教育資料與圖書館學，第 37 卷，第 4 期，頁 376-403，民 89 年 6 月

基於 XML 之分散式模糊知識管理系統模式

淡江大學 教資系
助理教授 林信成

An XML-based Distributed System Model of Fuzzy Knowledge Management

Sinn-Cheng Lin

Assistant Professor
Department of Educational Media and Library Sciences,
Tamkang University,
Taipei, Taiwan, R.O.C.
E-Mail: sclin@mail.tku.edu.tw

摘要

This paper focuses the attention on some issues of Web-based fuzzy knowledge management system. In the recent years, fuzzy theory has been an important research field of knowledge engineering. It provides a fundamental theory to deal with the linguistic information of human intelligence mathematically. On the other hand, the second-generation Web that based on XML technology is expected to extend the Internet beyond information delivery to knowledge management. To join together the advantages of both technologies, three major works are accomplished in this paper: First, a distributed system model of fuzzy knowledge management based on XML technology is proposed. Secondly, the structure of the fuzzy knowledge base is deeply analyzed for defining a fuzzy DTD. Finally, a prototype of DTD for creating fuzzy knowledge base by XML is developed.

一、緒論

「知識」是人類文明的重要資產，有關知識管理的研究主題理所當然成為人類關注的焦點。傳統上，圖書館是最典型的知識蘊藏寶庫，不過隨著資訊科技的發展，網路不但成為資訊流通和知識聚集的重要場所，更日漸朝自動化、智慧化的方向發展；一般認為，網路資源若能有效整合，輔以具備知識思維能力的智慧型系統（Intelligent Systems），將建構出人類有史以來最大的知識庫管理系統（Knowledge Base Management System），輔助人類進行推理與決策。

長久以來，人類一直夢想著有朝一日能夠創造出具有思維能力的機器，因此，人工智慧（Artificial Intelligence，簡稱 AI）自 1950 年代起便逐漸的成為一個重要的研究領域，涵蓋層面包括了專家系統（Expert Systems）、自然語言處理（Natural Language Processing）、電腦視覺（Computer Vision）、機器人學（Robotics）、機器學習（Machine Learning） ... 等。人工智慧研究所要解答的

一個最根本的問題是：人到底是如何思考的呢？其實，人類的思維過程不外乎藉由各種管道獲取知識（Knowledge），然後進行推理（Reasoning），最後做出決策（Decision），由此可見，「知識」乃是人類智慧中最基本的一環。然而，知識的獲得則需要經過資料處理及資訊管理的過程才能粹取出來，如圖 1 所示，大量的資料（Data）經過組織、整理之後成為有用的資訊（Information）；而眾多的資訊經過歸納、演繹之後，才能構成知識；有了足夠的知識，才能做出正確的推理與決策，此乃人類思維的基本體系。

