーキングメモリ

ワーキングメモリは一般に次の形式が記述できる。

$\{\pi_w(u)/(e_1\ e_2\ \cdots\ e_k), \cdots\}$

$\pi_w(u)$: 真理値空間 [0, 1] 上における可能性分布

e_i :　クリスプな値もしくは可能性分布で表現された漠然とした値

3.4.2　パターンマッチング

LIFE FEShell FPS ではルールとワーキングメモリのマッチングは 4 つの段階をへて実現される。以下ではこれをレベル 1 からレベル 4 とする (図 3.2 参照)。

いずれの段階のマッチング結果も真理値を表す空間 [0, 1] 上の可能性分布であり、真理値空間 [0, 1] で fuzziness の度合いを、またこの上の可能性分布で uncertainty の度合いと ambiguity の度合いをそれぞれ表現する。なお、例えば FLOPS [18, 25] のようにマッチ度を 1 つの数値で表現するシステムではこのように複数種類のあいまい性を同時に取扱うことはできない。

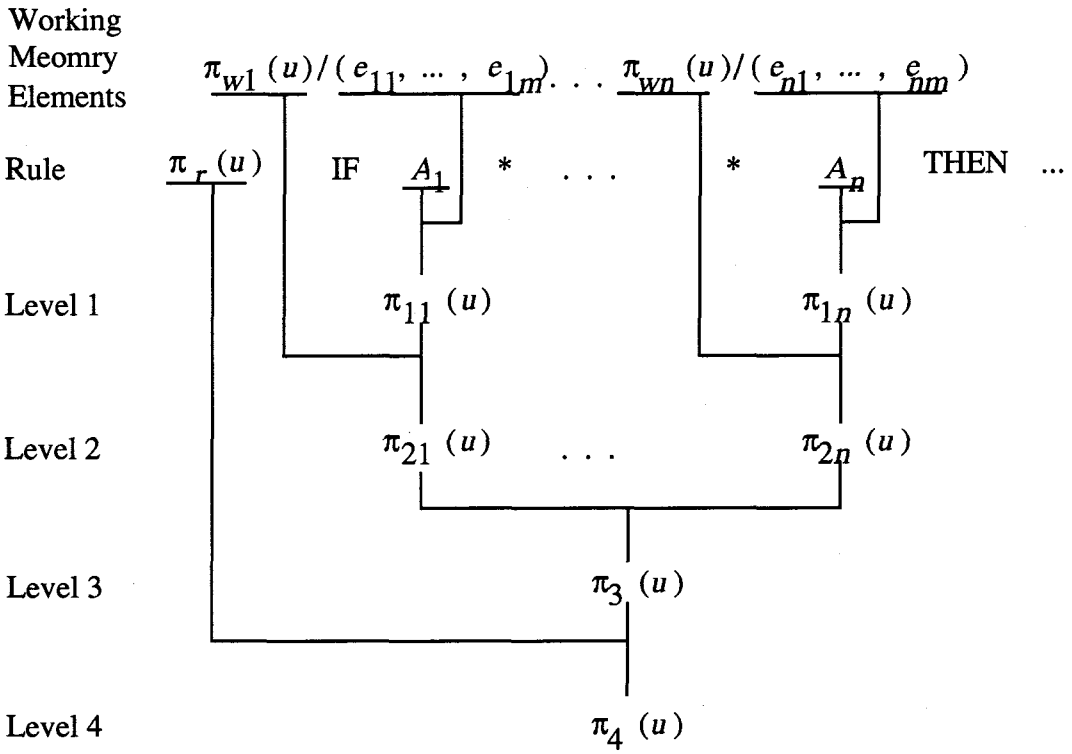

図 3.2 : マッチングのレベル

(1) レベル 1 のマッチング

これはルール中の A_i とワーキングメモリ中の e_i のマッチングである。このマッチングは (i) パターンとデータのマッチング、(ii) ファジィ述語とデータのマッチングの 2 種類に分けられる。さらに、(iii) 条件全体のマッチ度を計算する必要がある。

(i) パターンとデータのマッチング

これはパターンとして表された値とワーキングメモリ (データ) が一致しているか否かを判断する処理である。一般にパターン、データ共に可能性分布が記述可能なため、マッチング結果は真理値をあらわす空間 [0, 1] 上における可能性分布となる。なお、パターンあるいはデータのいずれか一方もしくは双方がクリスプな値である場合、定義域上のその値の可能性測度の値が 1 、他の値に対して可能性測度の値が

0と考えれば、パターン、データ共に可能性分布と考えることができるので、以下では可能性分布で表現された値同士のマッチングについてのみ述べる。

例えば、

$U=\{1,2,3,4,5,6,7,8,9,10\}$

を定義域とするパターン a とデータ b が次のような可能性分布関数で与えられていたとする。

$\Pi_a(u)=\{1/1, 0.8/2, 0.4/3\}$

$\Pi_b(u)=\{0.4/1, 1/2, 0.4/3\}$

この表現は、例えば b の値は1、2、3のいずれかであり、1である可能性測度の値が0.4、2で1、3で0.4であることを意味している。すなわち、b の値は1、2、3のいずれかでありそれ以外ではありえず、なかでも2である可能性がもっとも高い。そして、このパターンとデータのマッチングは、a と b の実際の値が一致している可能性を評価するものである。この場合、a、b 共に1、2、3のいずれかであり、実際に一致している可能性があるが、逆に一致していない可能性もある。これらの可能性の度合いを $\pi_a(u)$ および $\pi_b(u)$ から可能性理論に基づいて計算するのがマッチング処理である。これは、マッチングという観点からみると可能性分布で表現されたデータ同士のマッチングであるが、条件評価の観点から見ると暗黙に定義された述語 $Equal(x,y)$ によって未知のデータ a、b を評価していることになる。ただし、$Equal(x,y)$ は次の真理値関数 $\mu_{Equal}(x,y)$ によって定義される述語を表わす。

$$\mu_{Equal}(x,y)=\begin{cases}1, & x=y\\ 0, & x\neq y\end{cases}$$

このマッチングにおいては一般に結果はYesもしくはNoとはならないため、$Pos(a=b)$ と $Nec(a=b)$ の組み合わせで表現する方法と、意味的には全く等価である $Pos(a=b)$ と $Pos(a\neq b)$ の組み合わせで表現する方法がある。LIFE FEShell FPSでは後者の表記を採用し、具体的には次のように真理値空間 [0, 1] 上の可能性分布によって表現する。

$\{Pos(a\neq b)/0, Pos(a=b)/1\}=\{\pi(0)/0, \pi(1)/1\}$

具体的なマッチング結果の求め方は次の通りである。

$$\begin{aligned}\pi(1) &= Pos(a=b)\\ &= Pos\big((a=x_1 \wedge b=x_1)\vee(a=x_2 \wedge b=x_2)\vee\cdots\vee(a=x_n \wedge b=x_n)\big)\\ &= \max_i\big(Pos(a=x_i \wedge b=x_i)\big)\\ &= \max_i\big(\min\big(Pos(a=x_i), Pos(b=x_i)\big)\big)\\ &= \max_i\big(\min\big(\pi_a(x_i), \pi_b(x_i)\big)\big)\end{aligned}$$

ただし、 $X=\{x_1, x_2, \cdots, x_n\}$ は離散的であると考えている。

なお、可能性理論において一般には、

$$Pos(p\wedge q)=\min\big(Pos(p), Pos(q)\big)$$

は成立しないが、 p と q が独立 (non - interactive) である場合には成立する [24]。ここでは独立性を仮定している。

$$\begin{aligned}\pi(0) &= Pos(a\neq b)\\ &= Pos\big((a=x_1 \wedge b\neq x_1)\vee(a=x_2 \wedge b\neq x_2)\vee\cdots\vee(a=x_n \wedge b\neq x_n)\big)\\ &= \max_i\big(Pos(a=x_i \wedge b\neq x_i)\big)\\ &= \max_i\big(\min\big(Pos(a=x_i), Pos(b\neq x_i)\big)\big)\\ &= \max_i\big(\min\big(\pi_a(x_i), 1-Nec(b=x_i)\big)\big)\end{aligned}$$

ここで、 $Nec(b=x_i)$ は次のように求められる。

$$Nec(b=x_i)=\begin{cases}0, & \pi_b(x_j)=\pi_b(x_k)=1 \quad (j\neq k)\\ 0, & \pi_b(x_j)=1,\ \pi_b(x_k)<1 \quad (j\neq k)\\ 1-\pi_b(x_j), & \pi_b(x_i)=1,\ \pi_b(x_k)<1 \quad (j\neq k)\\ & \pi_b(x_j)=\max\{\pi_b(x_1), \pi_b(x_2), \cdots \pi_b(x_{i-1}), \pi_b(x_{i+1}), \cdots \pi_b(x_n)\}\end{cases}$$

なお、複数の x_i に対して $\pi_a(x_i)=1$ が成立するかもしくは複数の x_j に対して $\pi_b(x_i)=1$ が成立する場合、

$$Pos(a\neq b)=Pos(b\neq a)=1$$

が成立するが、ある x_i に対して $\pi_a(x_i)=\pi_b(x_i)=1$ が成立し、その他の x_j に対しては $\pi_a(x_j)<1$, $\pi_b(x_j)<1$ となる場合、一般には

$$Pos(a\neq b)\neq Pos(b\neq a)<1$$

となる。これは可能性分布を粗い離散データとして表現することに起因して発生する一種の誤差であり、注意を要する。

(ii)　ファジィ述語とデータのマッチング

これは真理値関数の形で定義されたファジィ述語とワーキングメモリ中のデータとのマッチングである。このマッチングは拡張原理に基づいて計算され、一般にマッチングの結果は真理値をあらわす区間 [0, 1] 上における可能性分布となる。具体的には、ファジィ述語 F の真理値関数を $\mu_F(x)$ とし、ワーキングメモリ中のデータが $\mu_d(x)$ なる可能性分布で与えられているとすると、マッチ度は拡張原理に基づいて区間 [0, 1] 上における可能性分布 $\pi_m(u)$ によって与えられる。

$$\pi_m(u)=\begin{cases}\sup x, & \{\pi_d(x)\quad \mu_F(x)=u\}\\ 0, & \mu_{F-1}(x)=0\end{cases}$$

また、ルール中でファジィ述語 F に「not」が付加されている場合のマッチ度 $\pi_{m'}(u)$ は F に対するマッチ度 $\pi_m(u)$ をもとにして次のように計算される。

$$\mu_m(u)=\mu_{m'}(1-u)$$

(iii)　条件全体のマッチ度

条件 A_i が $(a_{i1}, a_{i2}, \cdots, a_{in})$、ワーキングメモリが $\pi_w(u)/(e_1, e_2, \cdots, e_n)$ であるとき、a_{i1} と e_1、a_{i2} と e_2 等のマッチ度は上記 (i) および (ii) で得られる。これらを $\pi_{i1}(u)\,\pi_{i2}(u)\cdots\pi_{in}(u)$ とする。これらが得られるとこれらを統合して $(a_{i1}, a_{i2}, \cdots, a_{in})$ と $\pi_w(u)/(e_1, e_2, \cdots, e_n)$ のマッチ度、すなわち、レベル 1 のマッチ度 $\pi_1(u)$ を求める必要がある。このマッチ度は次の式で求められる。

$$\pi_1(u)=\sup_{u=T(u_1,u_2,\cdots,u_n)}\min\left(\pi_{i1}(u_1), \pi_{i2}(u_2), \cdots, \pi_{in}(u_n)\right)$$

(2)　レベル 2 のマッチング

これはレベル 1 のマッチング結果 $\pi_1(u)$ とワーキングメモリ $\pi_w(u)/(e_1, e_2, \cdots, e_n)$ における $\pi_w(u)$ との統合である。これは可能性理論と拡張原理に基づいて、次のように計算される。

$$\pi_2(u)=\sup_{u=\min(x,y)}\min\left(\pi_w(x), \pi_1(y)\right)$$

(3)　レベル 3 のマッチング

レベル 3 のマッチングは、ルールが 2 形式あるため、それぞれ異なった処理を行なう。

(i) 形式１のルールの場合

ルール条件部の各条件 A_i に対するレベル２のマッチング結果を条件中の「*」すなわち「and」、「or」の定義に従って統合する処理である。ここで「and」としては任意のＴノルム演算が、「or」としては任意のＴコノルム演算が使用可能である。

なお、可能性分布に対するＴノルム演算およびＴコノルム演算は拡張原理に基づいて計算する。例えば、 $\pi(u)$ と $\pi'(u)$ に対するＴノルム演算結果を $\pi''(u)$ とすると $\pi''(u)$ は次のように計算される。

$$\pi''(u) = \sup_{u=T(x,y)} \min\left(\pi(x), \pi'(y)\right)$$

(ii) 形式２のルールの場合

ルールの条件部の各条件 A_i に対するレベル２のマッチング結果の列を被積分関数としてファジィ測度 g_F によってファジィ積分(ショケ積分)する。ここでファジィ測度 g_F としては次の条件を満たす任意のものが使用可能である。

$$g_F(\phi) = 0$$
$$g_F(\Omega) = 1$$

ただし

$$\Omega = \{A_1, A_2, \cdots, A_n\}$$

この積分においては被積分関数が可能性分布の列となっていることが特徴的である。すなわち、被積分関数 h は次のように定義される関数である。

$$h : \Omega \rightarrow [0,1]^U$$
$$h(A_i) : U \rightarrow [0,1]$$
$$h(A_i) = \pi_{2i}(u)$$

ただし [0, 1] は可能性測度の値の空間であり、 U は真理値を表す区間 [0, 1] である。

この積分結果を $\pi_3(u)$ とすると、 $\pi_3(u)$ は次のように定義される。

$$\pi_3(u) = (c)\int h dg$$
$$= \sup_{u=(c)\int(u_1,u_2,\cdots,u_n)dg} \min\left(\pi_{21}(u_1), \pi_{22}(u_2), \cdots, \pi_{2n}(u_n)\right)$$

なお、ショケ積分による評価は $g_F(\Omega)=1$ なるファジィ測度を対象としていることから、レベル２のマッチング結果を可能性分布ではなく $U=[0,1]$ 上の１点の値(こ

れを $m_1, m_2, \cdots, m_n$ とする) とした場合、ショケ積分による評価結果を D とすると一般に次の式で表される関係が成立する。

$$T\left(\cdots T\left(T(m_1, m_2), m_3\right), \cdots m_n\right) \le D \le S\left(\cdots S\left(S(m_1, m_2), m_3\right), \cdots m_n\right)$$

ここで、T は任意のTノルム演算を、S は任意のTコノルム演算を示す。

このことより、ショケ積分による評価は、被積分関数に対してある種の平均演算をほどこしたものに等しくなる。このシステムにおいて平均演算を採用せずショケ積分を採用した理由は、ファジィ測度 g_F を与える方が平均演算子の同定よりユーザにとって容易だと判断したことによる。

(4)　レベル4のマッチング

ルールの条件部とワーキングメモリのマッチングはレベル3までで完了である。レベル4のマッチングは、レベル3のマッチング結果 $\pi_3(u)$ とルールに付与されたルール条件部と実行部の関係に関する確からしさ $\pi_r(u)$ を統合する処理である。この処理は次の式に基づいて実行される。

$$\pi_4(u) = \sup_{u=\min(x,y)} \min\left(\pi_3(x), \pi_r(y)\right)$$

ここで $\pi_4(u)$ はレベル4のマッチング結果を表し、この値がルールの実行部に引き渡される。

3.4.3　インプリケーション

一般にあいまい性を扱うプロダクションシステムにおいては、ルール条件部とワーキングメモリのマッチ度を当該ルールの結論部でどのように使用するかを決定する必要がある。通常前向き推論のシステムの場合、後ろ向き推論とは違って、原則として結論部には手続きが記述できるため、この手続き内でマッチ度を自由に使用可能とする、すなわち使用方法はユーザまかせとする場合が多い。さらに、あいまい性を扱うプロダクションシステムにおいては、一般に複数の異なったルールから類似の結論もしくは矛盾する結論が得られることがある。このような場合、これら結論をどう統合するかが問題となる。この問題に対しても前向き推論のシステムではルール実行部の手続きまかせとすることが可能である。LIFE FEShell FPS では、

レベル4のマッチング結果 $\pi_4(u)$ をルール実行部の手続き中で使用可能とすることによって、この値の使用方法はユーザにまかせている。

3.5 具体例

以上に述べたマッチング処理の具体例を以下に示す。

3.5.1 ワーキングメモリとルール

次のワーキングメモリとルールを考える。

(1) ワーキングメモリ

$$\left\{\pi_{w1}(u)/(d_1\ \pi_{d1}(x)),\ \pi_{w2}(u)/(d_2\ \pi_{d2}(y))\right\}$$

ただし、

$$\pi_{w1}(u)=\{1/1\}$$
$$\pi_{w2}(u)=\{0.2/0,1/1\}$$
$$\pi_{d1}(x)=\{0.5/170,1/171,0.6/172,0.2/173\}$$
$$\pi_{d2}(y)=\{0.6/59,1/60,0.8/61,0.4/62\}$$

(2) ルール

$$\pi_r(u)\quad \text{IF}\quad (d_1\quad Tall(t))\ and\ (d_2\quad v\le 60)\quad \text{THEN}\quad \cdots$$

ただし、

$$\pi_r(u)=\{0.2/0,1/1\}$$
$$\mu_{Tall}(x)=\{0.2/168,0.4/169,0.6/170,0.7/171,0.8/172,0.9/173,1/174,1/175,\cdots\}$$

3.5.2 マッチング

以下ではこれらのマッチング処理過程をトレースする。

(1) レベル1マッチング

$(d_1\ \pi_{d1}(x))$ と $(d_1\ Tall(t))$ のマッチング、および $(d_2\ \pi_{d2}(y))$ と $(d_2\ v\le 60)$ のマッチングがレベル1マッチングとなる。

(i) $(d_1\ \pi_{d1}(x))$ と $(d_1\ Tall(t))$ のマッチング

・ d_1 と d_1 のマッチング

これらは一致していることからマッチ度 $\pi_{111}(u)$ は次のようになる。

$$\pi_{111}(u)=\{1/1\}$$

・ $\pi_{d1}(x)$ と $Tall(t)$ のマッチング

可能性分布で表現されたデータとファジィ述語のマッチングである。この場合のマッチ度 $\pi_{112}(u)$ は次の式で得られる。

$$\pi_{112}(u)=\begin{cases}\sup x, & \{\pi_{d1}(x)\,\mu_{Tall}(x)=u\}\\ 0, & \mu_{Tall}(x)=0\end{cases}$$
$$=\{0.5/0.6,1/0.7,0.6/0.8,0.2/0.9\}$$

・ $\pi_{111}(u)$ と $\pi_{112}(u)$ の統合

これは $(d_1\ \pi_{d1}(x))$ と $(d_1\ Tall(t))$ のマッチ度 $\pi_{11}(u)$ を求める操作に対応する。

$$\pi_{11}(u)=\sup_{u=T(u_1,u_2)}\min(\pi_{111}(u_1),\pi_{112}(u_2))$$
$$=\{0.5/0.6,1/0.7,0.6/0.8,0.2/0.9\}$$

ここでTノルムとしてはminを使用している。

(ii)　$(d_2\ \pi_{d2}(y))$ と $(d_2\ v\le 60)$ のマッチング

上記と同様にして求めるとマッチ度 $\pi_{12}(u)$ は次のようになる。

$$\pi_{12}(u)=\{0.8/0,1/1\}$$

(2)　レベル2マッチング

ワーキングメモリ $\pi_{w1}(u)/(d_1\ \pi_{d1}(x))$ における $\pi_{w1}(u)$ と上記で得られた $\pi_{11}(u)$ の統合、および $\pi_{w2}(u)/(d_2\ \pi_{d2}(y))$ における $\pi_{w2}(u)$ と $\pi_{12}(u)$ の統合である。

(i)　$\pi_{w1}(u)$ と $\pi_{11}(u)$ の統合

統合結果を $\pi_{21}(u)$ とするとこれは次のように計算される。

$$\pi_{21}(u)=\sup_{u=\min(u_1,u_2)}\min(\pi_{w1}(u_1),\pi_{11}(u_2))$$
$$=\{0.5/0.6,1/0.7,0.6/0.8,0.2/0.9\}$$

(ii)　$\pi_{w2}(u)$ と $\pi_{12}(u)$ の統合

統合結果を $\pi_{22}(u)$ とするとこれは次のように計算される。

$$\pi_{22}(u)=\sup_{u=\min(u_1,u_2)}\min(\pi_{w2}(u_1),\pi_{12}(u_2))$$
$$=\{0.8/0,1/1\}$$

(3) レベル3マッチング

レベル3マッチングはルール条件部全体のマッチ度を求める操作である。この例の場合ルール条件部である「$(d_1 \quad Tall(t))\ and\ (d_2 \quad v \le 60)$」と対応するワーキングメモリとのマッチングである。この操作は、ルール中の個々の条件とワーキングメモリのマッチ度はすでにレベル2マッチングで次のように得られているため、これらマッチ度を「and」の定義に従って統合すればよい。

$$\pi_{21}(u) = \{0.5/0.6, 1/0.7, 0.6/0.8, 0.2/0.9\}$$
$$\pi_{22}(u) = \{0/0.8, 1/1\}$$
$$\pi_3(u) = \sup_{u=T(u_1,u_2)} \min(\pi_{21}(u_1), \pi_{22}(u_2))$$
$$= \{0.8/0, 0.5/0.6, 1/0.7, 0.6/0.8, 0.2/0.9\}$$

ここではTノルムとしてminを使用している。

(4) レベル4マッチング

レベル4マッチングは上記 $\pi_3(u)$ とルールに付与されている $\pi_r(u)$ の統合である。

$$\pi_r(u) = \{0.2/0, 1/1\}$$
$$\pi_3(u) = \{0.8/0, 0.5/0.6, 1/0.7, 0.6/0.8, 0.2/0.9\}$$
$$\pi_4(u) = \sup_{u=\min(u_1,u_2)} \min(\pi_3(u_1), \pi_r(u_2))$$
$$= \{0.8/0, 0.5/0.6, 1/0.7, 0.6/0.8, 0.2/0.9\}$$

以上により最終的なマッチング結果 $\pi_4(u)$ が得られたことになる。得られた値はインプリケーションにより実行部に引き渡され、実行結果に影響を与える。

3.6 おわりに

この章では、LIFE FEShell FPSで採用しているあいまい性取扱い機能について述べた。このシステムの特徴は次の通りである。

1) ルール条件部内におけるファジィな条件、ルールの条件部と結論部間の関係のあいまい性およびデータの中に不確実性と不完全性が同時に出現する状況が取扱える。
2) ルール条件部内の各条件間の関係として、and、orのみならず、ファジィ測度で定義される複雑な関係も取扱うことができる。

これらの特徴により、LIFE FEShell FPSは広範囲な悪構造問題に適用可能である。

第4章　ファジィフレームシステム

4.1 はじめに

この章では、まずフレームに関する基本的な考え方を示し、フレームシステムで扱うことのできるあいまい性について分析した上で、LIFE FEShell ファジィフレームシステム (FFS) [29, 34, 35] のあいまい性の取扱いについて述べる。LIFE FEShell FFS では、値や関係のあいまい性の表現に加えて、あいまい性の階層的定義ができる。

4.2 フレームによる知識表現

この節では、フレームに関する基本的な考え方と、フレームシステムの機能について述べる。

4.2.1　基本的な考え方

知識表現において、表現の単位が小さければ、複雑な問題を扱うときには足かせとなり、人間が日常行なっているような推論や記憶の効率のよさを、うまく説明できない。

例えば、車を運転することやレストランで食事する場合を想定してみよう。われわれは複雑な推論や計算をしながら行動しているとは思われない。車の運転やレストランでの食事についての典型的な状況の知識を用いて、現実の状況が予測又は期待される範囲内で推移するならば、特別な推論を必要としないで、自然かつ迅速に対応することができる。このような対応が出来るのは行動に必要な知識が運転や食

事などの事象を中心に効率的に組織化されているためと考えられる。なにか予想されないような事態が起こった場合に限り、特別の推論が駆動される。

また、プロダクションルールのように、知識を文章で表現する形式は理解しやすく作りやすいが、反対に知識の全体像がとらえにくい。人間の記憶モデルは、知識の寄せ集めといった単純なものではなく知識と知識が互いに関連づけられて二次元あるいは三次元的に記憶されていると考えられる。したがって、我々人間にとってより自然な知識は、もっと大きなまとまりであって、しかももっと構造化されている方がよい。

また、推論についても、全体を通して単一の方法を用いるよりも、問題解決の局面に応じてそれぞれ異なったやり方を用いる方がより自然である。

そのためには、事実に関する知識と手続きに関する知識がもっと密接に結びつくようにし、全体がさらに一つの体系として構造化される必要がある。フレームはこのような考え方に基づく知識を表現するための枠組である。

フレームによる知識表現は、人間の記憶及び認知の過程をモデル化するための枠組みとして1975年にMITのM. Minsky [62] によって提案された。フレーム表現は知識の構造的表現の一つであり、典型的な状況や事象などの概念的記述とそれらの間の階層的関係の記述を利用して、具体的な状況や事象に対する知識表現と問題解決を効率良く行なうことを目的としている。

フレームシステムは以下のような特徴を持っている。

- 人間が扱っているような概念の抽象度に基づく階層的な知識の表現や、物体の構造に関する構造的な知識を人間に理解しやすい自然な形で表現できる。
- 人間の行なっている予期および仮定を、デフォルトという仮の値を用い、欠落した情報を補って、類推や理解などの推論を進めることができる。
- 知識の階層構造やデフォルトを用いて推論することによって、知識ベースの記述量を小さくすることができる。
- 処理機構がコンパクトになり、フレームシステム全体が小さくできる。

各フレームには、事実や事象を表す宣言型知識だけでなく、情報を処理する手続き型知識を記述することができる。これらの点によって、フレーム型システムは汎用で強力なツールとなっている。

4.2.2 階層構造

構造化された知識の表現として、階層構造が利用される。以下では、階層構造における関係を抽象 - 具体関係、全体 - 部分関係、包含関係の3つに分類して、それぞれの特徴を述べる。

(1) 抽象 - 具体関係

抽象 - 具体関係では、上位にはより抽象的対象を、下位にはより具体的対象を定義し、下位の対象は上位の対象が持つ属性の一部を継承することができる。この関係は、例えば「canary is a bird」、あるいは「canary is a kind of bird」と書くことができるので、ISA 関係あるいは A-KIND-OF 関係とも呼ばれる。

例えば、動物が「死ぬ」という属性は、鳥にもその下位のカナリアにも共通である。そこで、動物という概念を鳥の上位に定義し、そこに「死ぬ」という属性を定義しておけば、継承によって鳥とカナリアが死ぬことをそれぞれに記述する必要がなくなる。

フレームは、このような記憶構造をとることによって、知識の体系化、記憶の経済性を達成している。さらに、特定の対象の知識（例えば鶏）が定義されていなくても、それが鳥の一種であることがわかれば、鳥および動物の知識を使ってある程度の推論が可能となるなど、知識を扱う上で柔軟性が高い。

(2) 全体 - 部分関係

全体 - 部分関係は、ある構造をもつ対象に関する知識表現を行なう場合、下位の対象が上位の対象の一部分の詳細情報を表現しているという関係を表している。これは PART-OF 関係と呼ばれるものの一部である。

全体 - 部分関係では、上位の対象から下位の対象に継承することができる属性に制限がある。

例えば、「机」に関する知識を表現する場合、「脚」や「引出し」に対して、色や材質については継承の対象と考えられるが、大きさや重さなどはそのままでは継承の対象とすることはできない。また、机という概念は「机」に当てはまるだけで、「脚」や「引出し」に机という概念を継承することはできない。

(3) 包含関係

包含関係は、下位の対象が上位の対象の一部分を構成するという関係を表している。これはPART-OF関係と呼ばれるものの一部である。

包含関係では、対象の抽象度に差がなく、包含関係が成り立つので、上位の対象から下位の対象に継承することができない。

例えば、「鞄」に関する知識表現を行なう場合、鞄が持っている属性をその下位の対象である「ノート」や「背広」に継承することはできない。

4.2.3 フレーム間ネットワーク

フレームを用いることによって知識の階層構造をうまく説明することができるが、実際の値の検索に関しては、階層構造だけでは十分な柔軟性を得ることが難しい。階層構造による検索で不十分な場合、対象フレームと類似したフレームの情報を検索することは推論の手段として有力である。あるフレームに類似したフレームのスロット値を検索するため、類似の度合を表す関係を用意し、その関係を利用してフレームの検索を行なうことが考えられる。これは、「ニアミス」という少しだけ異なる対象を表すフレームを差異ポインタで結んで、類似ネットワークを形成することによって可能となる。

4.2.4 付加手続き

フレームでは、事実を表す宣言型知識に加えて、処理手順を表す手続き型知識を記述することができる。これは付加手続きと呼ばれ、デーモンとサーバントがある。

(1) デーモン

デーモンとは、ある一定の条件が満たされたときに、システムが自動的に起動される手続きのことである。以下のデーモンは、フレームシステムの代表的なデーモンで、対応するスロットにアクセスがあったときに自動的に起動される。

- if-needed デーモン： スロット値が参照された時点でまだ値が入っていないとき、自動的に起動する。
- if-added デーモン： スロット値が代入されたとき、自動的に起動される。
- if-removed デーモン： スロット値が削除されたとき、自動的に起動される。

デーモンによって、フレームシステムは静的に知識を表現するだけでなく、動的に知識を利用した推論を行なうことが可能になる。

(2) サーバント

スロットに値を入れる代りに手続きを記述しておく。このような、仕組みをサーバントと呼ぶ。例えば、あるフレームのスロットが参照されたような場合、そこから別のフレームのサーバントに対してメッセージが送られることによって処理が行なわれる。サーバントは無条件かまたは利用者の指示に従って起動される。

4.2.5 デフォルト

人間が何かを見てそれが何であるかを理解する場合や、何かあることを想像するときには、対象に対する全てのことについて正確な情報を持っているわけではない。その時点では手に入らない情報に関しては、すでに知っている類似した情報を利用して必要な情報を仮定していると考えることができる。これは、フレームシステム上でフレームのスロットに具体的値が割り当てられる処理がなされることと対応させることができる。このとき、割り当てられる仮の値をデフォルトと呼ぶ。このようなデフォルトはスロットに弱く結合され、後になって確定的な情報が与えられたとき、それと置き換えられる。

例えば、誰かが「太郎はボールを投げた」と言っているのを聞いた場合、抽象的なボールではなく、テニスボールや野球のボールのような特定のボールを思い浮かべるであろう。そのボールは、デフォルトの大きさ、デフォルトの色、デフォルトの重さなどの属性値をもつはずである。これらの値は、その人の過去の経験で得られた情報に基づいて割り当てられる。

4.3 フレームにおけるあいまい性とその表現

この節では、前節で示したフレームの知識表現において扱う必要のあるあいまい性について述べる。

4.3.1 関係のあいまい性

階層関係に基づいた値の継承は、関係の種類によって制限を受けたり継承自身不可能であったりする。階層関係の種類としては、抽象 - 具体、全体 - 部分、包含関係などがある。それぞれの関係は異なった種類のあいまい性が存在するため、あいまい性の取扱いも、各継承関係ごと、各知識表現の対象ごとに異なる機構が必要である。

以下では、各継承関係に対する解釈とそれが持つあいまい性の評価を行ない、各関係に対する取扱いを示す。

(1) 抽象 - 具体関係とそのあいまい性

抽象 - 具体関係では、下位のクラス間にある共通の属性や値を抽出し、上位のクラスに記述している。従って、この関係にはあいまい性は含まれず、全ての値を上位クラスから継承することができる。

(2) 全体 - 部分関係とそのあいまい性

全体 - 部分関係では、継承することのできる属性に制限がある。上位と下位のクラスは共通の属性を持っているが、その値は常に同じとは限らない。上位と下位のクラスで完全に同じ値となるもの、関連のある価を取るもの、まったく無関係な値をとるものが混在しているため、ある属性は継承可能であるかもしれないし、ある属性は継承不可能であるかもしれない。従って、推論に際しては、属性と値との間にあるこのようなあいまい性を取扱う必要がある。

(3) 包含関係とそのあいまい性

包含関係では、関係付けはクラス内の属性と無関係な要因で決定される。そのため、属性やその値は上位と下位のクラスで異なっているのが一般的である。従って、このような関係では値の継承はできない。包含関係では、上位と下位のクラス間の関係の強さや関係の確からしさといったあいまい性を取扱う必要がある。

4.3.2 値のあいまい性とファジィ述語

専門家が知識表現を行なう場合、その表現はあいまいである。特に自然言語を用いた表現ではそれが顕著である。そのため、専門家からの知識獲得に際しては、システム側であいまいな表現を許すことが重要となる。

あいまいな表現を許すためには、推論機構の拡張だけでなく、あいまい性を評価する方法が必要である。ファジィ述語は、あいまい性評価のための方法の一つである。評価の過程では、値自身のあいまい性に加えて、その値が属性としてどれだけふさわしいか、といったあいまい性を処理する必要がある。

我々は値のあいまい性を2つのタイプに分類した。

(1) 誤差・雑音

誤差・雑音は測定に伴うあいまい性であり、これを完全に取り除くことは原理的に不可能である。この種のあいまい性は、測定機器の性能と要求される測定精度によって、区間、オーダー、分布などを用いて表現することができる。

(2) 感覚量の表現

「高い」、「大きい」といった自然言語を用いて人間の感覚量を値として表現する場合、具体的な基準値や絶対的な定義といったものが示されない場合が多い。

「このテーブルより3cmほど高い」、「この道路より少し広い」といった、具体的な対象との比較表現を用いる場合も多いが、これには、比較の対象もその差分もあいまい性を含んでいる。

4.3.3 推論機能におけるあいまい性

指定のスロットに値が無かったり、スロット自身がないと、フレームシステムは値を得るために推論を起動する。フレームシステムには継承とデーモンという2種類の推論機能を持っている。継承を使った推論では、関係のあいまい性を取扱うための仕組が必要である。デーモン関数の定義はフレームシステムとは無関係であるが、継承を利用して値を得るような場合は関係のあいまい性が問題となる。

4.4 LIFE FEShell ファジィフレームシステム

通常のフレームの概念に基づくシステムは多数インプリメントされているが、利用者の表現しようとする知識には多くのあいまい性が含まれているため、うまく表現できないことが多い。LIFE FEShell FFS は、フレームの概念に基づいて、階層構造を持つ知識を表現するとともに、ファジィ論理 [3-6] と可能性理論 [7-9] を用いてあいまい性を取扱うように拡張を行なっている。

具体的には、LIFE FEShell FFS ではフレームに以下のような拡張をおこなった。

・スロット値にクリスプな値に加えて、可能性分布を許す

・スロットに対するグレードの設定とその値にクリスプな値に加えて、可能性分布を許す

・継承の度合いの設定とその値にクリスプな値に加えて、可能性分布を許す

・あいまい性の表現に対応した推論関数

・可能性分布を取扱うためのタームおよびプレディケートの階層的宣言と利用

以下に特徴を概説する。

4.4.1 データ構造

フレームは、関連する属性とその値の対をひとまとめにしたものと考えることがでる。1 組の属性とその値のことをスロットという。フレームは、複数のスロットを持ち、スロットは複数のファシットを持つことができる。さらにファシットは複数の値を持つ。LIFE FEShell FFS では、値の部分にファジィ集合を扱え、グレード (GR) を付加することができる。

次に LIFE FEShell FFS のデータ構造の概念を示す。

```
(フレーム
 (スロット1
        (ファシット1
                (値1   GR)
                (値2   GR)
                  :
                  :       )
        (ファシット2
                (値1   GR )
                  :
                  :       ))
 (スロット2
        (ファシット1
                (値1   GR)
                (値2   GR)
                  :
                  :       )
        (ファシット2
                (値1   GR )
                  :
                  :       )))
```

4.4.2 あいまい性の取扱い

以下ではLIFE FEShell FFSのあいまい性の取扱いとして、関係のあいまい性、値のあいまい性、推論機能におけるあいまい性に関して述べる。

(1) 関係のあいまい性の取扱い

抽象 - 具体関係にはあいまい性は含まれない。

全体 - 部分関係では、上下クラスの属性の間のあいまい性を取扱う必要がある。これは、スロットごとにグレードを付けることによって表現可能である。

包含関係では、関係の強さや関係の確からしさといったあいまい性を取扱う必要がある。これは、リンクごとにグレードを付けることによって表現可能である。

(2) 値のあいまい性の取扱い

誤差・雑音を含むデータと感覚量の定量化は、スロット値として、クリスプな値に加えて可能性分布を書けるようにすることによって、区間と分布を用いて表現することができる。

感覚量の比較表現は、後述するタームの相対宣言によって表現する。

(3) 推論機能におけるあいまい性の取扱い

継承によって推論を行なう関数に対して、後述する方法によって値と関係のあいまい性を取扱うように機能拡張を行なう。

4.4.3 言語によるあいまい性の表現

LIFE FEShell FFS では値として可能性分布を許すことによって値のあいまい性を扱うことを可能にしている。さらに、より人間に理解しやすい形にするために、例えば可能性分布 {0.5/165, 1/170, 0.5/175} を「about170」というタームとして宣言しておくことによりって、知識表現の中で言語的な表現を行なうことができる。

しかし、言語的な表現は属性ごとに類似の物が多く、全てのタームに対して異なる名前を与えることは、知識の記述量が増えるに従って困難になる。

そこで、同じ言葉に複数の定義を与え、対象に応じてシステムが自動的に最適な定義を選択することを考える。この仕組みを提供すれば、知識提供者にとってより親しみやすいシステムを構築することが可能となる。

タームの宣言を行なうに際して、同じまたは類似の対象では同じ定義を利用する可能性が高く、重複するタームの宣言に対して、関係を経由して継承によって記述量を減らすことは自然である。この場合、意味的に関連があることが継承の条件となるため、抽象 - 具体、全体 - 部分、包含といった関係の種類によらず、独立にタームの宣言を継承によって推論することができる。

プレディケート (ファジィ述語) もタームと言語表現的意味においては同様の性質と継承の条件を持っていると考えられるので、タームと同様の仕組みで宣言することができる。

具体的には、LIFE FEShell FFS には各フレームごと、各スロットごとにそれぞれタームとプレディケートの宣言を行なうことができる枠組がある。さらに、タームやプレディケートがそのフレームで定義されていない場合には、継承するものとし、関係を経由して自動的に検索する推論関数を用意した (図 4.1 参照)。

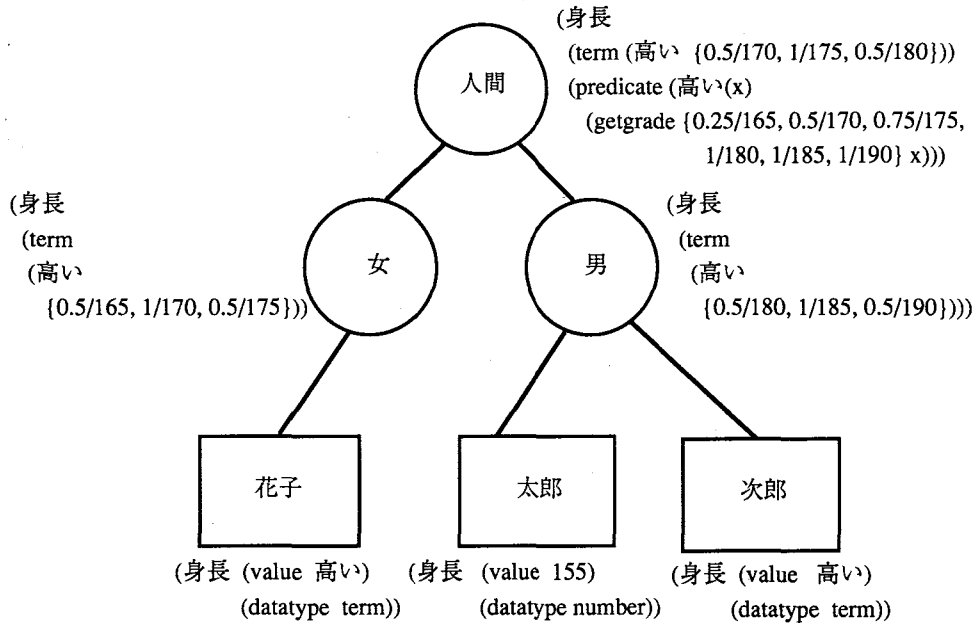

図 4.1 : 階層構造とあいまい性定義の概念

この枠組を利用し、感覚的な表現はタームを各フレームごと、スロットごとに宣言することによって、また、比較に基づく表現はタームの相対的表現利用することによって、自然に表現することができる。一般的な知識は、全てのフレームの基となるフレームを設け、そこに知識を可能性分布を用いて記述することができるようにした。

4.4.4 タームの宣言

タームは、各フレームごと、各スロットごとに宣言することができる。フレームでの宣言には予約スロット名「define」を、スロットでの宣言には予約ファシッと名「term-set」を用いる。

(term-set (term名 ファジィ集合))

さらに、タームは上位のフレームに宣言された同名のタームを利用して、相対的に現在のフレームの中にタームを宣言する仕組みがある。

(term-set (term名 (rel-def 演算関数 term または数)))

これは、上位フレームに定義されているタームの値をもとにして、相対的な値を演算関数と引き数によって求め、それがこのターム名の値となることを意味する。

次にターム宣言の例を示す。

```
(man
  (sub (value (boy 0.5)))
  (define (term-set (tall {0.5/50 1/60 0.5/70})))
  (height (value 175)
          (term-set (tall {0.5/170 1/175 0.5/180}))))
(boy
  (super (value (man  0.5)))
  (sub (value (taro 0.9)))
  (height (term-set (tall (rel-def - about5))))
          (about5 {0.5/4 1/5 0.5/6})))
```

上のようにフレームが定義されている場合、boy の height の term の tall の値は、上位フレーム man の height の term の tall の値から about5 を引くことによって得られるので、次のようになる。

{0.5/164 0.5/165 0.5/166 0.5/169 1/170 0.5/171 0.5/174 0.5/175 0.5/176}

4.4.5 プレディケートの宣言

プレディケートは、各フレームごと、各スロットごとに宣言することができる。フレームでの宣言には予約スロット名「define」を、スロットでの宣言には予約ファシッと名「predicate-set」を用いる。以下にプレディケート宣言の例を示す。

```
(man
  (define (predicate-set (tall (x) (getgrade {0.5/40 0.6/45 0.7/50 0.8/55 1/60} x))))
  (height (value 175)
          (predicate-set (tall (x) (getgrade {0.5/156 0.6/157 0.7/158 0.8/159 1/160} x)))))
```

ここで、getgrade とは引数に対するグレード値を返す関数である。

4.4.6 クラスとインスタンス

フレームをクラスとして定義する場合、LIFE FEShell FFS の全ての関数を利用することができる。フレームをインスタンスとして定義する場合、フレームはその下位としてクラスやインスタンスを持つことができないので、リンク関係の関数などに制限が生じる。これを識別するために、予約スロット名「instance」を用いる。次にクラス man とインスタンス taro の例を示す。

```
(man
  (sub (value (taro 0.9)))
  (define (term-set (tall {0.5/50 1/60 0.5/70})))
  (height (predicate-set (tall (x) (getgrade {0.8/169 0.9/170 1.0/171} x)))
          (term-set (tall {0.9/169 1.0/170 0.9/171}))))
```

```
(taro
 (super (value (man 0.9)))
 (instance t)
 (height (value tall)
         (datatype term)))
```

4.4.7 推論関数 get-value

LIFE FEShell FFS は、あいまい性を含んだ知識を用いて推論を行なうため、専用の推論関数を持っている。まず、 get-value について述べる。

get-value は指定されたフレームの継承を考慮してスロット値を求める検索関数である。引数としてフレーム名とスロット名をとる。 get-value には次のような特徴がある。

・スロット値として、可能性分布 (ターム) を扱うことができる

・タームの宣言の検索を行なう

・グレードで可能性分布を扱うことができる

スロット値のデータ型が term に指定されている場合、上位フレームに定義されているタームの値を検索し、その値を返す。タームの値の検索方法は、対象フレームの上位フレームの対象スロットの term-set ファシット、 system-frame の対象スロットの term-set ファシット、対象フレームの上位フレームの define スロット、 system-frame の define スロットの順に検索し、みつからない場合は処理不能とする。

(1) スロット値が数値の場合の処理例

```
(woman
  (sub (value (hanako 0.7)))
  (height (predicate-set (tall (x) (getgrade {0.5/156 0.6/157 0.7/158 0.8/159 1/160 1/161} x)))
          (term-set (tall {0.8/159 1/160 0.8/161}))))
(hanako
  (super (value (woman 0.7)))
  (instance  t)
  (height (value 159)))
```

上のように、フレームが定義されている場合、このフレームに対して

(get-value hanako height)

の処理を考える。フレーム hanako の height スロットの値は 159 であるので、これを値として得る。

```
<cl>    (get-value hanako height)     ;花子の身長は?
<cl>    159
```

(2) スロット値がタームの場合の推論例

```
(man
  (sub (value (taro 0.3)))
  (height (predicate-set (tall (x) (getgrade {0.8/168 0.9/169 1.0/170 1.0/171} x)))
          (term-set (tall {0.9/169 1.0/170 0.9/171}))))
(taro
  (super (value (man 0.3)))
  (instance t)
  (height (value tall)
          (datatype term)))
```

上のように、フレームが定義されている場合、このフレームに対して

(get-value taro height)

の処理を考える。まずフレーム taro の height スロットの値は tall であるので、これを値として得る。この値のデータ型は term なので、上位フレームに定義されているタームの tall の定義

{0.9/169 1.0/170 0.9/171}

を得る。これとリンクグレードを組み合わせたものを値として返す。リンクが多段の場合は、継承の度合いをデフォルトとして min を用いて計算するが、今回は継承が 1 段であるので、その値は 0.3 である。従って、答は

({0.9/169 1.0/170 0.9/171} 0.3)

となる。

```
<cl>    (get-value taro height)                ;太郎の身長は?
<cl>    ({0.9/169 1.0/170 0.9/171} 0.3)
```

(3) 多重継承の場合の推論例

多重継承の場合、 get-value は、リンクグレードの 1 番大きい上位フレームからリンクグレードの大きい順番に上位フレームを探し、それらの値にデフォルトで定義している計算関数を演算したものを値として返す。ただし、求めた値のデータ型が term の場合はさらに、上の方法で検索したタームの定義を値の計算に利用する。

```
(man
  (sub (value (japanese 0.7) (boy 0.8)))
  (height (predicate-set (tall (x) (getgrade {0.5/165 0.7/170 1/175 1/180} x)))
          (term-set (tall {0.5/170 1/175 0.5/180}))))
(japanese
  (super (value (man 0.7)))
  (sub (value (jiro 0.8))))
(boy
  (super (value (man 0.8)))
  (sub (value (jiro 0.6)))
  (height (term-set (tall {0.5/165 1/170 0.5/175}))))
```

```
(jiro
  (super (value (boy 0.6) (japanese 0.8)))
  (instance t)
  (height (value tall)
          (datatype term)))
```

上のように、フレームが定義されている場合、このフレームに対して

(get-value jiro height)

の処理を考える。フレーム jiro の height スロットの値は tall であるので、これを値をして得る。この値のデータ型は term なので、上位フレームからタームの tall の定義を探すことになる。