圖 1 人類思維的基本體系

然而，人工智慧發展多年，雖然一直試圖利用電腦（Computer）來模擬人類的知識思維，卻經常遭遇瓶頸。究其原因，電腦強於人腦的只在其計算能力罷了，對於無法以有效的計算法則解決的問題，如概念、思考、推理、識別 ...等，電腦就無能為力了。在真實世界中，人類有許多思維過程是非常「模糊」（Fuzzy）的，例如：「如果路況很不好，則車速慢一點」，其中「很不好」、「慢一點」都是沒有明確定義的模糊概念，但是人腦就是能夠處理，實在神奇。想一想，人腦並未經過詳細精確的運算，卻能做出正確的判斷，電腦卻辦不到，為什麼？因為數值計算並不等同於知識思維！事實上，當人們在做決策時，經常是使用一些模糊語意規則，諸如：「如果　　，則　　」，但一直沒有適合的方法來供電腦遵循，直到 L. A. Zadeh 於 1965 年提出了模糊集合論（Fuzzy Sets）[1]和 1972 年的近似推理（Approximation Reasoning）[2]之後，總算有了一個數學理論可以處理人類思維的語意資訊。「Fuzzy」一詞在字典中的解釋是「似毛絨的」，引申為「模糊的」、「朦朧的」之意，英文上的含意是負面的，因此起初並未廣被一切講求精確的工程研究人員所接受。直到 70 年代初期出現成功的工業應用[3]，才逐漸受到關注，但仍停留在學術研究階段；80 年代末期，日本結合各界力量，大舉開發 Fuzzy 家電產品，才將其推向市場，引起廣泛注意。90 年代起國內外的產、官、學界紛紛投入大量的人力物力進行模糊理論之研究，各式各樣的應用也紛紛出爐，印證了模糊理論的實用性。如今，模糊理論已經成為一門專門探討人腦如何利用模糊訊息或不完全資料，不需經過精密繁雜的計算過程，仍能作出正確判斷，進而瞭解人腦是如何從模糊中找出路的理論[4]，其涵蓋範圍極為廣泛，包括模糊集合（Fuzzy Set）、模糊關係（Fuzzy Relation）、模糊邏輯（Fuzzy Logic）、模糊量測（Fuzzy Measure）、模糊推理（Fuzzy Reasoning）...等[5]；應用領域更是包含了模糊控制（Fuzzy Control）、模糊決策（Fuzzy Decision）、模糊專家系統（Fuzzy Expert System）、模糊資訊檢索（Fuzzy Information Retrieval）、模糊影像處理（Fuzzy Image

[1] L. A. Zadeh, "Fuzzy Sets," Information and Control, Vol. 8, pp. 338-353, 1965.
[2] L. A. Zadeh, "Outline of a new approach to the analysis of complex systems and decision processes," *IEEE Trans. on Systems, Man and Cybernetics*, SMC-3, no. 1, pp.28-44, 1973.
[3] E. H. Mamdani, "Applications of fuzzy algorithms for simple dynamic plant," *Proc. IEE*, vol. 121, no. 12, pp. 1585-1588, 1974.
[4] 林信成、彭啟峰，「Oh! Fuzzy — 模糊理論剖析」，第三波，民 83。
[5] G. J. Klir and T. A. Folger, *Fuzzy Sets, Uncertainty, and Information*, Prentice-Hall: New Jersey, 1988.

Processing)、模糊模式識別（Fuzzy Pattern Recognition)、模糊規劃（Fuzzy Programming)、模糊資料庫（Fuzzy Data Base） ... 等[6]。

本文旨在針對以模糊理論為基礎所建構的通用模糊推論系統（Fuzzy Inference System)，提出一個基於 XML 之分散式知識管理模式，以因應網路時代的需求。模糊推論系統簡單的說，是一個「以模糊知識為依據，以模糊推理為方法的智慧型系統」，其主要核心是由知識庫（Knowledge Base）和推論引擎（Inference Engine）所構成。傳統上，模糊推論系統都是採用集中式知識管理方式，將知識庫與推論引擎集中於一處，並未探討分散式管理的課題。然而，在網際網路蓬勃發展的今日，分散式管理的需求日益殷切。而 XML(eXtensible Markup Language，可擴展標注語言) 則是由 W3C 所倡議的新一代的網路資源整合技術，由於 XML 具有可擴展性、高度結構化和良好的資料組織能力，能夠有效的表達網路上各種知識，為資料的交換和處理提供新的機制[7]，因此，本研究遂採用其作為分散式知識管理的解決方案之一，進而探討其可行性及實用性。