まず、jiro フレームの上位フレームのうちリンクグレードの大きい japanese フレームから検索を行なう。japanese フレーム経由でもっとも始めに見つかるタームの tall の定義は

{0.5/170 1/175 0.5/180}

である。これとリンクのグレードを組み合わせる。リンクグレードはデフォルトとして min 演算を行なうので、

({0.5/170 1/175 0.5/180} 0.7)

を得る。この値を作業変数に入れ、次の検索に移る。

次に、jiro フレームの上位フレームのうち次にリンクグレードの大きい boy フレーム経由の検索を行なう。boy フレームにはタームの tall の定義があるので、その値とリンクのグレードを組み合わせ、

({0.5/165 1/170 1/165} 0.6)

を得る。この値を作業変数に追加する。

作業変数には jiro フレームの上位フレーム全ての検索結果が入ることになる。結局、この例では japanese フレーム経由、boy フレーム経由で求めた値が入っており、作業変数の内容は、

(({0.5/170 1/175 0.5/180} 0.7) ({0.5/165 1/170 1/165} 0.6))

である。

作業変数のデータに適当な処理を行ない最終的な答を得る。処理の関数をうまく記述することによって多重継承の特徴を生かした処理を行なうことができる。この例題のデフォルトの計算関数は、単純に各データのグレード部分を比較し、値が max となるデータを採用するので、演算結果は

({0.5/170 1/175 0.5/180} 0.7)

となる。

```
<cl>    (get-value jiro height)                ;次郎の身長は?
<cl>    ({0.5/170 1/175 0.5/180} 0.7)
```

4.4.8 推論関数 comp-value

comp-value は get-value によってスロット値を検索し、その値と指定されたプレディケートを比較する関数である。これにより、「太郎の身長は高いか?」といった質問が可能となる。

comp-value はまず、get-value によって値を求め、指定されたプレディケートをタームと同じ方法で検索し、求めた値と指定されたプレディケートの演算結果を答として返す。この時、ターム、プレディケートはフレームの継承を利用して検索する。

(1) スロット値が数値の場合の推論例

```
(woman
  (sub (value (hanako 0.7)))
  (height (predicate-set (tall (x) (getgrade {0.6/157 0.7/158 0.8/159 1/160 1/161} x)))
          (term-set (tall {0.8/159 1/160 0.8/161}))))
(hanako
  (super (value (woman 0.7)))
  (instance t)
  (height (value 159)))
```

上のように、フレームが定義されている場合、このフレームに対して

(comp-value hanako height tall)

の処理を考える。まず、フレーム hanako の height スロットの値は 159 であるので、これを値として得る。次に、comp-value の引数 tall は comp-value の定義からプレディケートなので、tall というプレディケートを探すことになる。この例では上位フレームに定義されているプレディケートの tall の定義

(tall (x) (getgrade {0.6/157 0.7/158 0.8/159 1/160 1/161} x))

を得る。そして、プレディケートである tall と 159 を演算し、演算結果 0.8 を得る。これとリンクグレードとを組み合わせて答として返す。この例では値は 0.7 であので、答は

(0.8 0.7)

となる。

```
<cl>    (comp-value hanako height tall)        ;花子の身長は高いか?
<cl>    (0.8 0.7)
```

(2) スロット値がタームの場合の推論例

```
(man
  (sub (value (taro 0.5)))
  (height (predicate-set (tall (x) (getgrade {0.5/169 0.8/170 1/171 1/172} x)))
          (term-set (tall {0.3/169 1/170 0.5/171}))))
(taro
  (super (value (man 0.5)))
  (instance t)
  (height (value tall)
          (datatype term)))
```

上のように、フレームが定義されている場合、このフレームに対して

(comp-value taro height tall)

の処理を考える。まずフレーム taro の height スロットの値は tall であるので、これを値として得る。この値のデータ型は term なので、上位フレームに定義されているタームの tall の定義

{0.3/169 1/170 0.5/171}

を得る。次に、comp-value の引数のプレディケート tall に対しては、上位フレームに定義されているプレディケートの tall の定義

(tall (x) (getgrade {0.5/169 0.8/170 1/171 1/172} x))

を得る。そして、拡張原理を用いてタームの tall の値とプレディケートの tall を演算する。

$$\begin{aligned} \text{tall(tall)} &= \text{tall}(\{0.3/169\ \ 1/170\ \ 0.5/171\}) \\ &= \{0.3/\text{tall}(169)\ \ 1/\text{tall}(170)\ \ 0.5/\text{tall}(171)\} \\ &= \{0.3/0.5\ \ 1/0.5\ \ 0.5/1\} \end{aligned}$$

これにリンクグレードを組み合わせる。継承のグレードは 0.5 であるので、答えは

({0.3/0.5 1/0.5 0.5/1} 0.5)

となる。

```
<cl>    (comp-value taro height tall)   ;太郎の身長は高いか?
<cl>    ({0.3/0.5 1/0.5 0.5/1} 0.5)
```

(3) 多重継承の場合の推論例

```
(man
  (sub (value (japanese 0.7) (boy 0.8)))
  (height (predicate-set (tall (x) (getgrade {0.5/165 0.7/170 1/175 1/180} x))))
          (term-set (tall {0.5/170 1/175 0.5/180})))
(japanese
  (super (value (man 0.7)))
  (sub (value (jiro 0.8))))
(boy
  (super (value (man 0.8)))
  (sub (value (jiro 0.6)))
  (height (term-set (tall {0.5/165 1/170 0.5/175}))))
(jiro
  (super (value (boy 0.6) (japanese 0.8)))
  (instance t)
  (height (value tall)
          (datatype term)))
```

上のように、フレームが定義されている場合、このフレームに対して

(comp-value jiro height tall)

の処理を考える。フレーム jiro の height スロットの値は tall であるので、これを値をして得る。この値のデータ型は term なので、上位フレームからタームの tall の定義を探し、tall の定義

{0.5/170 1/175 0.5/180}

と、経由したリンクのグレードから

({0.5/170 1/175 0.5/180} 0.7)

を得る。そして、comp-value の引数プレディケート tall の値を japanese フレームから検索し、プレディケートの tall の定義にリンクのグレードを考慮し、

((tall (x) (getgrade {0.5/165 0.7/170 1/175 1/180} x)) 0.7)

を得る。

求めたターム tall とプレディケート tall の定義を用いて一変数関数に対する拡張原理の計算を行なうと、

{0.5/0.7 1/1}

となる。この値と継承の度合いを組み合わせ、

({0.5/0.7 1/1} 0.7))

を得る。この値を作業変数に入れ、次の検索に移る。

次に、boy フレーム経由の検索を行なう。boy フレームにはタームの tall の定義があるので、その値とリンクのグレードを組み合わせ、

({0.5/165 1/170 1/165} 0.6)

を得、comp-value の引数プレディケート tall の定義を boy フレームから検索、その値にリンクのグレードを組み合わせ、

((tall (x) (getgrade {0.5/165 0.7/170 1/175 1/180} x)) 0.6)

を得る。求めたターム tall とプレディケート tall の定義を用いて一変数関数に対する拡張原理の計算を行なうと、

{0.5/0.5 1/0.7 0.5/1}

となり、この値にリンクのグレードを組み合わせ、

({0.5/0.5 1/0.7 0.5/1} 0.6)

を得る。この値を作業変数に入れる。

作業変数には japanese フレーム経由、boy フレーム経由で求めた値が入っており、作業変数の内容は、

(({0.5/0.7 1/1} 0.7) ({0.5/0.5 1/0.7 0.5/1} 0.6))

である。この例題のデフォルトの計算関数は、単純に各データのグレード部分を比較し、値が max となるデータを採用するので、演算結果は

({0.5/0.7 1/1} 0.7)

となる。

```
<cl>    (comp-value jiro height tall)   ;次郎の身長は高いか?
<cl>    ({0.5/0.7 1/1} 0.7)
```

4.4.9 予約語

LIFE FEShell FFS は以下のような予約語を用意している。これにより、一般的なフレームシステムの機能に加え、あいまい性の階層的記述やその検索を行なうことが可能になっている。

(1) スロット名

mother, super	上位フレームを定義する。
daughter, sub	下位フレームを定義する。
instance	インスタンス・フレームを定義する。
define	ターム、プレディケートをフレームごとに定義する。

(2) ファシット名

value 値であることを定義する。

default 値がデファルト値であることを定義する。

term-set ファジィ集合を定義する。

predicate-set ファジィ述語を定義する。

datatype 値のデータタイプを定義する。

(3) データタイプ

number 値のデータタイプが数値であることを定義する。

term 値のデータタイプがタームであることを定義する。

string 値のデータタイプが文字列であることを定義する。

(4) フレーム名

system-frame 全フレームのターム、プレディケート、グレードを定義するフレーム。

4.4.10 システム変数

システム変数 *system-val* は、システム・スイッチとして次のようなものを持っていて、ユーザが好みのシステムの状態を選択できるようになっている。

warning 異常状態の警告を表示するかしないかのスイッチ。

t 表示する。

nil 表示しない。

inherit 推論時に継承をすることを許すかどうかのスイッチ。

t 継承を考慮する。

nil 継承を考慮しない。

next-state システムが次の状態に移行するときに次の状態に行くことの表示をするかどうかのスイッチ。

t 次の状態へ行くことを表示する。

nil 次の状態へ行くことを表示しない。

now-state 現在の処理をしていることを表示するかどうかのスイッチ。

t 現在処理していることを表示する。

nil 現在処理していることを表示しない。

old-frame フレームに書き変えがあったとき、古い値を表示するかどうかのスイッチ。

t フレームを表示する。

nil フレームを表示しない。

state-print 推論の状態の過程を表示するかどうかのスイッチ。

t 推論状態の過程を表示する。

nil 推論状態の過程を表示しない。

4.5 おわりに

この章では、フレームシステムによって扱うことのできる値および関係に関するあいまい性について分析し、LIFE FEShell FFS で採用しているあいまい性の取扱い機能について述べた。

LIFE FEShell FFS では、値や関係のあいまい性を取扱うために、可能性分布による表現を採用した。これにより、誤差や感覚的な表現が可能になった。

また、あいまい性をタームやプレディケートといった形で言語的に表現することを可能とした。さらに、タームやプレディケートの宣言にあたっては、同一の言葉に対して複数の宣言を許し、対象に応じて適切な宣言を採用するよう、フレーム間の関係を利用してあいまい性の宣言の継承、階層的管理、推論などの枠組みを提供している。これにより、知識提供者にとってより親しみやすいシステムを構築することが可能となった。

第5章　開発環境

5.1 はじめに

この章ではLIFE FEShellの機能の内、その開発環境[33, 37, 38]について述べる。まず、従来のシェルとファジィシェルに関して開発環境の観点から分析を行ない、問題点および解決案とLIFE FEShell開発環境の設計目標を示す。次に、Unix上でX Windowを使ったユーザインタフェースを提供するに際して、従来のプロセス間通信における問題点を指摘した上で、LIFE FEShellで用いた方法について述べ、各オブジェクトの編集の実際とインプリメントの詳細を述べる。

5.2 問題点の分析と設計目標

この節では、従来のシェルとファジィシェルに関して開発環境の観点から分析を行ない、問題点および解決案とLIFE FEShell開発環境の設計目標を示す。

5.2.1　開発環境に関する問題点の分析

開発環境は、推論エンジンとユーザインタフェースの2つから構成される。推論エンジンは、推論やデバッグ機能を提供し、ユーザインタフェースは、推論結果、推論知識やデバッグ情報の入出力と編集をサポートする。

以下では開発環境の観点から、知識表現、ユーザインタフェース、推論エンジンについての問題点の分析を行なう。

(1) 知識表現

ファジィシェルでは、あいまいな知識を取り込むため、従来のシェルに比べてその知識表現がある程度複雑になってしまう。そのためファジィシェルでは 1) 知識表現の書式が複雑になるという問題が生じる。それに伴って 2) 知識が直感的にわかりにくく、3) 入力の誤りが増える傾向がある。複数の知識表現をサポートする場合、表現ごとに異なった形式で入力する必要があり、1)、2)、3) はさらに深刻な問題となる。

あいまい性を表現するために各システムで色々な方法が採用されている。メンバーシップ関数を定義したり、可能性分布 [63, 28] であいまい性を表現することも多い。この場合、4) メンバーシップ関数、可能性分布の入力・編集が煩雑であるという問題が生ずる。

以上の問題の解決には、知識表現の形式の改善などでは限界があり、開発環境を整備する必要がある。1)、2)、3) は知識表現にあいまい性を取り込んだために必然的に発生した問題であり、開発環境で知識表現の形式を意識せずに入力を可能にすることが有効である。4) に対しては、知識表現に必要な情報量が多いために発生した問題であり、開発環境でそれをサポートする機能を提供することが有効である。

(2) ユーザインタフェース

コンピュータ上における作業環境として、GUI (Graphical User Interface) の有効性が認識され、エキスパートシステムにおいても、GUI を採用するものが増えている。しかし、従来のシェルで GUI を採用したものの中にも 1) 異なる形式の知識ごとに異なるユーザインタフェースであるものが多い。また、ファジィシェルでは 2) GUI を採用したものはほとんどない。

GUI は知識の入力を行なう時だけでなく、知識の入力・編集、推論過程・推論結果の表示にも有効である。知識の入力・編集を行なう場合にはユーザインタフェース側が主導的であるが、推論過程・推論結果を表示する場合には推論エンジン側が主導的であるので、3) ユーザインタフェースモジュールから推論エンジンを制御するだけでは不十分である。

従って、1)、2) より、ユーザインタフェースとして GUI を採用し、異なる形式の知識に関しても統一的な操作で入力・編集ができる環境が望まれる。また、3) よ

り、推論エンジンとユーザインタフェースモジュールの両方から双方向的に制御できる必要がある。

(3) 推論エンジン

推論エンジンは推論方式の変更に対して柔軟に対応できることが望ましい。柔軟性は、開発言語としてインタプリタ型高水準言語の Lisp や Prolog などを利用することによってある程度対応できる。しかし、GUI の実装と言う面から見ると、このような言語は 1) GUI を利用するためのインタフェースが提供されていないか不十分なことが多い。2) 処理速度に問題がある。また、推論エンジンとの境界があいまいになり 3) デバッグが困難になる。

OS としてUNIXを利用する場合、ユーザインタフェース開発には C 言語が一般的によく使われている。C 言語は、ウインドウシステムのライブラリを直接利用できるため 4) 開発効率が上がるし、5) 処理速度が速い。推論エンジンと完全に独立になるので、6) ユーザインタフェース開発のデバッグ作業が容易になる。7) 推論エンジンとユーザインタフェースを並行して開発することができるなどの利点がある。C 言語を用いる欠点として、推論エンジンとユーザインタフェースのプログラムが独立になるため 8) 通信による情報交換が必要である。C 言語はプロセス間通信のインタフェースを備えているが、他の言語の多くは不完全で、プロセス間通信を行なう仕組みの開発が大きな負担になることが予想される。

従って、1) から 7) より、推論エンジンとユーザインタフェースモジュールは独立に開発することが望ましい。8) に対しては解決策を提案する [64]。

5.2.2 設計目標

上述した問題点の分析とそれに対する考察より、ファジィシェルでは特にユーザインタフェースが開発環境の改善に重要な役割をはたすものと考えられる。そこで、以下の点を開発環境に対する設計目標とした。

1) 知識表現の形式をユーザインタフェースに反映させ、入力・編集を助けること。

2) メンバーシップ関数や可能性分布の入力・編集が容易に行なえること。

3) 複数の知識表現の編集に際して統一的な操作で入力・編集が行なえること。

4) 推論エンジンとユーザインタフェースモジュールの両方から双方向的に制御できること。

5) 推論エンジンとユーザインタフェースモジュールが独立であること。

以上の目標を達成する目的でLIFE FEShell用開発環境の設計・実装を行なった。開発環境はエディタを拡張する形で実現しているので、これをオブジェクトエディタ (OE) と呼んでいる。

5.3 通信機能のインプリメント

設計目標5) を実現する上で問題となる通信機能のインプリメントに際して、従来のプロセス間通信に対する問題点を指摘し、LIFE FEShellをインプリメントするに際して用いた解決策について述べる。

5.3.1 プロセス間通信の方法と問題点

Unix上でX Windowを使ったユーザインタフェースを提供するようにアプリケーションの機能拡張をするに際して、Unixはプロセス間通信の方法として、パイプ、ソケット、ストリーム、共有メモリなどを提供している。これらの方法のいずれを利用する場合も、各プロセスごとに通信のための特別な処理を必要とし、例えば、割り込み処理や、ループでの入力の監視などを、通信を行なうプロセスごとに記述する必要がある。

しかし、ベースとしてLispやPrologなどの言語など(以下ベース処理系と呼ぶ) を用いてアプリケーション (以下、ベース処理系と開発部分を含めて単にアプリケーションと呼ぶ) を開発している場合には、これらUnixのプロセス間通信機能を利用するための組み込み機能は提供されていないことが多い。そこで、他のプロセスとの入出力も処理できるようにベース処理系自身を拡張する必要が生ずる。一般的な例を図5.1に示す。

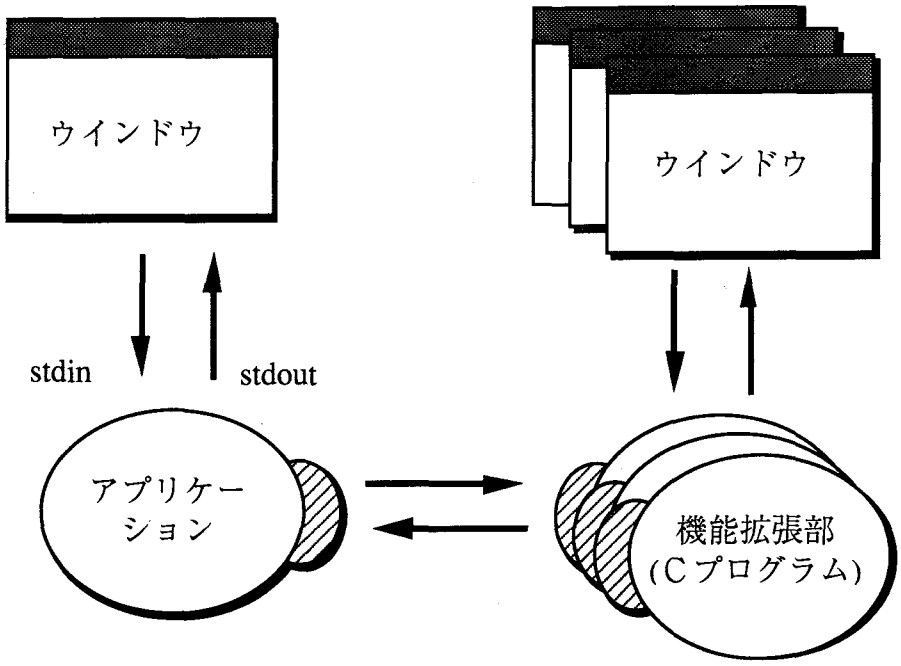

図 5.1 : 通常の通信方法

図 5.1 において、機能拡張部はユーザインタフェースとして各種のウインドウを提供している。従って、アプリケーションは、利用者とは X Window のターミナルウインドウから標準入出力 (stdin と stdout) を利用して情報交換し、機能拡張部とは通信用に記述された専用インタフェースを用いて情報交換を行なっている (一般的には、機能拡張用の任意のプログラムであってもよい)。図 5.1 では機能拡張部はひとつのプロセスであるが、複数のプロセスからなる場合も同様である。

通信用インタフェースを開発するにあたって、プロセス間通信は Unix が提供するいずれの機能を用いてもよいが、アプリケーションと機能拡張部の双方に処理部 (図 5.1 の斜線部分) が必要となり、ベース処理系を拡張することになる。この拡張にはベース処理系に対する高度の専門知識が必要であり、各処理系固有の問題を解決する必要が生じる。

5.3.2 提案する通信方法

ベース処理系の拡張を行なわないで、利用者だけでなく、他のプロセスとも、標準入出力を利用して通信を行ない、機能拡張を行なうことを考える。

今回は、X Window を用いたシステムを前提とするため、アプリケーションはターミナルウインドウを利用して利用者との情報交換を行なっている。標準入出力を

含む管理用モジュールを機能拡張部側に設ければ、アプリケーションの標準入出力も機能拡張部で管理することができる (図 5.2 参照)。

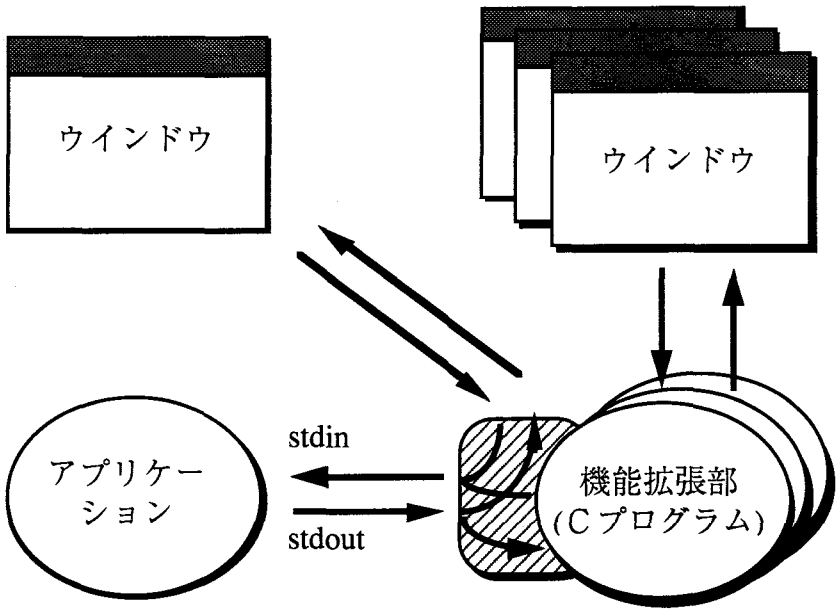

図 5.2 : 採用した通信方法

利用者からアプリケーションへの入力は、機能拡張部の通信用インタフェース部を経由して stdin を通してアプリケーションに送られる。他のプロセスからの情報もこれと同様に送られる。アプリケーションから機能拡張部に対しては、エスケープシーケンスを付加することによって、stdout からの情報が利用者に対する表示情報なのか他のプロセスに対するものなのかの識別を行なう。

図 5.2 に示すとおりこの方法では、通信用インタフェースを全て機能拡張部側で記述することができる。そのため、ベース処理系は外部との通信を標準入出力だけで行なうことができ、通信のための拡張を行なう必要がなくなる。さらに、開発が機能拡張部に限定されるため、デバッグが容易になるなどの開発効率の向上が期待される。

この方法の利点は、ベース処理系を拡張し、通信のための特別な処理を記述する必要がないことである。

利用者からの入力と他プロセスからの入力を識別する必要がない場合には、アプリケーション上の記述を全く変更する必要がない。機能拡張の範囲がユーザインタフェースの向上程度である場合、アプリケーションにほとんど手を加えることなく

拡張することができる。さらに複雑な機能拡張の場合も、通信にあたって通信相手を指定する程度の処理で拡張することができ、従来の方法に比べ、アプリケーション側の作業量が大幅に少なくなる。

5.4 開発環境の実際

この章では、OEが扱うべきオブジェクトの詳細とOEを用いた編集の実際について述べる。その後、設計目標1)から4)をOEでどのように実現したかについて述べ、OEの詳細について述べる。