二、知識工程與模糊推論系統

（一）知識工程

「知識」是人類思維的基本要素，因此，要讓機器如同人類一樣理解事物，就必須研究以機器擷取、表達、處理知識的技術與方法，也就是所謂的知識工程（Knowledge Engineering)，專家系統（Expert System）便是知識工程最成功的應用。專家系統利用「知識管理」(Knowledge Management）概念，是一個「具有模擬人類專家決策能力的電腦系統」[8]。一個典型的專家系統，必須利用各種知識擷取(Knowledge Acquisition)管道獲取專家的領域知識(Domain Knowledge)再以知識表示法（Knowledge Representation）將人類知識轉存成知識庫（Knowledge Base)，而在知識庫之上有一個具備邏輯推理能力的推論引擎（Inference Engine)，使用者經由使用者介面輸入欲查詢的事實（Facts)，推論引擎將該事實與知識庫中的先備知識進行比照，做出建議性的推理和決策，如圖 2 所示。所以專家系統可視為是「一個以知識為依據，以推理為方法的智慧型系統」[9]。依此觀之，「知識」和「推理」不但是人類智慧的結晶，更是專家系統中不可或缺的核心要素。

[6] 同註4，頁 1-7~頁 1- 16。
[7] 林信成，「XML 相關技術與下一代 Web 出版趨勢之研究」，教育資料與圖書館學，第三十七卷，第二期，民 88 年 12 月。
[8] 同註10, Page 1.
[9] 葉怡成、郭耀煌，專家系統—方法、應用與實作，民 80，全欣資訊圖書，頁 7。

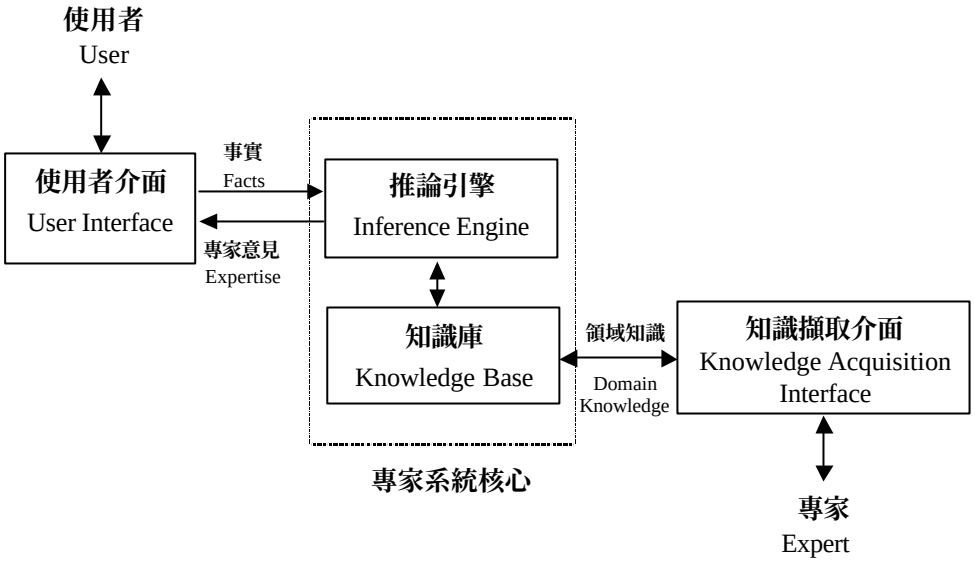


圖 2 專家系統概念圖

知識表示法是表達知識的基本方法。目前，已被提出的知識表示法眾多，較重要的有：規則式（Production Rules）**表示法、邏輯與集合**（Logic and Sets）**表示法、語意網路**（Semantic Nets）**表示法、框架**（Frames）**表示法、劇本**（Scripts）**表示法、物件導向表示法**（Object-Oriented Representation）**等**[10]**。在眾多知識表示法中，「規則式表示法」無疑的是最符合人類思維模式，也是應用最廣的一種，其最主要的特色是將人類知識描述成一條一條「如果　　則　　」的推論規則，以供推論引擎使用。**