5.4.1 オブジェクトとその編集

OEが扱うべきオブジェクトを次に示す。

・ルール
・ワーキングメモリ(WM)
・ターム
・プレディケート
・フレーム

さらに、FPSのルール実行部やFFSのサーバントに記述するための関数の宣言と、FPSやFFSの動作オプションを指定するために次のものが必要である。

・関数
・システムオプション

設計目標3)を実現するために、オブジェクトにアクセスするための基本的なユーザインタフェースは全てのオブジェクトに対して共通である。

設計目標1)、2)実現するために専用のウインドウを設計した。

以下では、各オブジェクトに対するインタフェースについて述べる。OEの具体的なユーザインタフェースを示すため、ルール、ターム、プレディケート、フレームの各オブジェクトでは実際の入力・編集の手順を示す。

(1) ルール

FPSでは、ルールはブロック単位に定義できる。ルールブロックの宣言を次に示す。

(*rule* RBName &conflict CName &parent-rb PRBName [list of rules])

ここで、RBNameはこのルールブロックの名前、CNameは競合解消ルールを表す関数の名前、PRBNameは上位のルールブロックの名前、[list of rules]はIF～THEN～ルールの列である。

ルールの編集では、OEはルール、ルールブロック、競合解消ルール、上位ルールブロックのためのインタフェースを提供する。

ルール関連ウインドウを図5.3に示す。

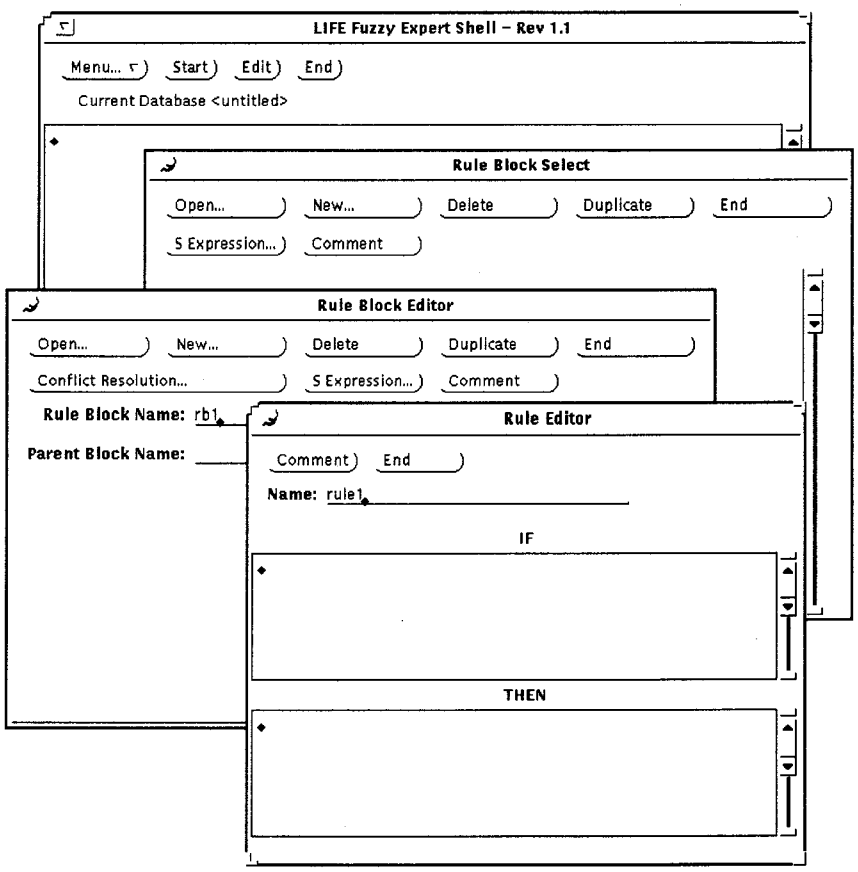

図5.3：ルール関連ウインドウ

OEを利用して新規にルールを作成する場合の手順を示す。

1) MainウインドウのMenuボタンからRule Blockを選択するとRule Block Selectionウインドウが表示される。

2) 新規ルールブロックを作成するため、Rule Block SelectionウインドウのNewボタンを選択すると、Rule Block Nameダイアログによってルールブロック名の入力を求めた後、Rule Selectionウインドウが表示される。

3) 新規ルールを作成するため、Rule Selection ウインドウの New ボタンを選択すると、Rule Name ダイアログによってルール名の入力を求めた後、Rule Editor ウインドウが表示される。

4) Rule Editor ウイドウによって必要な情報を入力する。

(2) ターム

設計目標 2) を実現するため、ターム編集用のユーザインタフェースを提供する。タームとして次の 3 つを考える。

1) 全体集合が数値で、グレード値として数値をとる可能性分布

2) 全体集合が数値以外のもので、グレード値として数値をとる可能性分布

3) その他の方法で定義される可能性分布

このうち、性質のわかっている 1) と 2) に対しては専用のインタフェースを用意した。3) については、将来の機能拡張の面も考慮にいれ、アクセスするためのユーザインタフェースは共通とし、編集に関してはテキストエディタを利用する。

ターム関連ウインドウを図 5.4 に示す。

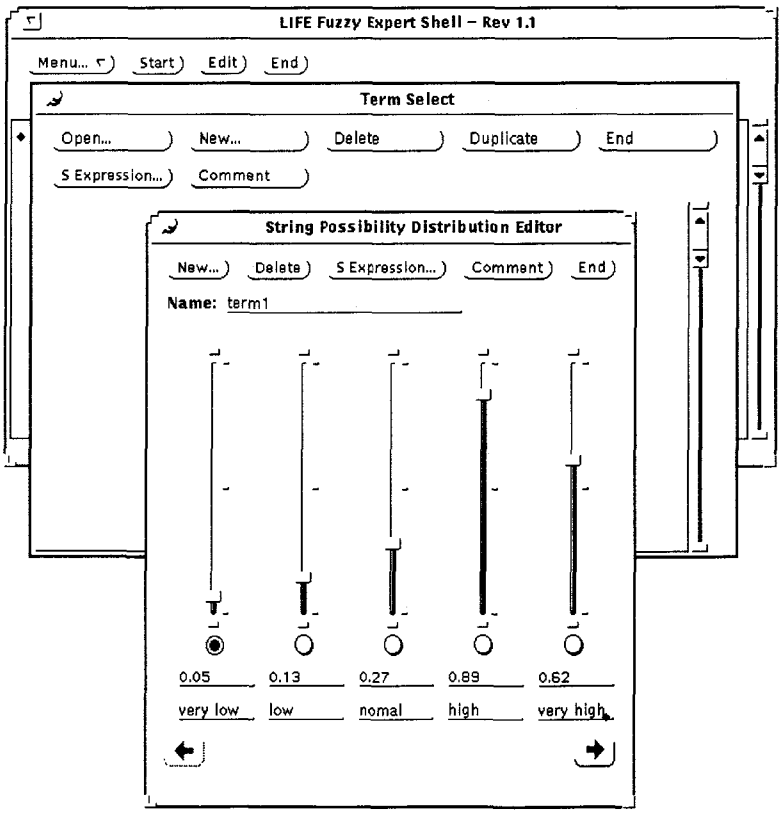

図 5.4 : ターム関連ウインドウ

OE を利用して新規に上の 2) のタームを作成する場合の手順を示す。

1) Main ウインドウの Menu ボタンから Term を選択すると Term Selection ウインドウが表示される。

2) 新規タームを作成するため、 Term Selection ウインドウの New ボタンを選択すると、 Term Name ウイドウが表示される。

3) 数値可能性分布の場合、上限値、下限値、刻み値を入力する。文字列可能性分布の場合、文字列の個数を入力する。ここでは、文字列可能性分布を選択すると String Possibility Distribution Editor ウインドウが表示される。

4) String Possibility Distribution Editor ウインドウで横軸に文字列とその値を入力する。

(3) プレディケート

プレディケートはファジィ述語 (メンバーシップ関数) の集合から構成されている。ファジィ述語は離散表現と連続表現の両方で表現可能である。

設計目標 2) を実現するため、プレディケート編集用のユーザインタフェースを提供する。プレディケイトは、連続表現としてよく用いられる s 関数や pi 関数等を含む次の 10種類を考える。

1) s 関数 (2 次関数) [8] によって定義されるプレディケイト

2) z 関数 (2 次関数) [8] によって定義されるプレディケイト

3) pi 関数 (2 次関数) [8] によって定義されるプレディケイト

4) tr 関数 (1 次関数) [8] によって定義されるプレディケイト

5) trs 関数 (1 次関数) [8] によって定義されるプレディケイト

6) trz 関数 (1 次関数) [8] によって定義されるプレディケイト

7) trp 関数 (1 次関数) [8] によって定義されるプレディケイト

8) 数値を全体集合とし、グレード値として数値を返す専用の関数によって定義されるプレディケイト

9) 数値以外を全体集合とし、グレード値として数値を返す専用の関数によって定義されるプレディケイト

10) その他の方法で定義されるプレディケイト

このうち、性質のわかっている 1) から 9) に対しては専用のインタフェースを提供した。10) については、将来の機能拡張の面も考慮にいれ、アクセスするためのユーザインタフェースは共通とし、編集に関してはテキストエディタを利用する。

プレディケート関連ウインドウを図 5.5 に示す。

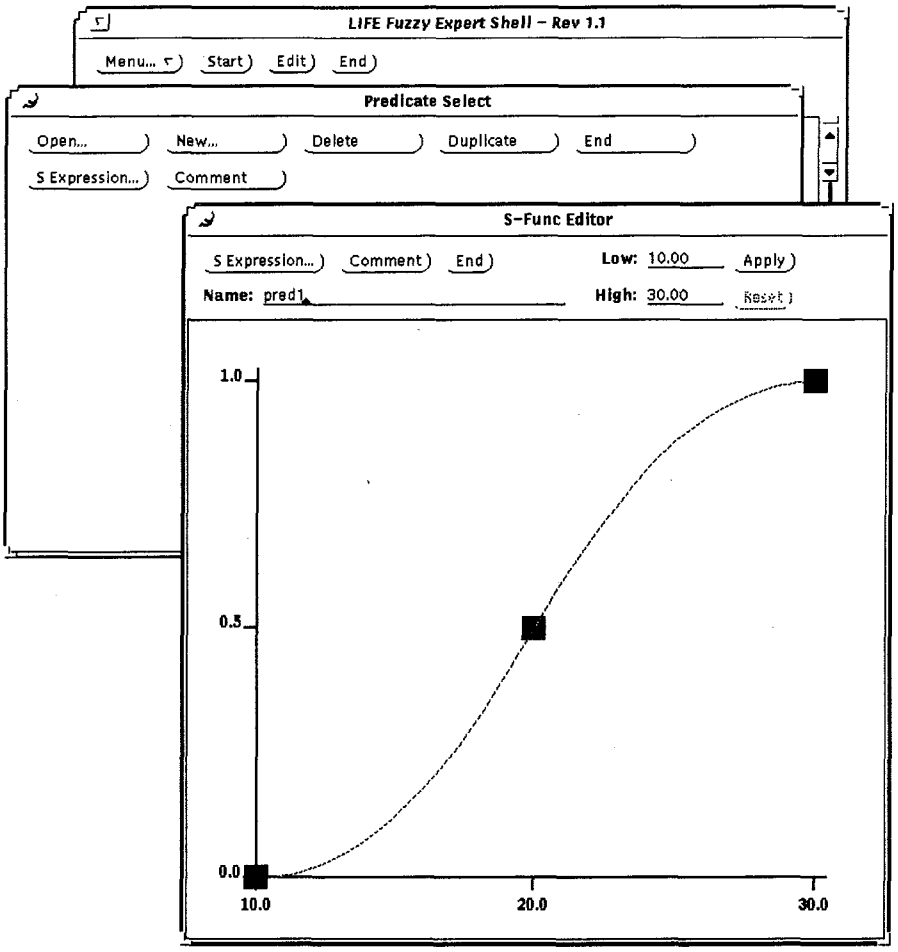

図 5.5 : プレディケート関連ウインドウ

OE を利用して新規にプレディケートを作成する場合の手順を示す。

1) Main ウインドウの Menu ボタンから Predicate を選択すると Predicate Selection ウインドウが表示される。
2) 新規プレディケートを作成するため、Predicate Selection ウインドウの New ボタンを選択すると、Predicate Name ウイドウが表示される。
3) プレディケートの形状によって、必要なボタンを選択する。ここでは、S 関数を選択すると S Function Editor ウインドウが表示される。
4) S Function Editor ウインドウで必要な数値を入力する。

(4) フレーム

LIFE FEShell FFS は一般的なフレームシステムの機能に加え、あいまい性の階層的記述やその検索を行なうことができる。

ターム、プレディケートは、各フレームごとでも、各スロットごとでも宣言することができる。

さらに、タームは上位のフレームに宣言された同名のタームを利用して、相対的に現在のフレームの中にタームを宣言する仕組みがある。

従って、フレームに対して、スロット値の編集に加えて、スロット値に対するグレード、双方向のリンク、リンクに対するグレード、フレーム毎のターム、プレディケートの宣言、スロット毎のターム、プレディケートの宣言、各予約語をサポートする。

フレーム関連ウインドウを図 5.6 に示す。

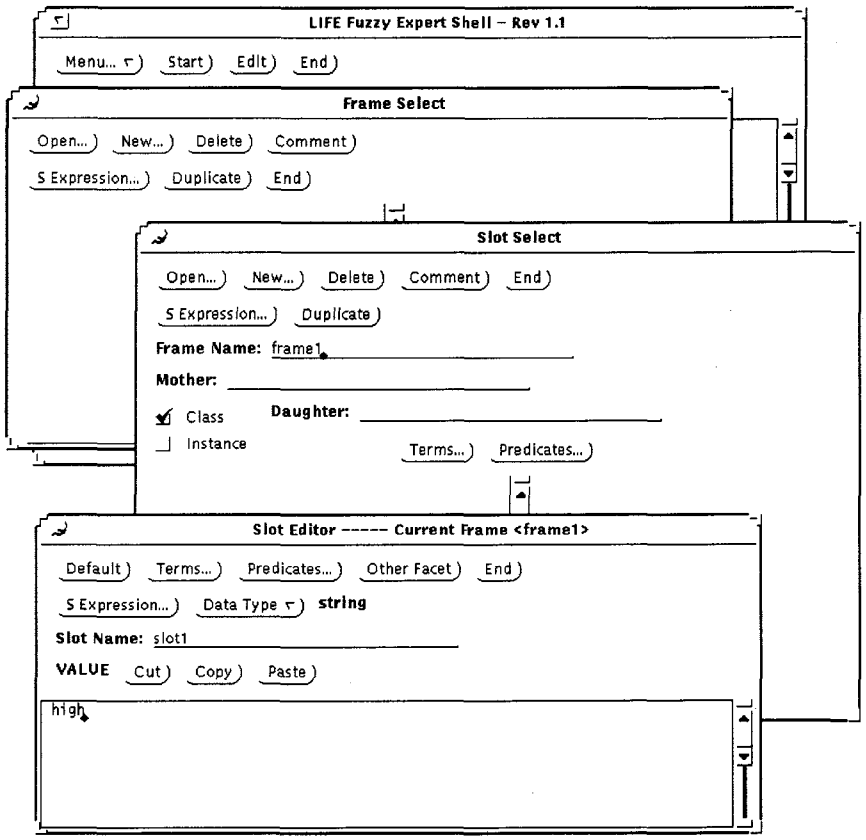

図 5.6 : フレーム関連ウインドウ

以下に OE を利用して新規にフレームを作成する場合の手順を示す。

1) Main ウインドウの Menu ボタンから Frame を選択すると Frame Selection ウインドウが表示される。

2) 新規フレームを作成するため、 Frame Selection ウインドウの New ボタンを選択すると、 Frame Name ダイアログによってフレーム名の入力を求めた後、Slot Selection ウインドウが表示される。

3) Slot Selection ウインドウでは、上位フレーム、下位フレームの指定と、対象フレームに対するターム、プレディケートの宣言が可能である。 Term ボタン、 Predicate ボタンを選択すると、それぞれ Term Selection ウインドウ、Predicate Selection ウインドウが表示される。

4) 新規スロットを作成するため、 Slot Selection ウインドウの New ボタンを選択すると、 Slot Name ダイアログによってスロット名の入力を求めた後、Slot Editor ウイドウが表示される。

5) スロットへの書き込みのほとんどは Value ファシットに対するデータ入力であるので、 Slot Editor ウイドウは Value ファシットの入力領域を持っている。さらに、データタイプの指定、対象スロットに対するターム、プレディケートの宣言が可能である。 Term ボタン、 Predicate ボタンを選択すると、それぞれ Term Selection ウインドウ、 Predicate Selection ウインドウが表示される。

6) 予約されているファシット以外のファシットを作成するため、 Other Facet ボタンを選択すると、 Facet Selection ウインドウが表示される。

7) 新規ファシットを作成するため Facet Selection ウインドウの New ボタンを選択すると、 Facet Name ダイアログによってファシット名の入力を求めた後、Facet Editor ウイドウが表示される。

8) Facet Editor ウイドウに必要な値を入力する。

(5) ワーキングメモリ (WM)

OE は WM ブロックと WM のためのインタフェースを提供する。

WM の宣言を以下に示す。

(*working-memory* WMBlockName WMOption [list of WM])

WMBlockName　　WM ブロックの名前。無くてもよい。

WMOption　　WM ブロックの付加情報。無くてもよい。

[list of WM] データの列。

(6) 関数

主にルールの結論部の関数を記述するための関数をサポートする。アクセスするためのユーザインタフェースは共通とし、編集に関してはテキストエディタを利用する。

(7) システムオプション

デバッグモードの指定や推論過程の表示などFPS、FFSのオプション設定をサポートする。アクセスするためのユーザインタフェースは共通とし、編集に関してはテキストエディタを利用する。

5.4.2 インプリメント

以下では、設計目標1)から4)をOEでどのように実現したかの詳細について述べる。

(1) 専用ウインドウ

ウインドウを設計するにあたって、設計目標1)を実現するためには、1.1)入力可能な情報のテンプレートを作る。1.2)取扱う知識を分析し、頻度の低いものはメニューなどに隠す。1.3)必要な情報のみを入力の時点で指定する、などの方法が考えられる。1.1)では、全ての情報が常に表示されるため煩雑である。1.3)では、どのような情報が編集可能であるのかを知っておく必要があるため、利用に際して教育が必要である。1.2)では設計の時点で取扱う知識の分析が必要であるが、利用者に最も負担が少ないため、OEでは1.2)の方法を採用し、各知識表現に対して専用のウインドウを提供する。これには設計目標2)を実現するためのウインドウが含まれる。

1.2)を選択したため、完全に設計目標3)を実現することは困難である。OEが持つウインドウの種類として、各オブジェクトを選択するためのセレクションウインドウ、各オブジェクトの知識の形式をサポートしたエディタウインドウ、可能性分布やファジィ述語のための専用ウインドウといくつかのダイアログボックスがある

が、ウインドウの基本的なデザインと操作を統一するなど、可能なかぎり共通なインタフェースを提供する。

(2) 動作モード

設計目標 4) および 5) を実現するため、LIFE FEShell では推論エンジンと OE を独立に開発し通信を行なう方法を採用した。従って LIFE FEShell は、推論エンジンスタンドアロンモード、オブジェクトエディタスタンドアロンモード、推論エンジンと OE が通信を行なうノーマルモードの 3 つの動作モードを持つ。ノーマルモードにおける推論エンジンと OE の通信のため専用の通信プロトコルを作成した。

推論エンジンスタンドアロンモードでは、推論エンジンは一般の Unix プロセスとして動作する。このモードは、新しい推論方式の実現など主に推論エンジンの開発に利用される。

オブジェクトエディタスタンドアロンモードでは、X サーバのクライアントとして動作する。このモードはユーザインタフェースを利用して知識の入力・編集を行なうことができ、多量の知識データを一括して入力する時などに利用される。

ノーマルモードの設計にあたって、設計目標 5) を実現するためには、5.3 節で述べたプロセス間通信に標準入出力を利用する方法 [64] を採用する。

ノーマルモード (図 5.7) では OE の子プロセスとして推論エンジンプロセスが動作する。推論エンジンと OE の間の通信チャンネルは一本の pipe によって実現され、表示情報と制御情報の 2 種類の情報を双方向でやり取りする。表示情報は、推論エンジンのためのユーザインタフェースであるメインウインドウ上で利用者から推論エンジンに対するコマンドとして打ち込まれるキー入力や、推論エンジンからメインウインドウに表示するのために送られる文字列である。制御情報は、推論エンジンから OE に対して送られるコマンドや、OE から推論エンジンに対して送られるレスポンスである。推論エンジンと OE はこの 2 種類の情報を識別し処理する必要がある。そこで、エスケープシーケンスを利用して制御情報を表示情報から区別することによって、一本の通信チャンネルによって、全ての情報をやりとりすることを考える。推論エンジンから OE に対して送られるコマンドは「beginning of transmission」からはじまり、改行コード (ASCII LF '¥012') で終わる。「beginning of transmission」は OE コンパイル時に設定する一文字以上の文字列であり、デフォル

トはエスケープコード (ASCII ESC '¥033') である。OE から推論エンジンに対して送られるレスポンスも同様に表現される。コマンドとレスポンスは日本語処理を考慮して1バイトおよび2バイト文字からなる単語またはスペースで区切られたS式のリストである。

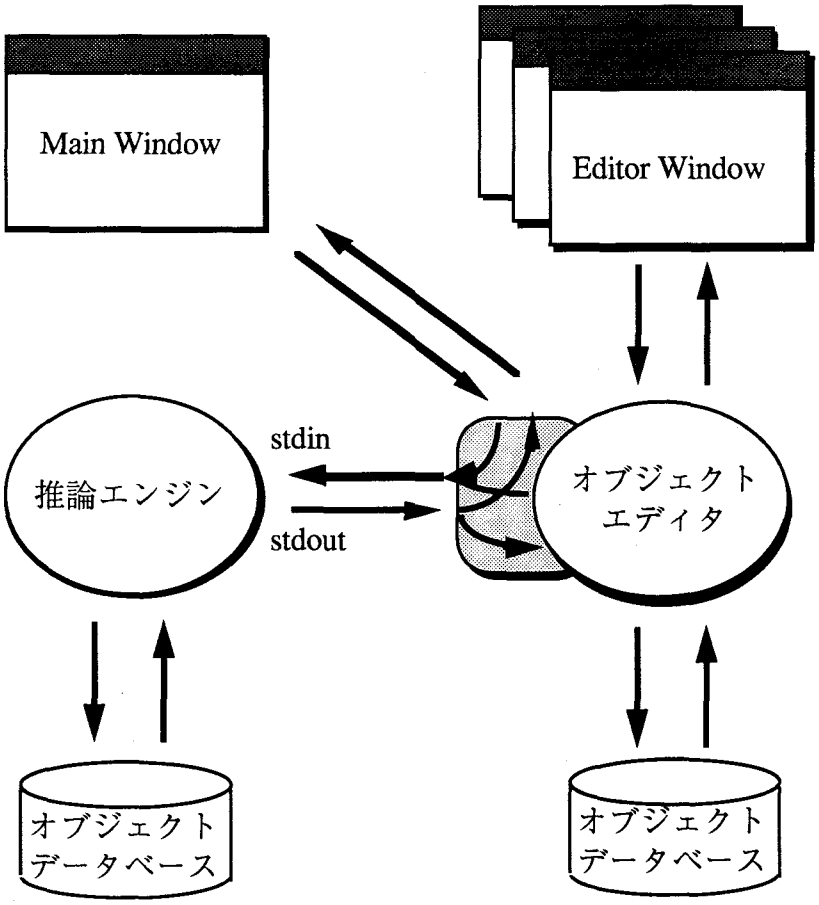

図 5.7 : ノーマルモードの構成

OE からの知識情報は推論エンジンが取扱う形式に変換され、stdin を経由して推論エンジンに送られるため、推論エンジン側では、知識情報が OE からのものか利用者がキー入力したものかの区別をする必要がない。このため、推論エンジンは OE からの入力を管理する部分を持つ必要がなくなった。推論エンジンから、OE に対する制御コマンドを送る場合は、標準のプリント関数を用いて stdout を経由してコマンドを送ることができる。