（二）模糊理論

傳統上，知識庫和推論引擎大都採用布林邏輯（Boolean Logic）**和明確推理**（Crisp Reasoning）**來運作。然而如前所述，在真實世界中，人類有許多思維過程是非常「模糊」的，這與需要精確數據才能計算的電腦，在運作上顯然有著極大的差異。模糊理論的發展使得人類思維與電腦運算之間搭起了一座交流的橋樑，它可以將模糊概念加以量化，例如，人類日常用語中有許多曖昧不明、含意模糊的語句：昨天「很熱」、你的車速「太快」了、他並「不太老」...，其中「很熱」、「太快」、「不太老」並未指明真正的數值是多少，但人腦卻可以很容易的接收並處理，這些模糊語意詞**（Fuzzy Linguistic Terms）**經由模糊集合**（Fuzzy Set）**的歸屬函數**（Membership function）**加以量化成數值後，便可交由電腦處理，而電腦也因此而稍具人腦思考的雛形。**

以下說明在模糊理論中幾個與本文有關的主題。

1. **模糊集合**（Fuzzy Set）

在論域（Universe of discourse）X **中的任一模糊集合** A**，可由以下的歸屬函數加以描述：**

$$m_A : X \rightarrow [0,\ 1] \qquad (1)$$

[10] Giarratano and Riley, Expert Systems: Principle and Programming, PWS-KENT, 1989, Page 63-102.

其中 $x \in X$ **，** $m_A(x)$ **代表** x **隸屬於** A **的歸屬度**。

2. **常用歸屬函數類型**

幾種常見的標準型歸屬函數如下所述：

(1) S **型（**S-Shape**）歸屬函數：**

$$S(x;l,r)=\begin{cases}0, & \text{for } x\le l\\ 2\left(\dfrac{x-l}{r-l}\right)^2, & \text{for } l\le x\le \dfrac{l+r}{2}\\ 1-2\left(\dfrac{r-x}{r-l}\right)^2, & \text{for } \dfrac{l+r}{2}\le x\le r\\ 1, & \text{for } x\ge r\end{cases} \tag{2}$$

S **型歸屬函數常用於表述諸如重、厚、長、大、熱、胖、高、好、多、昂貴、年老 ... 等代表較高數量級的模糊概念，** l **和** r **分別代表其左、右端點**。

(2) Z **型（**Z-Shape**）歸屬函數：**

$$Z(x;l,r)=\begin{cases}1, & \text{for } x\le l\\ 1-2\left(\dfrac{x-l}{r-l}\right)^2, & \text{for } l\le x\le \dfrac{l+r}{2}\\ 2\left(\dfrac{r-x}{r-l}\right)^2, & \text{for } \dfrac{l+r}{2}\le x\le r\\ 0, & \text{for } x\ge r\end{cases} \tag{3}$$

顯然地， $Z(x; l, r) = 1\text{-}S(x; l, r)$**，兩者互為模糊補集**。Z **型歸屬函數常用於表述諸如輕、薄、短、小、冷、瘦、矮、壞、少、便宜、年輕 ... 等代表較低數量級的模糊概念，** l **和** r **分別代表其左、右端點**。

(3) π **型（**π-Shape**）歸屬函數：**

$$p(x;b,m)=\begin{cases}S(x;m-b,m-\dfrac{b}{2},m), & \text{for } x\le m\\ Z(x;m,m+\dfrac{b}{2},m+b), & \text{for } x\ge m\end{cases} \tag{4}$$

π **型歸屬函數常用於表述諸如普通、中等、差不多是** m ... **等代表中等數量級或近似某個數值** m **的模糊概念。因此，** m **代表該函數的中點，** b **為其擴張度**。

(4) **三角型（**Triangular**）歸屬函數：**

$$Tri(x;l,m,r)=\begin{cases}(x-l)/(m-l), & l\le x<m\\ 1, & x=m\\ (r-x)/(r-m), & m<x\le r\\ 0, & otherwise\end{cases} \tag{5}$$

其中， l, m, r **分別代表該函數的左端點、中點、右端點**。

(5) **梯型（**Trapezoidal**）歸屬函數：**

$$Trap(x;l,m,n,r)=\begin{cases}(x-l)/(m-l), & l\le x<m\\ 1, & m\le x\le n\\ (r-x)/(r-n), & n<x\le r\\ 0, & otherwise\end{cases} \tag{6}$$