この方法を採用したため、推論エンジンでは、知識入力の方法として、定まった形式に従ってファイルとして作成された知識を読み込む機能と同様の形式に従って

利用者がキー入力を行なう機能を持っているのみである。前者は主に初期の知識入力に、後者は主に知識のチューニングに利用される。

また、FPS のデバッガの持つ推論のトレース、ブレイクポイントの設定、内部状態の表示、指定サイクル後のブレイク、ヒストリ機能、推論サイクルを戻すといった機能に対するユーザインタフェースの提供も、OE に対する制御コマンドの一部として標準のプリント関数を用いて stdout を経由してコマンドを送ることができる。

このように、双方向通信が必要な場合も、推論エンジン側では開発を大幅に簡略化することができた。これは、プロセス間通信のための開発が不要となり、OE との通信に際して OE のコマンドを管理する部分を開発するだけでよく、通信処理を含めて、Lisp プログラムのレベルですべての処理を記述することができるためである。

(3) 通信プロトコル

定義されたコマンド群をプロトコルという。OE の通信プロトコルには、オブジェクト操作、ウインドウ操作、オブジェクトデータベース操作の 3 種類のコマンドが含まれる。具体的には、オブジェクトの編集・表示、オブジェクトデータベースからの削除、ウインドウのクローズ、移動、サイズ変更、ウインドウの状態確認、推論エンジンと OE のオブジェクトデータベースの同一性を保つための機能などである。オブジェクトの編集・表示には、さらに、OE のオブジェクトデータベースの情報に対するものと、推論エンジンから一時的に与えられた情報に対するものがある。

以下にプロトコルの代表的なコマンドを示す。

(i) edit

これはオブジェクトの編集のために、推論エンジンから指定のオブジェクトに対するウインドウを表示させるためのコマンドを表わす。ワーキングメモリブロック、ルールブロックなどオブジェクト以外の物に対してはセレクションウインドウを表示する。OE のオブジェクトデータベースの情報に対するものと、推論エンジンから一時的に与えられた情報に対するものがある。

以下にOEのオブジェクトデータベース内のルールブロックを編集するための書式を示す。それぞれ、ルールブロックセレクション、ルールセレクション、ルールエディットのウインドウが表示される。

(edit <pos> rule)
(edit <pos> rule <rule block name>)
(edit <pos> rule <rule block name> <rule name>)

ここで<pos>はX windowの書式の位置情報で、省略した場合はデフォルト位置に表示する。

(ii) show

これはオブジェクトの表示のために、推論エンジンから指定のオブジェクトに対するウインドウを表示させるためのコマンドを表わす。ワーキングメモリブロック、ルールブロックなどオブジェクト以外の物に対してはセレクションウインドウを表示する。OEのオブジェクトデータベースの情報に対するものと、推論エンジンから一時的に与えられた情報に対するものがある。

以下に推論エンジンから一時的に与えられたルールブロックを表示するための書式を示す。それぞれ、指定の知識に対するルールブロックセレクション、ルールセレクション、ルールエディットのウインドウが表示される。

(show <pos> rule)
(show <pos> rule <rule block name> (<rule block expr>))
(show <pos> rule <rule block name> <rule name> (<rule expr>))

ここで<pos>はX windowの書式の位置情報で、省略した場合はデフォルト位置に表示する。<rule block expr>、<rule expr>はFPSの書式の知識表現である。

(iii) delete

これはオブジェクトをオブジェクトデータベースから削除するためのコマンドを表わす。指定のWMを削除するための書式を示す。

(delete wm <wm name>)

(iv) close

これは現在表示されているウインドウを推論エンジンからクローズするためのコマンドを表わす。ウインドウの指定はオブジェクトの名前か、ウインドウIDで行な

う。ウインドウ ID はウインドウを開く時に OE によって一意に付けられる。以下にタームとウインドウ ID を用いてウインドウをクローズするための書式を示す。

```
(close term <term name>)
(close window <window id>)
```

5.5 おわりに

複数の知識表現に対して統一的なユーザインタフェースを実現し、各知識表現の書式をサポートすることを通して、あいまいな知識やデータを扱うためのユーザインタフェースを提案した。

設計目標 1) を実現するため、各オブジェクト毎に専用のウインドウを設け、知識表現の形式をサポートした。設計目標 2) を実現するため、タームとプレディケートに専用のグラフィカルなユーザインタフェースを提供した。設計目標 3) を実現するため、オブジェクトの選択、作成、削除など基本的な操作は全てのウインドウで共通になるよう設計した。設計目標 4) を実現するため、通信プロトコルを利用し、推論エンジンからウインドウの表示、移動、サイズ変更、終了などの操作を可能にした。さらに、通信プロトコルには OE のデータベースの操作やユーザインタフェースを一時的に利用するための機能が含まれる。設計目標 5) を実現するため、推論エンジンと OE を別プロセスで動作するように設計した。ここで問題となるプロセス間通信部分の開発負荷を文献 [38] で提案した方法で回避した。

第6章　ファジィルールのチューニング

6.1 はじめに

ファジィルールの実行結果は、1) 条件部の各条件のファジィ集合、2) 各条件のマッチ度から条件部全体のマッチ度を計算する方法、3) 条件部のマッチ度を実行部に反映させる方法 (インプリケーション)、4) 発火の候補となるルールの組み合せ方法 (コンビネーション) により影響を受ける。1) に関してはメンバーシップ関数を定義するパラメータをチューニングする方法など既に多くの研究がなされている [26, 27]。2) については、大部分のシステムが min、max を利用している。これらの結合演算子もチューニング可能であるが、そのような研究は少ない。3) についてはマムダニのインプリケーションを、4) については max を利用しているものがほとんどであり、チューニングに関する研究は少ない。

この章では、結合演算子としてパラメータによって性質が変化する T ノルム (and 演算子に対応する)、T コノルム (or 演算子に対応する) [5] を採用し、結合演算子の自動チューニングの新手法を提案し、シミュレーションによる評価を行なう [42-44]。

6.2 従来の研究と課題

一般のエキスパートシステムでは知識表現としてクリスプなプロダクションルールが利用されている。知識のチューニングは、プロダクションルール自身を追加、削除することによって行なわれている。

ファジィルールのチューニングは、メンバーシップ関数の形状の変更によって行なわれている。これに伴い、従来専門家によって行なわれていたメンバーシップ関数のチューニングを、入出力データから自動的に行なう研究が盛んである [26, 27]。

しかし、メンバーシップ関数を自動チューニングすると、当然ながらメンバーシップ関数の形状が変化してしまうことがあげられる。もともと、メンバーシップ関数は、専門家の知識を基に作成されており、それぞれに意味を持っているはずであるが、自動チューニングは、メンバーシップ関数から意味を奪い取ってしまう結果となる。従って、メンバーシップ関数の自動チューニングを行なった知識に対しては、専門家による再チューニングが非常に困難になってしまう。

条件部全体のマッチ度は各条件を結合演算子によって結合することで得られる。実際には、大部分のシステムは結合演算子としてmin、maxを利用している。これの拡張についてはいくつかの研究が行なわれ、結合演算子として異なる種類のTノルム、Tコノルム[5]を利用すると、まったく同一のルールとメンバーシップ関数を利用しているファジィコントローラの制御性が変化することが報告されている[65-67, 5, 68-70]。これは、結合演算子による知識のチューニングの可能性を示すものと考えられるが、従来の研究では、知識のチューニングに関する考察は十分とはいえない。

また、専門家がプロダクションルールで知識表現を行なうにあたって、複数の条件を条件部に用いる場合が多い。この場合、専門家は各条件間に暗黙の内に複雑な関係を前提としていることが知られている[54, 63]。条件部のマッチ度は、各条件を結合演算子を用いて計算しているが、加藤等の研究[53-56]は、結合演算子として単純なmin、max等だけでは専門家の知識を素直に表現することができないことを示している。各条件間に複雑な関係がある場合、結合演算子をmin、max等に固定したまま、メンバーシップ関数のチューニングだけでこれに対応することができないことは明らかである。

以上まとめると、1) メンバーシップ関数の自動チューニングは知識の保守性を低下させる結果となる。2) 結合演算子による知識のチューニングについては十分考察されていない。3) メンバーシップ関数のチューニングだけでは対応できない場合がある。等の問題が明らかになた。

6.3 結合演算子のチューニング方法

上記の問題点を解決するために、ここでは、パラメータ付きTオペレータのパラメータをバックプロパゲーションを用いて自動チューニングする方法を提案する。

6.3.1 システム構成

問題を単純化するため、図6.1のようなシステムを対象とする[65]。システムはファジィコントローラとプラントの2つから構成される。ここで、$x(n)$は、n番目のサンプリングにおける制御の目標値、$e(n)$はエラー、$de(n)$はエラーの変化、$du(n)$は制御信号、$y(n)$はプラントの出力を表す。

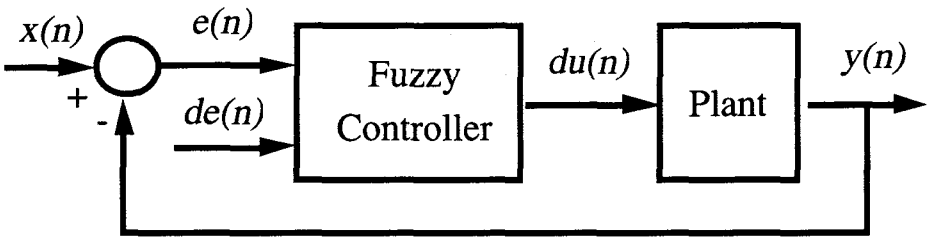

図6.1：システム概要

(1) プラント

結果の一般性を高めるため、次にあげるような、2種類のプラントに対して実験を行なった。

1) 一次遅れプラント： $y' + ay = bu$

2) 二次遅れプラント： $y'' + ay' + by = cu$

ここで、a、b、cは定数を表す。

(2) ファジィコントローラ

ルールの条件部がエラー(e)とエラーの変化(de)、実行部が制御信号(du)からなるファジィコントローラを考える。N個のルールがある場合、i番目のルール

If e is E_i and de is dE_i, then du is dU_i

を表すファジィ関係R_iのメンバーシップ関数μ_{Ri}は以下の式で与えられる。

$$\mu_{Ri}(e,de,du) = I\left(\mu_{Ei}(e),\mu_{dEi}(de),\mu_{dUi}(du)\right)$$

ここで、$I(\bullet,\bullet,\bullet)$は一般的なインプリケーション関数を表している。今回はインプリケーションとしてマムダニのインプリケーションを採用したため、ルール全体のファジィ関係は次の式で表される。

$$\mu_R(e,de,du) = \overset{N}{\underset{i=1}{S}}\left[T\left[\mu_{Ei}(e),\mu_{dEi}(de),\mu_{dUi}(du)\right]\right]$$

ただし、T はTノルムを、S はTコノルムを示している。

エラー、エラーの変化、制御信号がファジィ数 E' 、dE' 、dU' で与えられる場合、式は次のように表される。

$$\mu_{dU'}(du) = \sup_{(e,de)} T[\mu_{E'}(e), \mu_{dE'}(de), \mu_R(e, de, du)]$$

今、ファジィルールとして次の４つを与える。

If e is negative and de is negative, then du is positive,

If e is negative and de is positive, then du is zero,

If e is positive and de is negative, then du is zero,

If e is positive and de is positive, then du is negative.

ここで、negative 、positive 、zero には次のメンバーシップ関数を利用する。

NE : negative error

PE : positive error

NdE : negative change in error

PdE : positive change in error

NdU : negative change in process input

ZdU : zero change in process input

PdU : positive change in process input

図 6.2 、図 6.3 、図 6.4 にそれぞれのメンバーシップ関数の形状を示す。

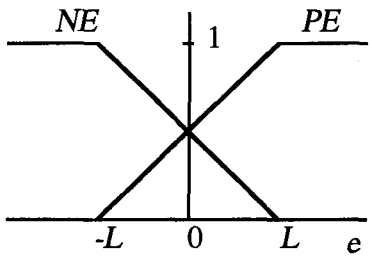

図 6.2 : Membership Function for E.

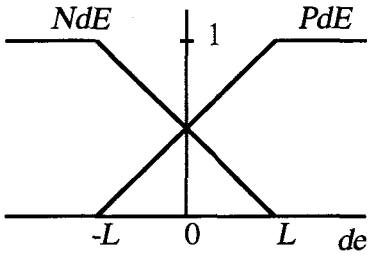

図 6.3 : Membership Function for dE.

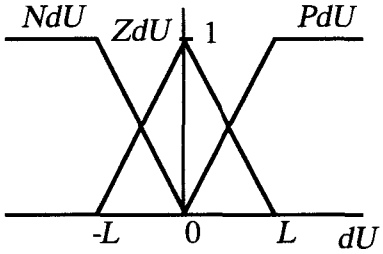

図 6.4 : Membership Function for dU.

$e(n)$ 、 $de(n)$ は次の式で計算する。

$$e(n) = [x(n) - y(n)] \times G_E$$
$$de(n) = [y(n-1) - y(n)] \times G_{dE}$$

ここで、 G_E 、 G_{dE} は定数である。

実際の入力 $e(n)$ 、 $de(n)$ はクリスプな値となるので、 dU' は以下の式となる。

$$\begin{aligned}\mu_{dU'}(du) &= \mu_R(e, de, du) \\ &= S[T[\mu_{NE}(e), \mu_{NdE}(de), \mu_{PdU}(du)], \\ &\quad T[\mu_{NE}(e), \mu_{PdE}(de), \mu_{ZdU}(du)], \\ &\quad T[\mu_{PE}(e), \mu_{NdE}(de), \mu_{ZdU}(du)], \\ &\quad T[\mu_{PE}(e), \mu_{PdE}(de), \mu_{NdU}(du)]]\end{aligned}$$

$du(n)$ は dU' から重心法を用いて計算する。

$$du(n) = \frac{\sum \mu_{dU'}(du) \times du}{\sum \mu_{dU'}(du)}$$

(3) コントローラネットワーク

この研究では、バックプロパゲーションによってTノルム及びTコノルムの自動チューニングを行なうため、ファジィコントローラをネットワークで表現した。フ

ァジィコントローラネットワークを図6.5に示す。ただし、TはTノルムを、SはTコノルムを示し、太い線はメンバーシップ関数を表している。

この研究では図6.5のTおよびSにパラメータ付きTオペレータを採用し、パラメータの自動チューニングを行なう。

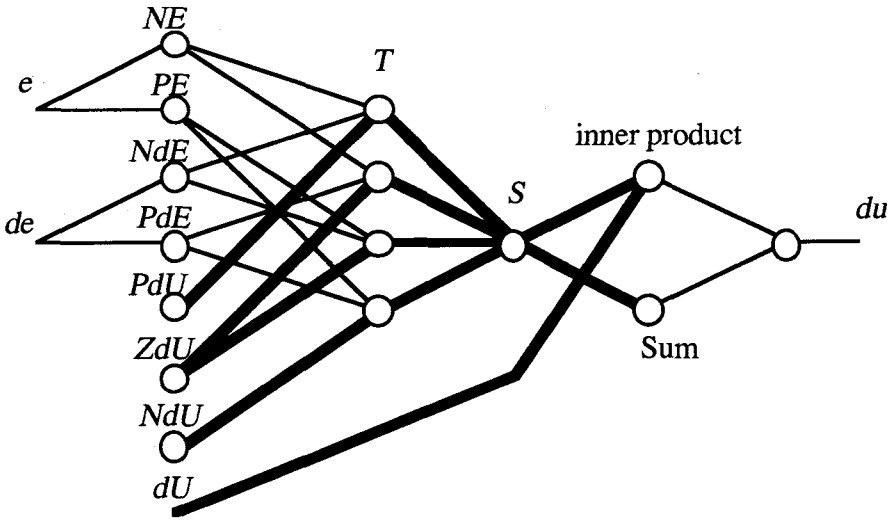

図6.5：ファジィコントローラのネットワーク

(4) 評価関数

ファジィコントローラネットワークの内部パラメータに対して教師無し学習を行なうため、図6.1のシステムの評価関数として誤差E_eを定義する。

$$E_e = \sum_n \frac{1}{2}\left(e(n)^2 + G_U \times du(n)^2\right)$$
$$= \sum_n \frac{1}{2}\left(\left(x(n) - y(n)\right)^2 + G_U \times du(n)^2\right)$$

ここで、G_Uは定数である。この関数は、設定された目標値とプラント出力との差が小さいほど、また、制御に利用するエネルギーが小さいほど評価が高くなる。

(5) プラントネットワーク

誤差はコントローラ入力とプラント出力の間で定義されている。コントローラネットワークの自動チューニングを行なうためには誤差とコントローラ出力の間の関係を知る必要があるが、実際のプラントを対象にした場合、この関係を知ることは非常に困難である。

そこで、プラント部分にもネットワークを利用することを考える。システム構築に際して、まず、ファジィコントローラとは独立に、プラントネットワークの内部パラメータの学習をバックプロパゲーションによって行なう。この場合、対象となる実際のプラントからのデータに基づいた教師有り学習を行なう。学習が収束したらプラントネットワークの内部パラメータを固定し、ファジィコントローラと接続してシステムを構築する。ファジィコントローラの学習の時点では、プラントネットワークの内部パラメータは固定され、プラントをシミュレートするとともに、誤差をファジィコントローラネットワークまで逆伝播する。

これにより、誤差とコントローラ出力の間の関係を知ることができ、パラメータの自動チューニングが可能となる。

(6) パラメータ付きTオペレータ

パラメータによって性質が変化するTオペレータが数多く提案されている[5]。誤差逆伝播によるパラメータの自動チューニングを行なうには、パラメータによる微分可能性が要求される。そこで、今回は、微分可能なものの中で、カバーするTオペレータの多さからDombiのTオペレータを、また、計算量の少なさからHamacharのTオペレータを採用した。

6.3.2 誤差伝播

図6.5の内、誤差伝播に関連する部分を取り出したのが図6.6である。

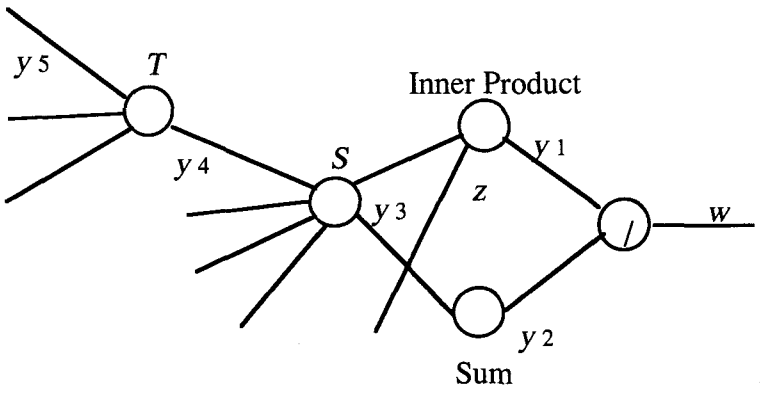

図6.6：対象ネットワーク

各ノードの出力を w および y_1 から y_5 とすると、次のように表現できる。

$$w = \frac{y_1}{y_2}$$
$$y_1 = \sum_z z \cdot y_3$$
$$y_2 = \sum y_3$$
$$y_3 = S(p_S, y_4, \cdots)$$
$$y_4 = T(p_T, y_5, \cdots)$$

各出力はそれぞれ一つ前の出力による微分が計算できる。それらを用いて以下の式が得られる。

$$\frac{dE_e}{dp_S} = \frac{dE_e}{dy}\frac{dy}{du}\frac{du}{dw}\left(\frac{dw}{dy_1}\frac{dy_1}{dy_3} + \frac{dw}{dy_2}\frac{dy_2}{dy_3}\right)\frac{dy_3}{dp_S}$$
$$\frac{dE_e}{dp_T} = \frac{dE_e}{dy}\frac{dy}{du}\frac{du}{dw}\left(\frac{dw}{dy_1}\frac{dy_1}{dy_3} + \frac{dw}{dy_2}\frac{dy_2}{dy_3}\right)\frac{dy_3}{dy_4}\frac{dy_4}{dp_T}$$

ただし、 y の u におよる微分は、プラントネットワークをたどることによって得ることができる。

十分に長い時間間隔になるようにサンプリング回数 N を定める。 N 回のサンプリングを学習の単位とし、1単位の学習が終了する度にパラメータの変更を行ない、新しいパラメータのもとで次の学習を行なう。今回は50回のサンプリングを単位とし、変化がしきい値以下になるまで学習を繰り返した。

j 番目の学習の結果、 $j+1$ 番目のパラメータの値は次の式で計算される。

$$p_S(j+1) = p_S(j) - \frac{dE_e(j)}{dp_S(j)}$$
$$p_T(j+1) = p_T(j) - \frac{dE_e(j)}{dp_T(j)}$$

6.4 実験結果

前述のシステムを用い、プラントとして、一次遅れプラントと二次遅れプラントに対してシミュレーションを行なった。以下ではその結果とその評価を示す。ただし、二次遅れプラントでは制御性自身は異なるものの、一次遅れプラントと同様の評価であるので、対応する図のみを示す。

6.4.1 一次遅れプラントの制御

以下に、一次遅れプラントに対するシミュレーション結果の一部を示す。

(1) T_2 による結果

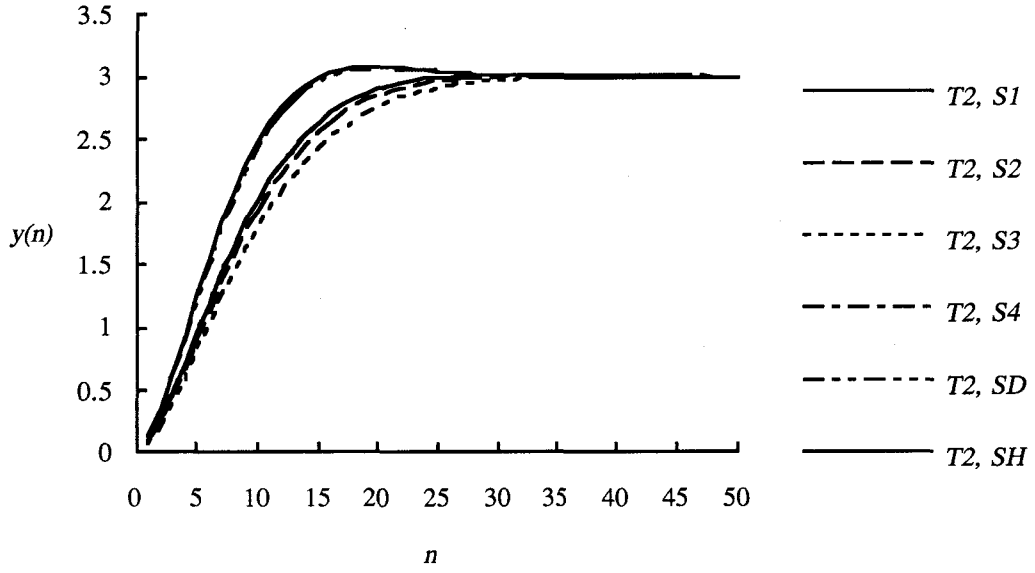

図 6.7 : T ノルムを T_2 に固定した場合

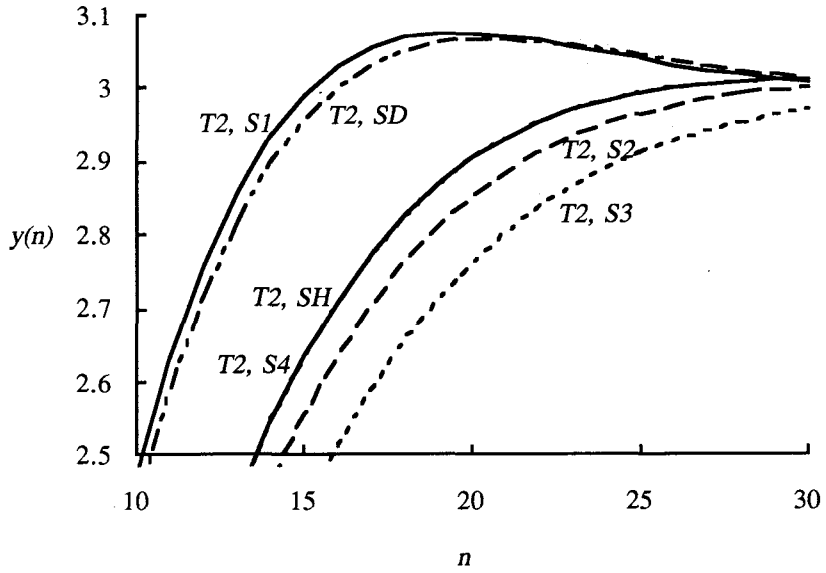

図 6.8 : T_2 に固定した場合の拡大

ファジィコントローラのTノルムを T_2 に固定した場合の結果を図6.7に示す。図6.7の内、立ち上がり部分を拡大した物を図6.8に示す。

ここで、S_D はDombiのTコノルムを表す。チューンイングの結果、パラメータ値は3.06に収束している。S_H はHamacharのTコノルムを表す。チューンイングの結果、パラメータ値は0で、さらに小さな値へ向かっているが、パラメータの定義域が0までであるので、0となった。$S_H(0)=S_4$ であり、実際、S_H と S_4 の図は完全に重なっている。

DombiのTコノルムのパラメータ値は $1<p<\infty$ なので、最適な強さ S は $S_4<S<S_1$ であることを示している。

また、HamacharのTコノルムのパラメータ値は $p<0$ なので、最適な強さ S は $S_4<S$ であることを示している。

結果を総合すると、このシステムに対する適切な強さのTコノルムは、S_4 と S_1 の間で、$S_D(3.06)$ に相当することがわかる。

(2) S_2 による結果

ファジィコントローラのTコノルムを S_2 に固定した場合の結果を図6.9に示す。図6.9の内、T_2、T_D、T_H の立ち上がり部分を拡大し、図6.10に示す。

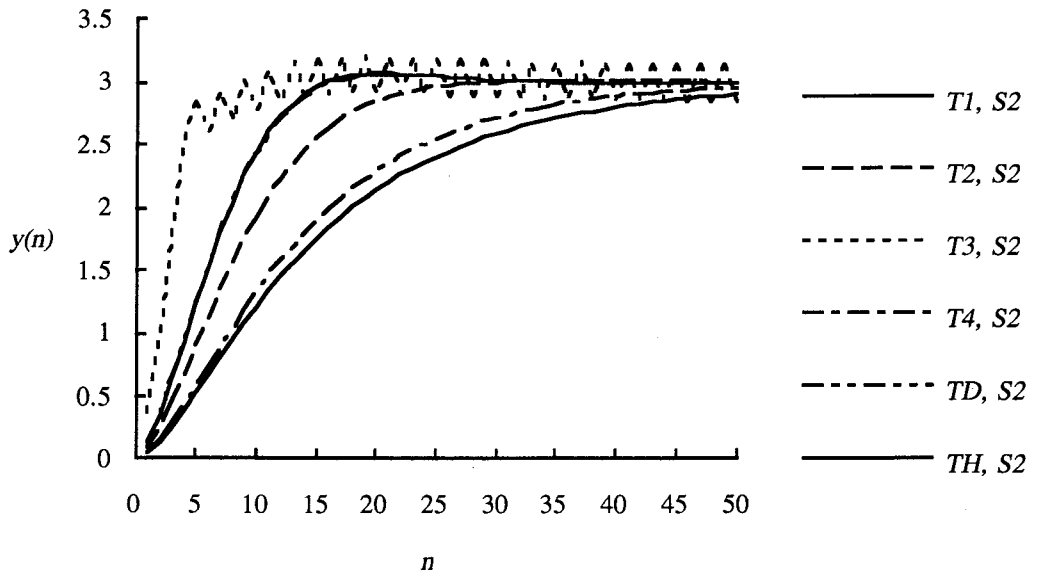

図6.9：Tコノルムを S_2 に固定した場合

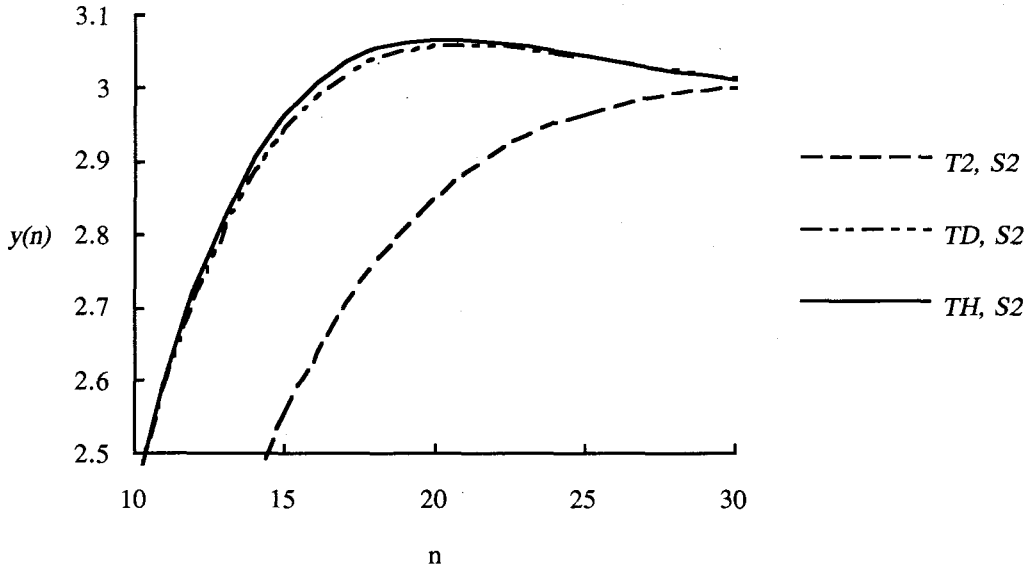

図 6.10 : S_2 に固定した場合の拡大

ここで、T_D は Dombi の T ノルムを表す。チューンイングの結果、パラメータ値は 0.20345 に収束している。また、T_H は Hamachar の T ノルムを表す。チューンイングの結果、パラメータ値は 2.65 に収束している。

Dombi の T ノルムのパラメータ値は $0<p<1$ なので、最適な強さ T は $T_5<T<T_4$ であることを示している。

また、Hamachar の T ノルムのパラメータ値は $1<p<\infty$ なので、最適な強さ T は $T_5<T<T_2$ であることを示している。

Dombi の T ノルムと Hamachar の T ノルムのグラフはほぼ重なっている。これは、Dombi の T ノルムと Hamachar の T ノルムが異なるパラメータ値でありながら、同一の強さに収束していることを示している。

結果を総合すると、このシステムに対する適切な強さの T ノルムは、T_5 と T_2 の間で、$T_D(0.20345)$ および $T_H(2.65)$ に相当する。

(3) T ノルムのチューニング

各 T コノルムに対して Dombi の T ノルムによってチューニングした結果を図 6.11 に示す。各パラメータ値を表 6.1 に示す。

各Tコノルムに対しHamacharのTノルムによってチューニングした結果を図6.12に示す。各パラメータ値を表6.2に示す。

図6.11と図6.12に示したグラフは、各Tコノルムに対して、T_D、T_Hのパラメータ値は異なるが、8本とも全て同一の曲線に収束している。これは、各Tコノルムに対してTノルムをチューニングすることによって、対象プラントに対する最適の制御特性を実現できること示している。

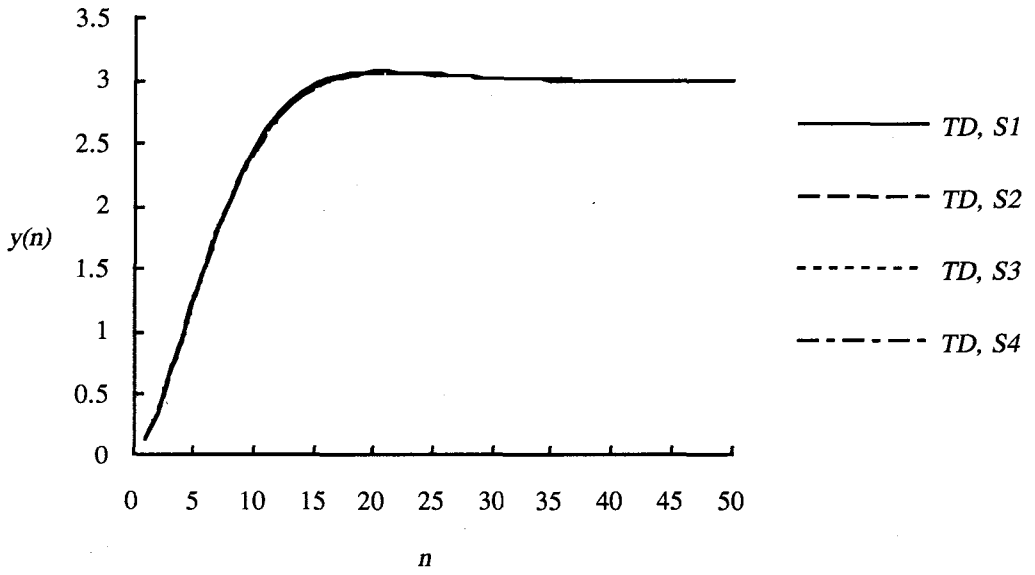

図6.11 : DombiのTノルムによるチューニング

表6.1 : DombiのTノルムのパラメータ値

	T_D	状況
S_1	0.371	収束
S_2	0.20345	収束
S_3	0.20266	収束
S_4	0.2043	収束

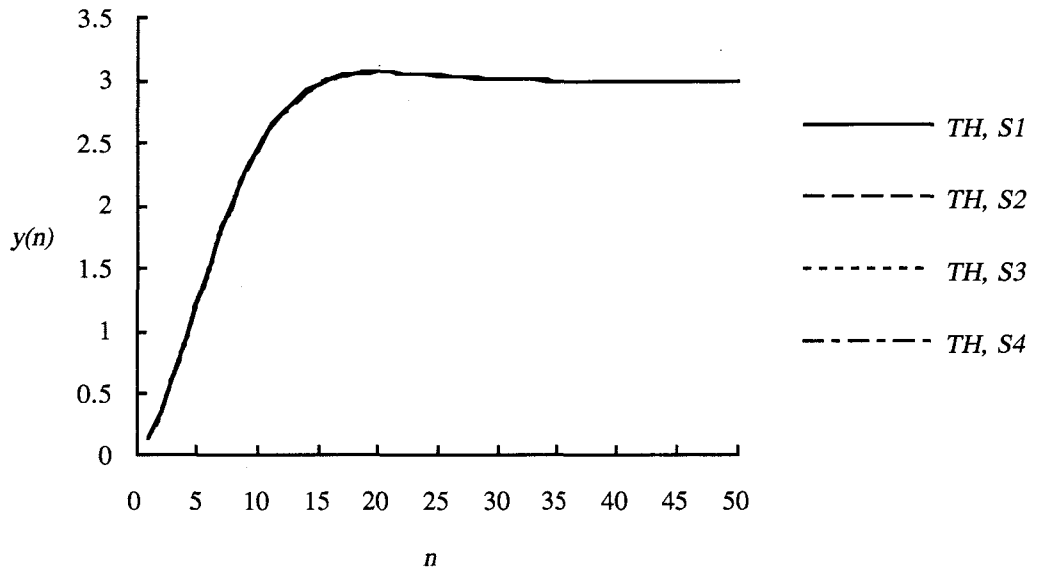

図 6.12 : Hamachar の T ノルムによるチューニング

表 6.2 : Hamachar の T ノルムのパラメータ値

	T_H	状況
S_1	0.941	収束
S_2	2.65	収束
S_3	2.966	収束
S_4	2.437	収束

(4) T コノルムのチューニング

各 T ノルムに対して Dombi の T コノルムによってチューニングした結果を図 6.13 に示す。各パラメータ値を表 6.3 に示す。

各 T ノルムに対して Hamachar の T コノルムによってチューニングした結果を図 6.14 に示す。各パラメータ値を表 6.4 に示す。

図 6.13 において、T_1、T_4 に対して、S_D のパラメータは $p \to \infty$ なので、$S_D \to S_1$ であることを示している。

また、図 6.14 で、T_1、T_4 に対して、S_H のパラメータは $p<0$ なので、$S_H < S_4$ であることを示している。

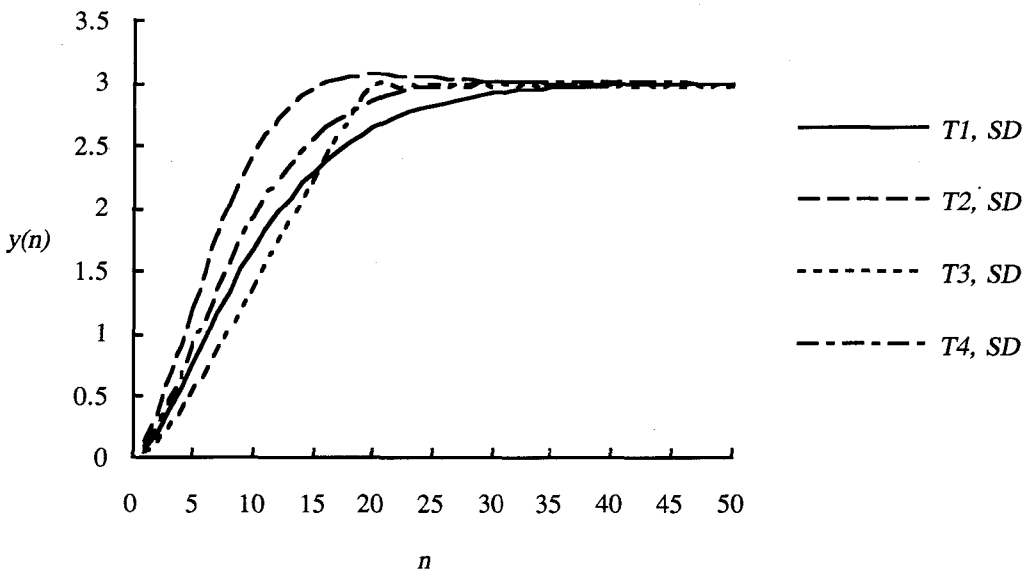

図 6.13 : Dombi の T コノルムによるチューニング

表 6.3 : Dombi の T コノルムのパラメータ値

	S_D	状況
T_1	∞	極値
T_2	3.6	収束
T_3	0.0529	収束
T_4	∞	極値

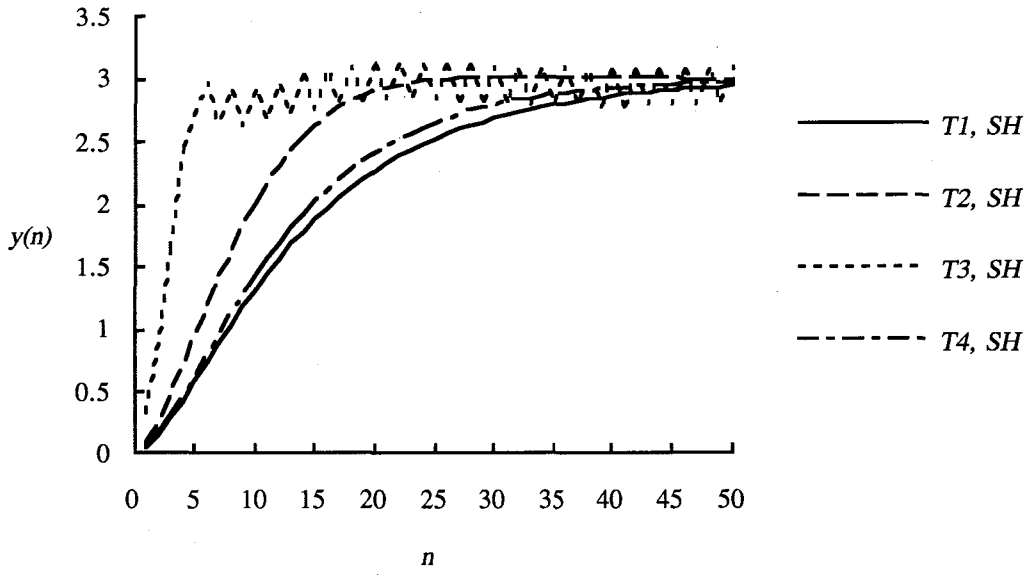

図 6.14 : Hamachar の T コノルムによるチューニング

表 6.4 : Hamachar の T コノルムのパラメータ値

	S_H	状況
T_1	0	極値
T_2	0	極値
T_3	∞	極値
T_4	0	極値

T_1 に対する T コノルムの性質を検討するため、必要なデータを図 6.15 に示す。ここで、極限として $S_D = S_1$ 、 $S_H = S_4$ と考え、その曲線を示した。

T_2, S_D で S_D は収束しており、このシステムの適切な制御特性とみなすことができる。実際、T ノルムのチューニングによって得た適切な制御特性と一致している。図 6.15 には T_D, S_1 を示した。

図 6.15 では、T コノルムが弱い程最適な制御特性に近づいている。従って、T コノルムのチューニングで最適な制御性を得るためには、S_1 よりもさらに弱い演算子が必要である。

T_4 に対するTコノルムの性質を検討するため、必要なデータを図6.16に示す。T_4 に対しても T_1 の場合とまったく同様の考察が可能である。

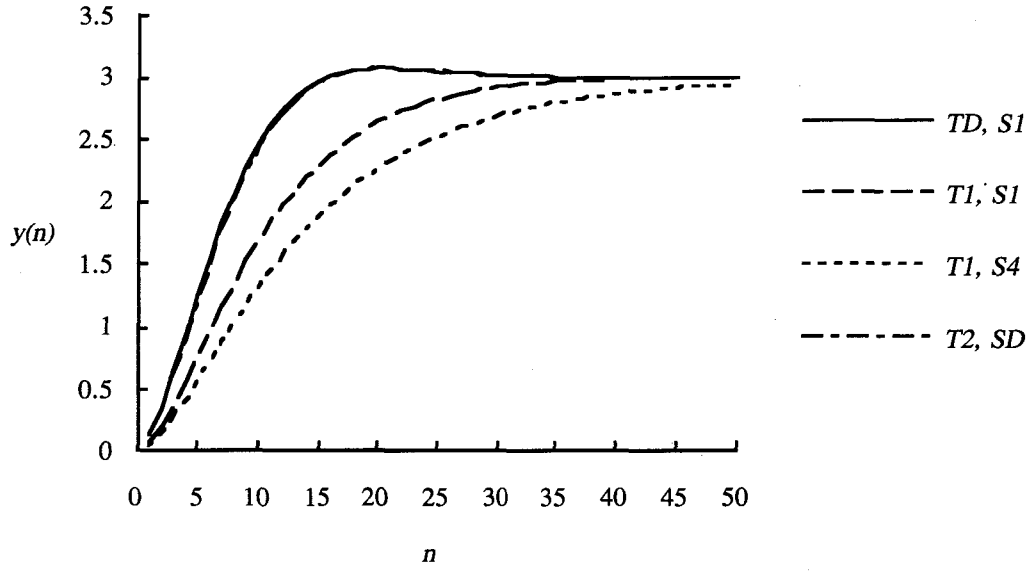

図 6.15 : T_1 に対する検討

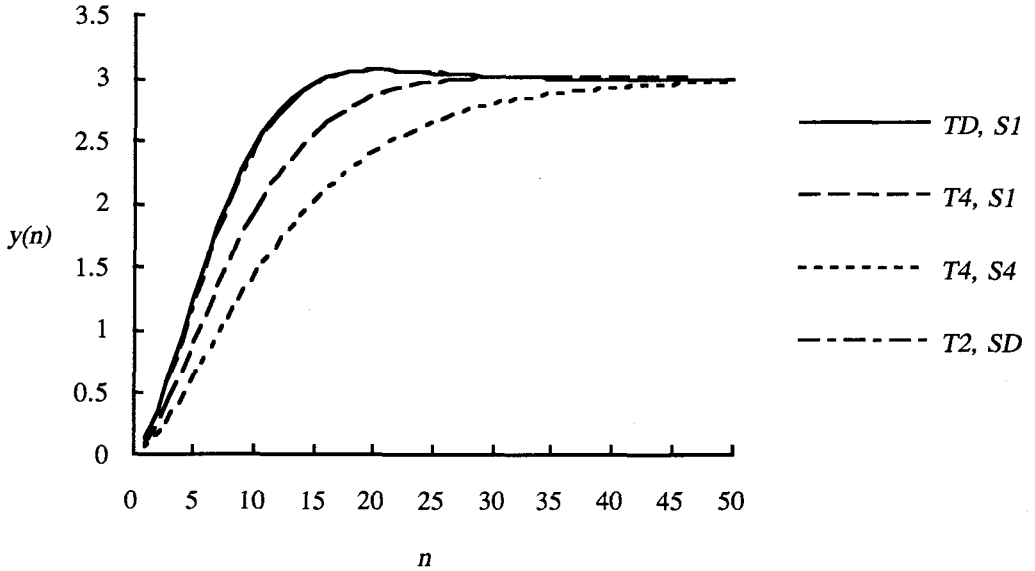

図 6.16 : T_4 に対する検討

(5) T_3 による結果

Tノルムを T_3 に固定した場合を図6.17に示す。

図6.17において、典型的なTコノルムでは振動し、全て同一の曲線になった。また、S_Dのパラメータは$0<p<1$なので、最適の強さSは$S_5<S<S_4$であることを示している。

また、図6.17において、S_Hのパラメータは$p\to\infty$なので、最適の強さSは$S\to S_5$であることを示している。S_Hが収束しなかったのは、パラメータ変化に対する感度が、パラメータ値が大きくなるに従って悪くなるため、計算機の有効数字を超えたものと考えられる。

このように、典型的なTオペレータでは制御できないシステムに対しても、チューニングよって制御することが可能である。

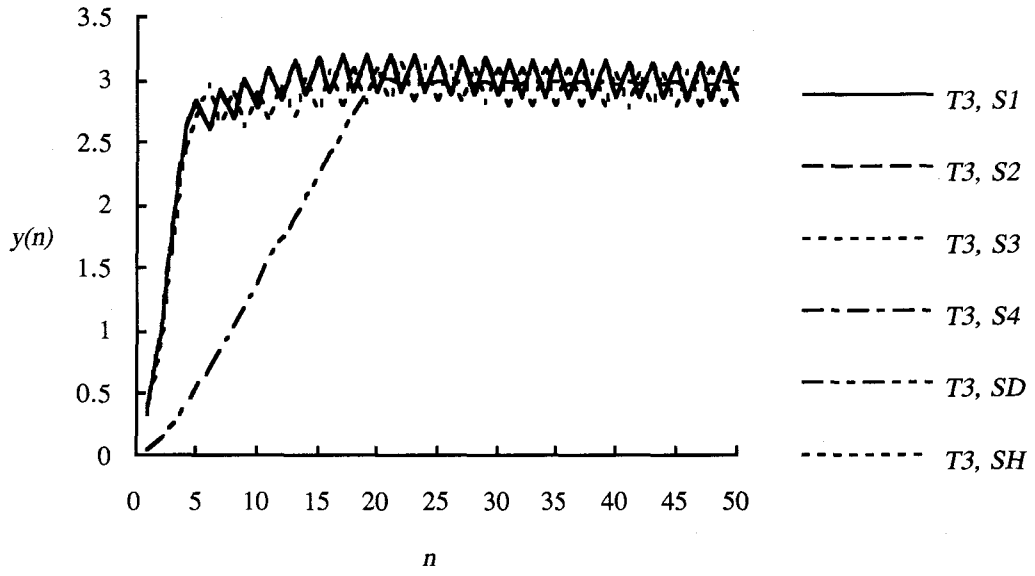

図6.17：TノルムをT_3に固定した場合

6.4.2 二次遅れプラントの制御

前述のシステムでセカンドオーダープラントを対象にシミュレーションを行ない、ファーストオーダープラントの場合と同様の結果をえた。以下に対応する図を列挙する。

(1) T_2 による結果

ファジィコントローラのTノルムを T_2 に固定した場合の結果を図 6.18 に示す。図 6.18 の内、立ち上がり部分を拡大し、図 6.19 に示す。

ここで、S_D のパラメータ値は 3.345 に収束している。また、S_H のパラメータ値は 0 で、さらに小さな値へ向かっている。

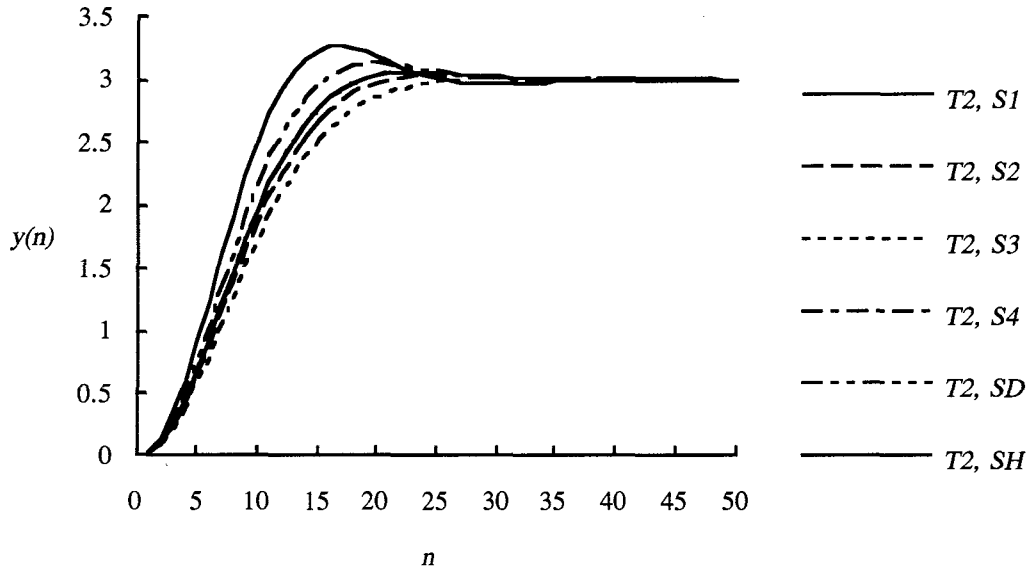

図 6.18 : Tノルムを T_2 に固定した場合

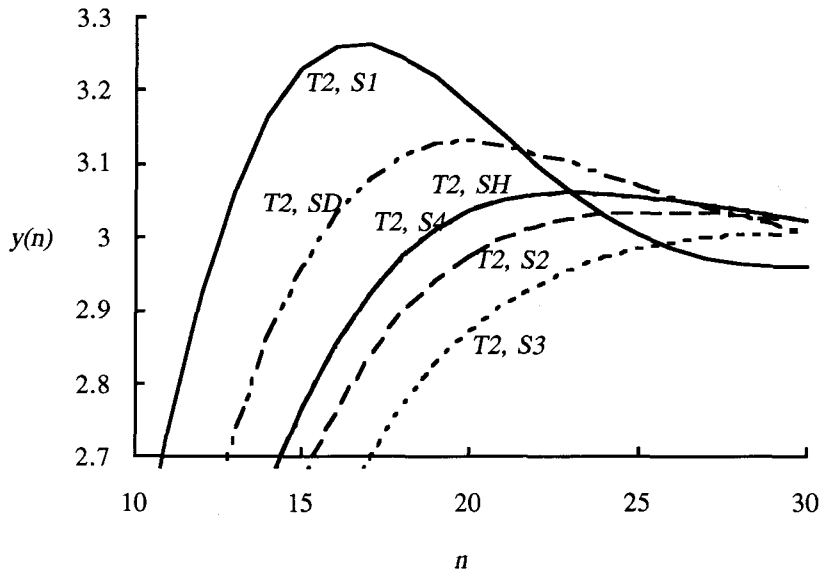

図 6.19 : T_2 に固定した場合の拡大

(2) S_2 による結果

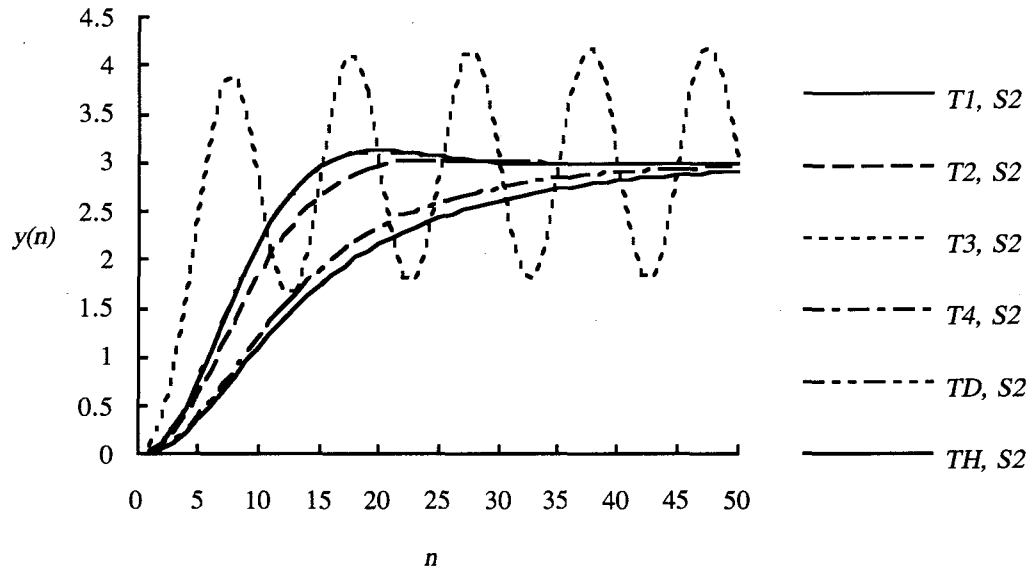

図 6.20 : T コノルムを S_2 に固定した場合

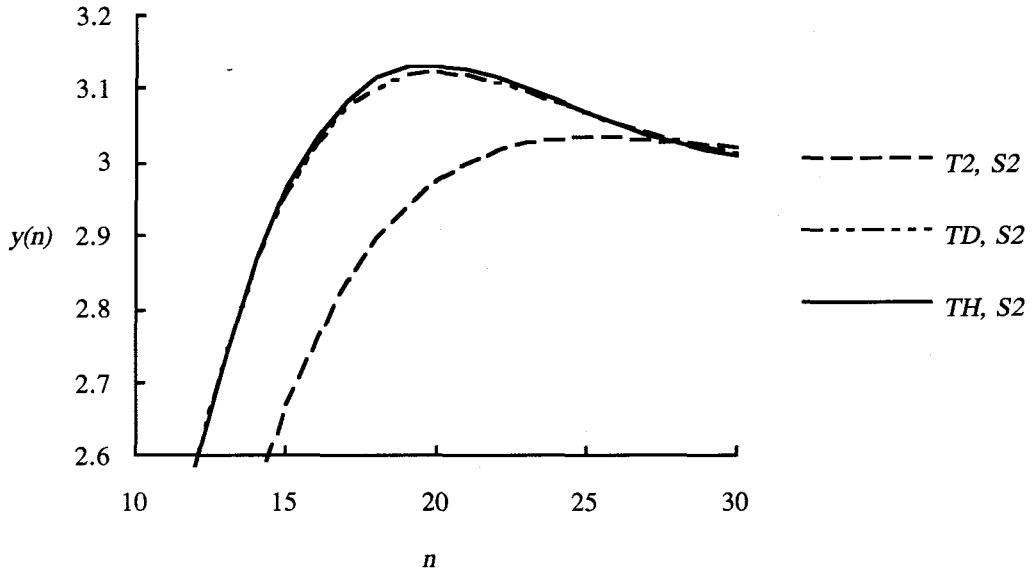

図 6.21 : S_2 に固定した場合の拡大

ファジィコントローラの T コノルムを S_2 に固定した場合の結果を図 6.20 に示す。図 6.20 の内、 T_2 、 T_D 、 T_H の立ち上がり部分を拡大した物を図 6.21 に示す。

ここで、T_D のパラメータ値は 0.2526 に収束している。また、T_D のパラメータ値は 1.7 に収束している。

(3) T ノルムのチューニング

各 T コノルムに対して Dombi の T ノルムによってチューニングした結果を図 6.22 に示す。各パラメータ値を表 6.5 に示す。

各 T コノルムに対して Hamachar の T ノルムによってチューニングした結果を図 6.23 に示す。各パラメータ値を表 6.6 に示す。

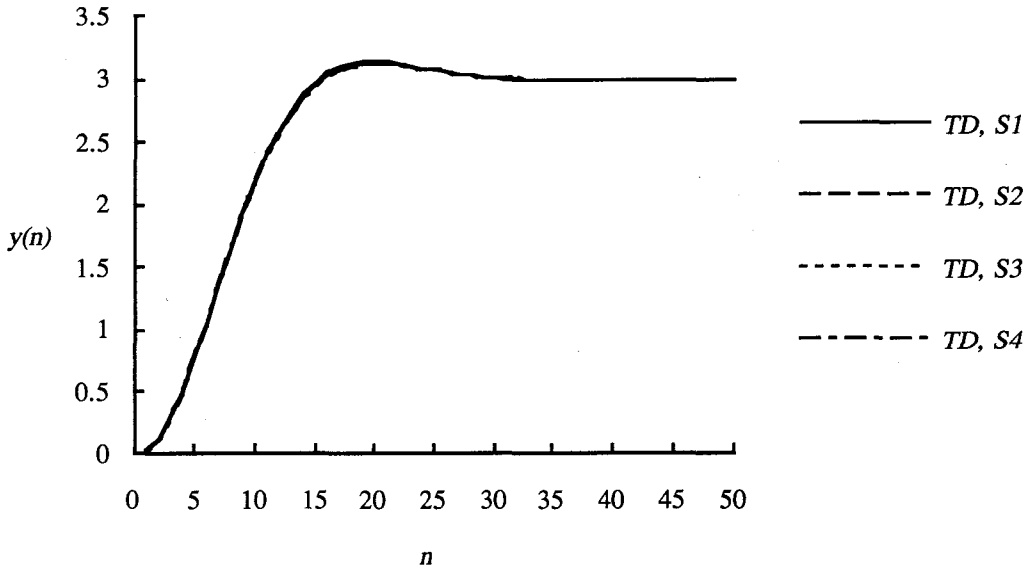

図 6.22 : Dombi の T ノルムによるチューニング

表 6.5 : Dombi の T ノルムのパラメータ値

	T_D	状況
S_1	0.547	収束
S_2	0.2526	収束
S_3	0.25	収束
S_4	0.2556	収束

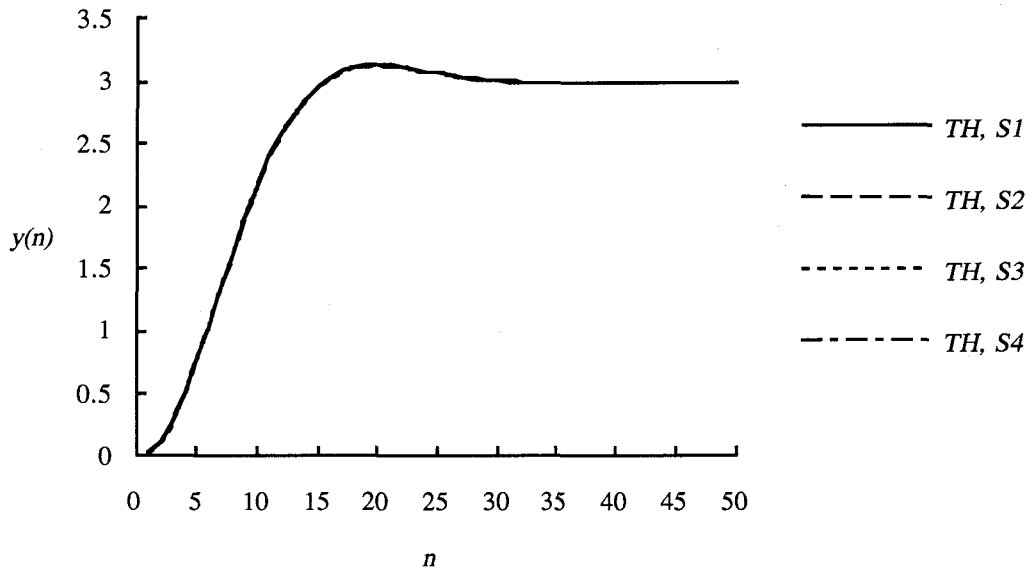

図 6.23 : Hamachar の T ノルムによるチューニング

表 6.6 : Hamachar の T ノルムのパラメータ値

	T_H	状況
S_1	0.3841	収束
S_2	1.7	収束
S_3	1.983	収束
S_4	1.477	収束

(4) T コノルムのチューニング

各 T ノルムに対して Dombi の T コノルムによってチューニングした結果を図 6.24 に示す。各パラメータ値を表 6.7 に示す。

各 T ノルムに対して Hamachar の T コノルムによってチューニングした結果を図 6.25 に示す。各パラメータ値を表 6.8 に示す。

T_1 に対する T コノルムの性質を検討するため、必要なデータを図 6.26 に示す。

T_4 に対する T コノルムの性質を検討するため、必要なデータを図 6.27 に示す。

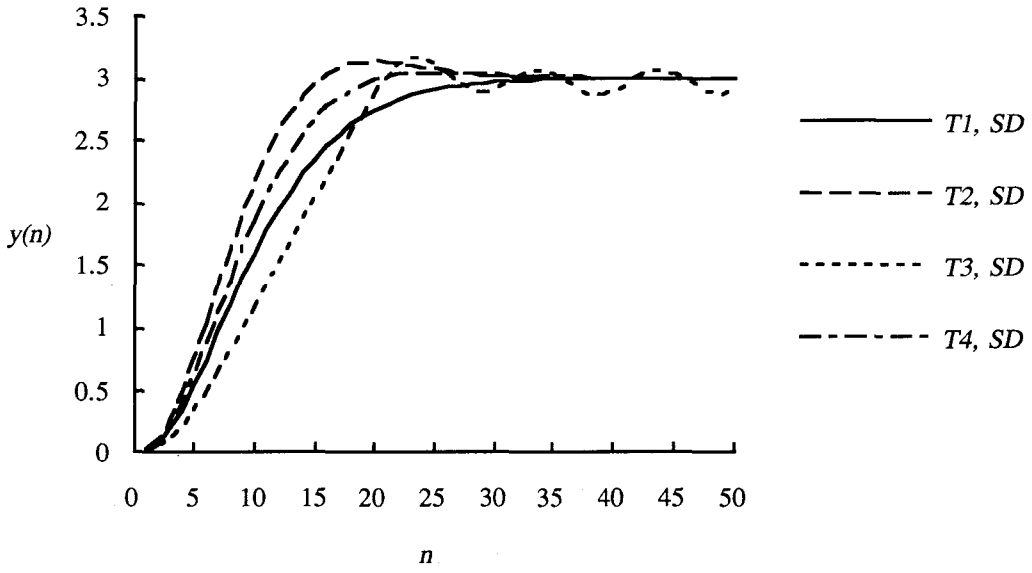

図 6.24 : Dombi の T コノルムによるチューニング

表 6.7 : Dombi の T コノルムのパラメータ値

	S_D	状況
T_1	∞	極値
T_2	1.3345	収束
T_3	0.05286	収束
T_4	∞	極値

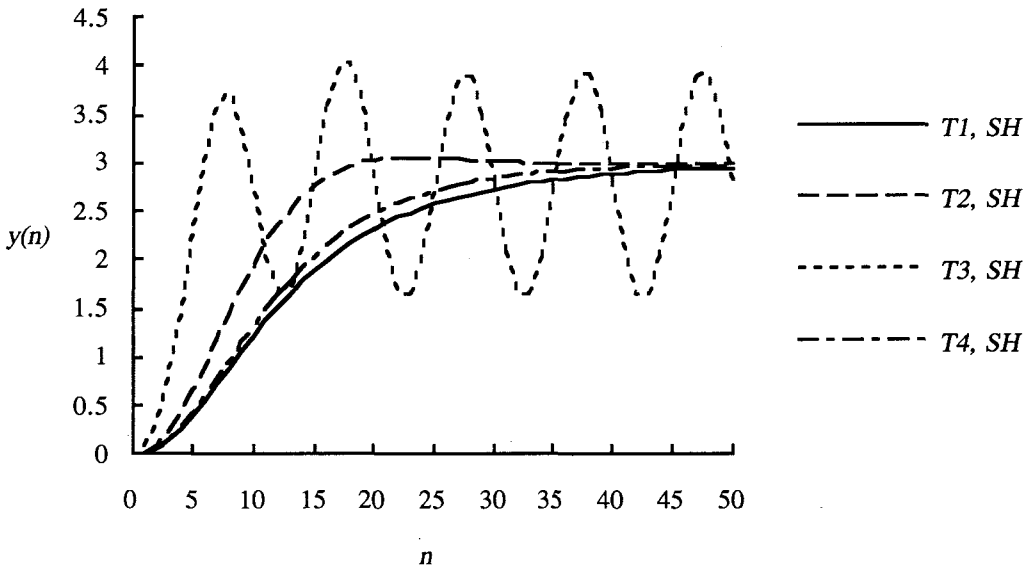

図 6.25 : Hamachar の T コノルムによるチューニング

表 6.8 : Hamachar の T コノルムのパラメータ値

	S_H	状況
T_1	0	極値
T_2	0	極値
T_3	∞	極値
T_4	0	極値

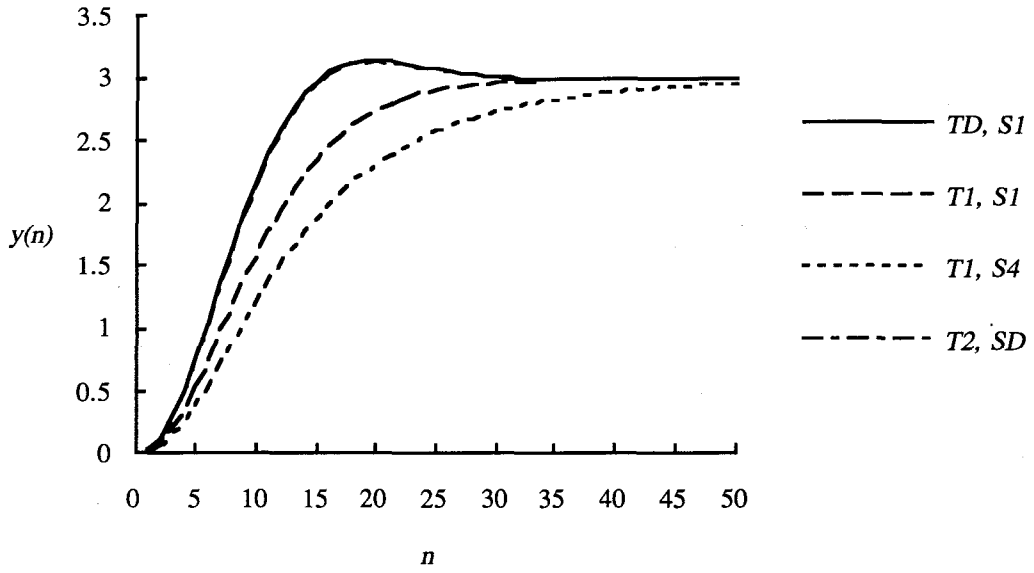

図 6.26 : T_1 に対する検討

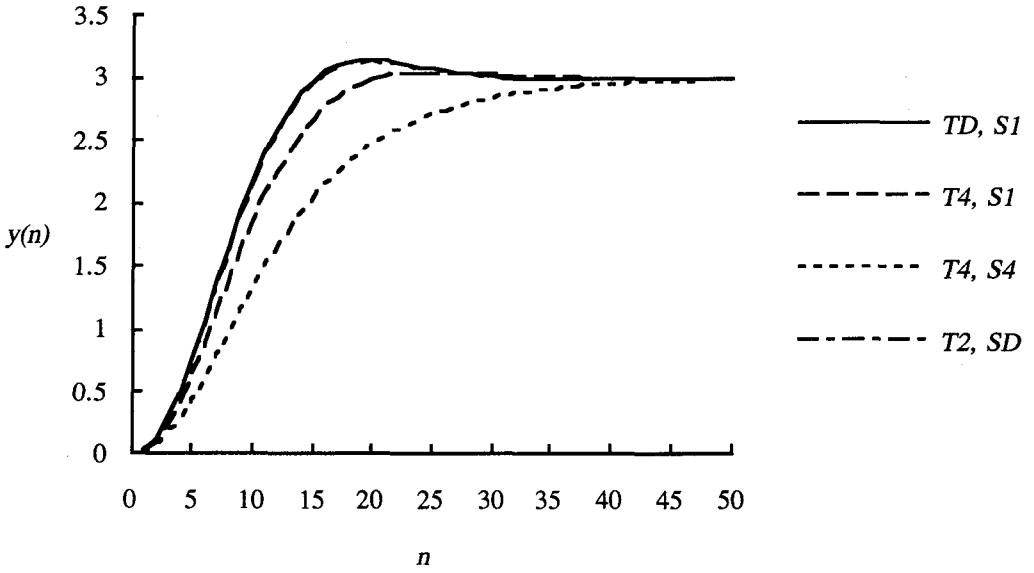

図 6.27 : T_4 に対する検討

(5) T_3 による結果

T_3 に固定した場合を図 6.28 に示す。

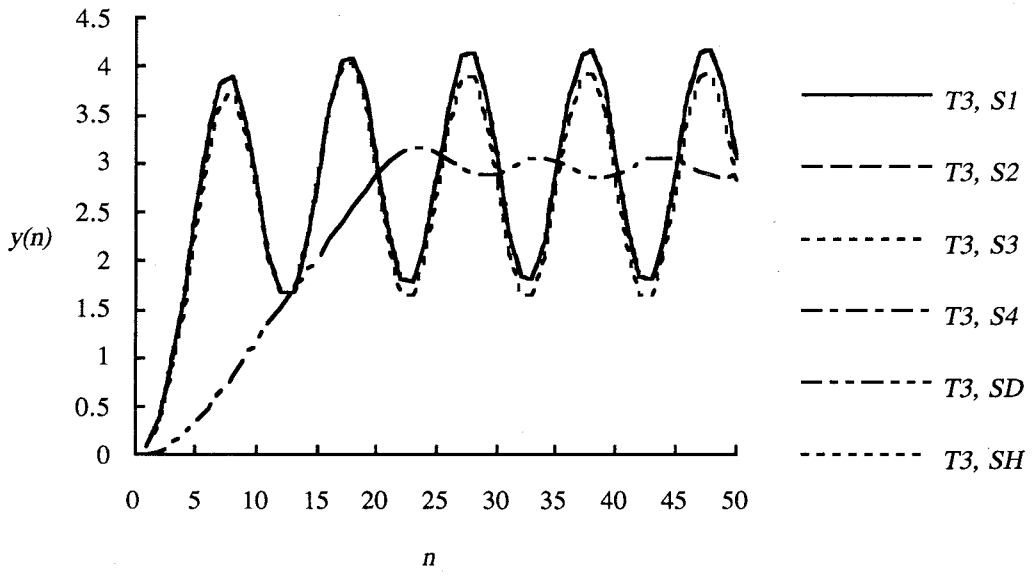

図 6.28 : T ノルムを T_3 に固定した場合

6.4.3 まとめ

上記の通り、一次遅れプラントと二次遅れプラントに対して、典型的な T オペレータ、Dombi の T オペレータ、Hamachar の T オペレータを用いてファジィコントローラのシミュレーションを行なった。以下に実験から得られた結果をまとめる。

T ノルムのチューニングでは、典型的な T コノルムの全てに対して収束し、最適な強さとして典型的な T ノルムの中間的な強さを得た。これは、T ノルムの最適な強さは典型的な T ノルムの中間にある場合があること示している。おそらく、ほとんどの場合は中間にあるものと思われる。

T コノルムのチューニングでは、極値をとる場合が多く、T_1 と T_4 に対しては、最適な強さは T コノルムよりさらに弱い領域にあることがわかった。また、T_3 に対しては、典型的な T コノルムでは制御できないが、チューニングを行なうことによって、ある程度の制御性ができた。これは、現在 T コノルムを用いている関数は、実際には T コノルムから中間演算子を含む領域をカバーする必要があること、また、T コノルムをチューニングすることによって、今まで制御できなかった対象を制御できる可能性があることを示している。

6.5 考察

ファジィルールを用いた制御で制御特性が悪い場合、次の原因が考えられる。

1) 専門家の知識自身が不完全であったため

2) 専門家の知識を十分表現できていなかったため

普通は、両方を含むと考えられる。1) にはメンバーシップ関数やTオペレータのチューニングによって対応できるが、2) の場合、どのような知識表現が有効であるか研究する必要がある。そこで、ファジィプロダクションルールのチューニングを、ルール利用の観点からは、最適な制御性の実現であり、知識表現の観点からは、自然言語の意味の定量化であると考える。以下にそれぞれの観点から考察を行なう。

6.5.1 制御性の向上

専門家の知識自身が不完全である場合、専門家がメンバーシップ関数をチューニングすることになる。この手法の拡張としてメンバーシップ関数の自動チューニングの研究が行なわれている。しかし、自動チューニングを行なった知識に対しては、専門家によるルールの更新など再チューニングが非常に困難であったり、各条件間に複雑な関係がある場合には、メンバーシップ関数のチューニングだけでは対応できない、などの問題点がある。

ファジィルールでは、メンバーシップ関数の他にも、各条件の結合演算子、インプリケーション演算子、コンビネーション演算子の拡張が考えられる。この研究ではまず、各条件の結合演算子の自動チューニングを行なった。結合演算子を用いてチューニングすることの主な利点として、次の3つがあげられる。

- メンバーシップ関数が変化しないので、自動チューニング後も専門家によるルールの追加、変更、削除が容易にできる。
- メンバーシップ関数のチューニングでは対応できない対象に対してチューニングできる。

6.5.2 意味の定量化

結合演算子の自動チューニングでは、専門家の用いている結合演算子の強さを同定することができる。さらに、図6.5のコントローラネットワークを見ると、誤差は、Tコノルム、Tノルムを経て、メンバーシップ関数まで逆伝播できるため、このシステムを拡張することによって、メンバーシップ関数、結合演算子、インプリケーション演算子、コンビネーション演算子の自動チューニングが可能である。

自動チューニングによって得られた結果により、このルールにおける結合演算子の強さはこの典型的Tオペレータに近い、このルールのインプリケーション演算子はほとんど変化していないなど、ルールのどの部分が人間にとって不自然になっているかを知ることができる。

さらに、結合演算子として加藤[54, 63]の相乗演算子を利用すると、自動チューニングの結果により、この「and」は強い相乗性を持っている、この「or」は弱い相乗性を持っているなど、推論方法に関係する自然言語の意味の定量化を行なうことができるものと思われる。

6.6 おわりに

プラントネットワークとコントローラネットワークからなるシステムを用いて、ファジィコントローラの結合演算子に、パラメータ付きTオペレータを採用し、定義した誤差をもとに、バックプロパゲーションを利用してこのパラメータの自動チューニングを行なう方法を提案した。

色々な種類のプラントに対するシミュレーションを行ない、提案したシステムを用いて自動チューニングが可能であることを示した。

一般に、各制御対象に対するTオペレータの最適な強さは典型的なTオペレータの強さの中間にあり、パラメータ付きTオペレータを利用することによって適切な強さを得ることができた。

第7章　結論

この論文は、ファジィシェルが抱える問題点の解決と知識獲得の改善を目的に、ファジィエキスパートシステム開発支援ツールLIFE FEShellの研究開発と、ファジィプロダクションルールの結合演算子の最急降下法による自動チューニングの研究を通して、エキスパートシステムにおけるあいまい性の取扱いに関する考察をまとめた。

7.1 各章のまとめ

以下に各章の概要と成果を要約する。

第1章では、論文の背景、目的、概要について述べ、LIFE FEShellの概要を示した。

第2章では、あいまい性を取扱うための理論として、ファジィ論理、Tノルムと Tコノルム、ファジィ集合論、ファジィ測度とファジィ積分、可能性理論を説明した。

第3章では、プロダクションシステムに現れるあいまい性について整理した上で、LIFE FEShell FPSで採用したあいまい性の取扱い機能について述べた。このシステムは、ルール条件部のファジィな条件、ルールの条件部と結論部の間の関係のあいまい性、データの不確実性と不完全性を取扱うことができる。さらに、ルール条件部の各条件の間の関係として、and、orに加えて、ファジィ測度で定義される複雑な関係も取扱うことができる。

第4章では、フレーム表現に関する基本的な考え方を説明し、フレームシステムで扱う値および関係に関するあいまい性について分析した上で、LIFE FEShell FFSにおいて採用したあいまい性の取扱い機能について述べた。このシステムでは、値

および関係のあいまい性をタームやプレディケートといった形で言語的に表現することが可能である。さらに、タームやプレディケートの宣言にあたっては、同一の言葉に対して複数の宣言を許し、対象に応じて適切な宣言を採用するよう、フレーム間の関係を利用してあいまい性の宣言の継承、階層的管理、推論などの枠組みを提供した。

第5章では、従来のシェルとファジィシェルに関して開発環境の観点から分析を行ない、LIFE FEShell 開発環境の設計目標を示した。次に開発環境が取扱うべきオブジェクトとその編集の実際について述べ、その詳細を示した。提案した開発環境を利用することによって、異なった形式の知識の入力や編集にあたって、統一的なインタフェースが有効であることが判明した。また、可能性分布の入力・編集をグラフィカルなインタフェースで行なうことによって、入力誤りを減らし、可能性分布の直感的な理解を助けることができ、チューニングの効率が向上することが判明した。プロトコルによって推論エンジン部分とユーザインタフェース部分を切り離し、有効な実装方法を提案した。これにより、大きなシステムの必要な部分に対して、それぞれ独立・平行に作業を行なうことができるため、全体としての開発の負担を極小に押さえることができた。

第6章では、知識獲得の改善をはかるため、ファジィルールの結合演算子とコントローラの制御性について述べ、プラントネットワークとコントローラネットワークからなるシステムを用い、ファジィコントローラの結合演算子にパラメータ付きTオペレータを採用し、バックプロパゲーションを用いてパラメータの自動チューニングを行なう方法を提案した。計算機シミュレーションを行ない、典型的なTオペレータの強さの中間にある最適な強さのTオペレータを得ることができた。

7.2 今後の課題

LIFE FEShell FPS は広範囲な悪構造問題に適用可能なシステムであるが、その機能実現に重点を置いたため、処理速度に問題が残っている。今後、高速なパターンマッチングのアルゴリズムに関する研究が必要である。

LIFE FEShell FFS は値のあいまい性を言語的に多義的に宣言することを可能にしているが、多重継承におけるあいまい性の取扱い対する考察が不十分であると考える。

開発環境に関する課題としては、あいまいな知識の入力に適したインタフェースを提供しているが、開発環境の構築自身にもファジィ論理を応用し、利用者に適応するユーザインタフェースを提供することが考えられる。

ファジィルールの結合演算子の自動チューニングで最適な強さのTオペレータを得ることができたが、パラメータ値が収束せず平均演算子の方へ、関数の定義域を超えてしまう場合があったことから、結合演算子としてTノルム、Tコノルムに加えて平均演算子の領域までカバーする関数を用いた研究が必要と思われる。また、チューニングにおいて、パラメータ付きTオペレータの性質とともにその導関数が学習速度などに大きく影響することがわかったため、Tオペレータとともにその導関数の研究が必要である。今後、メンバーシップ関数と結合演算子のチューニングの解釈、インプリケーションまで含めたファジィプロダクションルールのチューニング等に関して検討を行なうとともに、推論方法に関係する自然言語の意味の定量化を行なう。

謝辞

この論文をまとめるにあたり、ご多忙中にも関わらず種々の有益なご教授を賜わりました大阪大学基礎工学部システム工学科 田村坦之教授、井口征士教授、谷内田正彦教授ならびに馬野元秀助教授に深く謝意を表します。

この研究を遂行するにあたり、多大なご指導と終始懇切なご鞭撻を賜わった国際ファジィ工学研究所 寺野寿郎所長、中村和男副所長ならびに Anca L. Ralescu 所長補佐に心から感謝致します。

平素から、有益なご教授とご支援をいただいているシャープ株式会社技術本部 河田亨副本部長、情報技術研究所 大崎幹雄所長ならびに千葉徹部長に心から感謝致します。

LIFE FEShell の研究を遂行するにあたり、ご指導、ご助言をいただいた国際ファジィ工学研究所 深海悟博士 (現 NTT データ通信株式会社) ならびに小山宏主任研究員 (現花王株式会社) に深く感謝致します。ファジィルールのチューニングの研究を遂行するにあたり、ご指導、ご助言をいただいた国際ファジィ工学研究所 田野俊一主任研究員、加藤康記主任研究員 (現株式会社インテック) ならびに Therry Arnould 主任研究員に深く感謝致します。また、この遂行にご協力いただいた国際ファジィ工学研究所 村上譲司主任研究員 (現横河電機株式会社) ならびに村野健一主任研究員 (現日本鋼管株式会社) に御礼申し上げます。

常に励ましのお言葉をいただいた国際ファジィ工学研究所 中屋敷正人専務理事ならびに藤森聿子常務理事に心から感謝致します。

最後に、種々の面でご協力、ご支援いただいた国際ファジィ工学研究所ならびにシャープ株式会社技術本部情報技術研究所の皆様に厚く御礼申し上げます。

参考文献

[1] 田中幸吉 : 知識工学, 朝倉書店 (1984).

[2] 長尾真, 石田晴久, 稲垣康善, 田中英彦, 辻井潤一, 所真理雄, 田中育男, 米澤明憲 : 情報科学辞典, 岩波書店 (1990).

[3] L.A. Zadeh : "Calculus of Fuzzy Restrictions," in *Fuzzy Sets and their Applications to Cognitive and Decision Processes*, Academic Press, pp. 1-39 (1975).

[4] L.A. Zadeh : "The Concept of a Linguistic Variable and Its Application to Approximate Reasoning - I," *Information Science*, Vol. 8, pp. 199-249 (1975).

[5] 水本雅晴 : "ファジィ集合とファジィ推論," *第3回 ファジィシステムシンポジウム*, pp. 37-48 (1987).

[6] 水本雅晴 : "最近のファジー理論," *情報処理*, Vol. 29, No. 1, pp. 11-22 (1988).

[7] D. Dubois and H. Prade : "An Introduction to Possibilistic and Fuzzy Logics," in *Non-Standard Logics for Automated Reasoning*, Academic Press, pp. 287-326 (1988).

[8] D. Dubois and H. Prade : Possibility Theory, Plenum Press (1988).

[9] L.A. Zadeh : "Fuzzy Sets as a Basis for a Theory of Possibility," *Fuzzy Sets and Systems*, Vol. 1, No. 1, pp. 3-28 (1978).

[10] J.A. Freeman and D.M. Skapura : Neural Networks Algorithms, Applications, and Programming Techniques, Addison-Wesley Publishing Company (1991).

[11] B.J.A. Krose and P.P.v.d. Smagt : An Introduction to Neural Networks, University of Amsterdam (1993).

[12] D.E. Goldberg : Genetic Algorithm in Search, Optimization and Machine Learning, Addison-Wesley Publishing Company (1989).

[13] J.H. Holland : Adaptation in Natural and Artificial Systems, University of Michigan Press (1975).

[14] 北野宏明 : 遺伝的アルゴリズム, 産業図書 (1993).

[15] J.F. Baldwin and S.Q. Zhou : "A Fuzzy Relational Inference Language," *Fuzzy Sets and Systems*, Vol. 14, pp. 155-174 (1984).

[16] J. Bowen and J. Kang : "A Fuzzy Multi-paradigm Language," *Fuzzy Sets and Systems*, Vol. 34, pp. 263-291 (1990).

[17] J. Buckley : "Managing Uncertainty in Fuzzy Expert System," *Int. J. Man-Machine Studies*, Vol. 29, pp. 129-148 (1988).

[18] J. Buckley, W. Siler, and D. Tucker : "FLOPS, A Fuzzy Expert System: Applications and Perspectives," in *Fuzzy Logic in Knowledge Engineering*, Verlag TUV Rhinland, pp. 256-274 (1986).

[19] H. Farreny and H. Prade : "Default and Inexact Reasoning with Possibility Degrees," *IEEE Trans. System, Man, Cybern.*, Vol. SMC-16, No. 2, pp. 270-276 (1986).

[20] H. Farreny, H. Prade, and E. Wyss : "Approximate Reasoning in a Rule-Based Expert System Using Possibility Theory : A Case Study," in *Information Processing '86*, pp. 407-413 (1986).

[21] K.S. Leung and W. Lam : "Fuzzy Concepts in Expert Systems," *IEEE Computer Magazine*, Vol. 21, No. 9, pp. 43-56 (1988).

[22] K.S. Leung and W. Lam : "Fuzzy Expert System Shell Using Both Exact and Inexact Reasoning," *Journal of Automated Reasoning*, Vol. 5, pp. 207-233 (1989).

[23] K.S. Leung, M.H. Wong, and W. Lam : "Fuzzy Expert Database System," *Data & Knowledge Engineering*, Vol. 4, pp. 287-304 (1989).

[24] H. Prade : "A Computational Approach to Approximate and Plausible Reasoning with Application to Expert System," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 7, No. 3, pp. 260-283 (1985).

[25] W. Siler : "FLOPS: A Fuzzy Expert System Shell," *2nd IFSA Congress*, pp. 848-850 (1987).

[26] 古橋武 : "ニューラルネットワークによるファジィルールの表現," *日本ファジィ学会誌*, Vol. 5, No. 2, pp. 30-43 (1993).

[27] 堀川慎一, 染谷隆子, 船橋誠寿 : "ファジィニューラルネッワークの構成法 (III)," *第7回 ファジィシステムシンポジウム*, F5-2 (1991).

[28] H. Koyama, T. Miyoshi, S. Fukami, and M. Umano : "Management of Uncertainty in LIFE FEShell Fuzzy Production System," *Forth IFSA World Congress*, Brussels, Belgium, pp. 121-124 (1991).

[29] T. Miyoshi, S. Fukami, H. Koyama, and M. Umano : "Management of Uncertainty in LIFE FEShell Fuzzy Frame System," *Expert Systems With Applications*, Vol. 5, pp. 359-368 (1992).

[30] T. Miyoshi, H. Koyama, S. Fukami, and T. Takagi : "Overview of LIFE Fuzzy Expert System Shell," *Sino-Japan Joint Meeting on Fuzzy Sets and Systems*, Beijing, China, C2-10 (1990).

[31] T. Miyoshi, H. Koyama, S. Fukami, T. Takagi, and M. Umano : "LIFE Fuzzy Expert System Shell," *The First International Symposium on Uncertainty Modeling and Analysis (ISUMA'90)*, Maryland, USA, pp. 196-201 (1990).

[32] T. Miyoshi, H. Koyama, S. Fukami, T. Takagi, and M. Umano : "LIFE Fuzzy Expert System Shell," in *Analysis and Management of Uncertainty: Theory and Applications*, North-Holland, pp. 153-163 (1992).

[33] T. Miyoshi, H. Koyama, S. Fukami, S. Tano, and M. Umano : "Fuzzy Expert System Shell LIFE FEShell - Working Environment -," *Second International Symposium on Uncertainty Modeling and Analysis (ISUMA'93)*, Maryland, USA, pp. 153-160 (1993).

[34] T. Miyoshi, H. Koyama, S. Fukami, and M. Umano : "Fuzzy Frame System in LIFE FEShell," *Forth IFSA World Congress*, Brussels, Belgium, pp. 141-144 (1991).

[35] T. Miyoshi, H. Koyama, S. Fukami, and M. Umano : "Management of Uncertainty in LIFE FEShell Fuzzy Frame System," *The World Congress on Expert Systems*, Florida, USA, pp. 631-638 (1991).

[36] T. Miyoshi, H. Koyama, S. Fukami, and M. Umano : "Overview of LIFE FEShell," *Joint Hungarian-Japanese Symposium on Fuzzy System and Applications*, Budapest, Hungary, pp. 127-130 (1991).

[37] 三好力, 深海悟, 小山宏, 高木友博, 馬野元秀 : "LIFE FEShell におけるオブジェクトエディタ," *第6回 ファジィシステムシンポジウム*, FB3-3, pp. 267-270 (1990).

[38] 三好力, 深海悟, 馬野元秀 : "UNIX アプリケーションの機能拡張とプロセス間通信に関する一考察," *情報処理学会 論文誌*, Vol. 32, pp. 1030-1033 (1991).

[39] 小山宏, 三好力, 深海悟, 高木友博, 馬野元秀 : "LIFE FEShell におけるファジィプロダクションシステム," *第6回 ファジィシステムシンポジウム*, FB3-2, pp. 263-266 (1990).

[40] 小山宏, 深海悟, 三好力, 馬野元秀 : "LIFE FEShell ファジィプロダクションシステムにおけるあいまいさの取り扱いとその実現," *第7回 ファジィシステムシンポジウム*, F3-3, pp. 651-654 (1991).

[41] 深海悟, 三好力, 小山宏, 高木友博, 馬野元秀 : "ファジィエキスパートシステム構築支援ツール LIFE FEShell," *第6回 ファジィシステムシンポジウム*, FB3-1, pp. 259-262 (1990).

[42] T. Miyoshi, S. Tano, Y. Kato, and T. Arnould : "Operator Tuning in Fuzzy Production Rules Using Neural Networks," *Second IEEE International Conference on Fuzzy Systems*, San Francisco, California, USA, pp. 641-646 (1993).

[43] 三好力, 田野俊一, 加藤康記, T. Arnould : "ファジィプロダクションルールにおける結合子のニューラルネットによるチューニング," *情報処理学会 第45回 全国大会*, Vol. 2, 1H-3, pp. 77-78 (1992).

[44] 三好力, 田野俊一, 加藤康記, T. Arnould : "ファジィプロダクションルールの結合子による自動チューニング," *第9回 ファジィシステンシポジウム*, FA5-2, pp. 601-604 (1993).

[45] M. Umano : "Fuzzy-Set Manipulation System in LISP," *Second IFSA World Congress*, pp. 840-843 (1987).

[46] T. Yagyu, H. Yuize, M. Yoneda, M. Grabish, and S. Fukami : "Foreign Exchange Trade Support Expert System," *Forth IFSA World Congress*, Brussels, Belgium, Vol. Artificial Intelligence, pp. 214-217 (1991).

[47] H. Yuize, T. Yagyu, M. Yoneda, Y. Katoh, S. Tano, M. Grabish, and S. Fukami : "Decision Support System for Foreign Exchange Trading - Practical Implementation -," *IFES'91*, Yokohama, Japan, pp. 971-982 (1991).

[48] 田野俊一 : "意思決定支援システム," *ファジィ技術の先端的応用*, 日刊工業新聞社, pp. 419-454 (1993).

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

[50] T. Murofushi and M. Sugeno : "An Interpretation of Fuzzy Measures and the Choquet Integral with Respect to a Fuzzy Measure," *Fuzzy Sets and Systems*, Vol. 29, pp. 201-227 (1989).

[51] M. Sugeno : "Fuzzy Measures and Fuzzy Integrals - a Survey," in *Fuzzy Automata and Decision Process*, North Holland (1977).

[52] 菅野道夫 : "ファジィ測度とファジィ積分," *計測自動制御学会論文誌*, Vol. 8, pp. 218-226 (1972).

[53] Y. Kato, H. Yuize, M. Yoneda, K. Takahashi, S. Tano, Y. Yagyu, M. Grabish, and S. Fukami : "Gradual Rules in a Decision Support System for Foreign Exchange Trading," *The 2nd International Conference on Fuzzy Logic and Neural Networks (IIZUKA'92)*, pp. 625-628 (1992).

[54] Y. Kato, T. Arnould, T. Miyoshi, and S. Tano : "Conjunction and Disjunction with Synergistic Effect," *Second IEEE International Conference on Fuzzy Systems*, San Francisco, California, USA, pp. 225-230 (1993).

[55] 加藤康記, T. Arnould, 三好力, 田野俊一 : "ルール前件部における相関関係の処理方式," *第 9 回 ファジィシステムシンポジウム*, FA4-2, pp. 589-592 (1993).

[56] 加藤康記, 田野俊一, 三好力, T. Arnould : "ファジィプロダクションルールにおける相乗性について," *情報処理学会 第 45 回 全国大会*, Vol. 2, 1H-2, pp. 75-76 (1992).

[57] B. Buchanan and E.H. Shortliffe : Rule-Based Expert System - The MYCIN Experiments of the Stanford Heuristic Programming Project, Addison-Wesley (1984).

[58] E.H. Shortliffe : Computer-Based Medical Consultations : MYCIN, American Elsevier Publishing Co., Inc. (1976).

[59] G. Shafer : A Mathematical Theory of Evidence, Princeton University Press (1976).

[60] M. Umano : "A Fuzzy Production System," in *Fuzzy Logic in Knowledge Engineering*, Verlag TUV Rhinland, pp. 194-208 (1986).

[61] M. Umano : "Implementation of Fuzzy Production System," *Third IFSA World Congress*, pp. 450-453 (1989).

[62] M.L. Minsky : "A Framework for Representing Knowledge," in *The Psychology of Computer Vision*, McGraw-Hill (1975).

[63] Y. Kato, T. Arnould, T. Miyoshi, and S. Tano : "Processing of Correlations in Fuzzy Rules," *The Second World Congress on Expert Systems*, Risbon, Portogal (1994).

[64] A. Maeda, R. Someya, S. Yasunobu, and M. Funabashi : "A Fuzzy-based Expert System Building Tool with Self Tuning Capability or Membership Functions," *World Congress on Expert Systems*, USA, pp. 639-647 (1991).

[65] M.M. Gupta and J. Qi : "Design of Fuzzy Logic Controllers Based on Generalized T-operators," *Fuzzy Sets and Systems*, Vol. 40, pp. 473-489 (1991).

[66] 水本雅晴 : "T-ノルムと Fuzzy 制御," *第 1 回 ファジィシステムシンポジウム*, pp. 125-132 (1985).

[67] 水本雅晴 : "種々の Fuzzy 推論の下での Fuzzy 制御," *第 2 回 ファジィシステムシンポジウム*, pp. 131-136 (1986).

[68] 水本雅晴 : "ファジィ制御に対する改善法," *第 3 回 ファジィシステムシンポジウム*, pp. 153-158 (1987).

[69] 水本雅晴 : "ファジィ制御に対する改善法 (II)," *第 4 回 ファジィシステムシンポジウム*, pp. 91-96 (1988).

[70] 水本雅晴 : "ファジィ制御の改善法 (IV) (代数積-加算-重心法による場合)," *第 6 回 ファジィシステムシンポジウム*, TA1-1, pp. 9-13 (1990).