其中， l, m, n, r **分別代表該函數的左端點、左肩點、右肩點、右端點。**

(6) **高斯型（**Gaussian-Shape**）歸屬函數：**

$$G(x;m,b)=\exp\left(-\left(\frac{x-m}{b}\right)^2\right) \tag{7}$$

m **代表該函數的中點，** b **為其變異度（或擴散度）。**

三角型、梯型和高斯型歸屬函數都是屬於泛用型的歸屬函數，只要妥善調整其參數值便可以表達各種不同的模糊概念。

(7) **對於歸屬函數無法寫成任何函數形式的任一模糊集合** A**，可將各點** x_i **的歸屬度** m_i **條列出來，借用連加符號（**Σ **或** $+$ **）表成如下的離散型式：**

$$A=\sum_{i=1}^{n} m_i / x_i = m_1/x_1 + m_2/x_2 + \ldots + m_n/x_n \tag{8}$$

3. **模糊集合運算**

令 A **和** B **分別為論域** X **中的兩個模糊集合，則**

(1) **交集運算（**Intersection**）**

A **和** B **的交集記作** $A\cap B$**，亦為** X **中的一個模糊集合，其歸屬函數為：**

$$m_{A\cap B}(x)=T\big(m_A(x),m_B(x)\big) \tag{9}$$

其中 $T(.)$ **稱為** T-**範數（**T-norm**），是一個從**$[0,\ 1]\times[0,\ 1]$**映射至**$[0,\ 1]$**的函數。常用的幾個** T-norm **函數為：**

模糊交集（fuzzy intersection**）：** $T(a,\ b)=\min(a,\ b)$ (9a)
代數乘積（algebraic product**）：** $T(a,\ b)=ab$ (9b)
界限乘積（bounded product**）：** $T(a,\ b)=\max(0,\ x+y-1)$ (9c)

(2) **聯集運算（**Union**）**

A **和** B **的聯集記作** $A\cup B$**，亦為** X **中的一個模糊集合，其歸屬函數為：**

$$m_{A\cup B}(x)=C\big(m_A(x),m_B(x)\big) \tag{10}$$

其中 $C(.)$ **稱為** T-**共範數（**T-conorm**），是一個從**$[0,\ 1]\times[0,\ 1]$**映射至**$[0,\ 1]$**的函數。常用的幾個** T-conorm **函數為：**

模糊聯集（fuzzy union**）：** $C(a,\ b)=\max(a,\ b)$ (10a)
代數和（algebraic sum:**）：** $C(a,\ b)=a+b-ab$ (10b)
界限和（bounded sum**）：** $C(a,\ b)=\min(1,\ a+b)$ (10c)

(3) **補集運算（**Complement**）**

A **的補集記作** $\overline{A}$**，亦為** X **中的一個模糊集合，其歸屬函數為：**

$$m_{\overline{A}}(x)=1-m_A(x) \tag{11}$$

（三）模糊推論系統

利用模糊理論所建構的系統，泛稱為模糊系統（Fuzzy Systems）；在模糊系統中加入近似推理（或稱模糊推理），便形成模糊推論系統（Fuzzy Inference Systems）。近似推理是一種擬人式的推論方式，以開車為例：人類在駕駛時，並未實際測量道路彎度、車與路口路邊的距離，也從未考慮過精確的方向盤角度、油門或剎車的大小，只憑藉一些模糊的訊息及規則，像是「如果道路有點右彎，則方向盤向右打一點並開慢一點」、「如果距路口很遠且路況良好，則開快一點」，即能將汽車控制於正常車道上。人腦並未經過詳細精確的運算，卻能做出正確的判斷，模糊推論系統就是被發展出來模擬人類這種「過程模糊，結果精確」的推論方式。

一個典型的模糊推論系統，如圖 3 所示，主要由五大功能模組所構成：

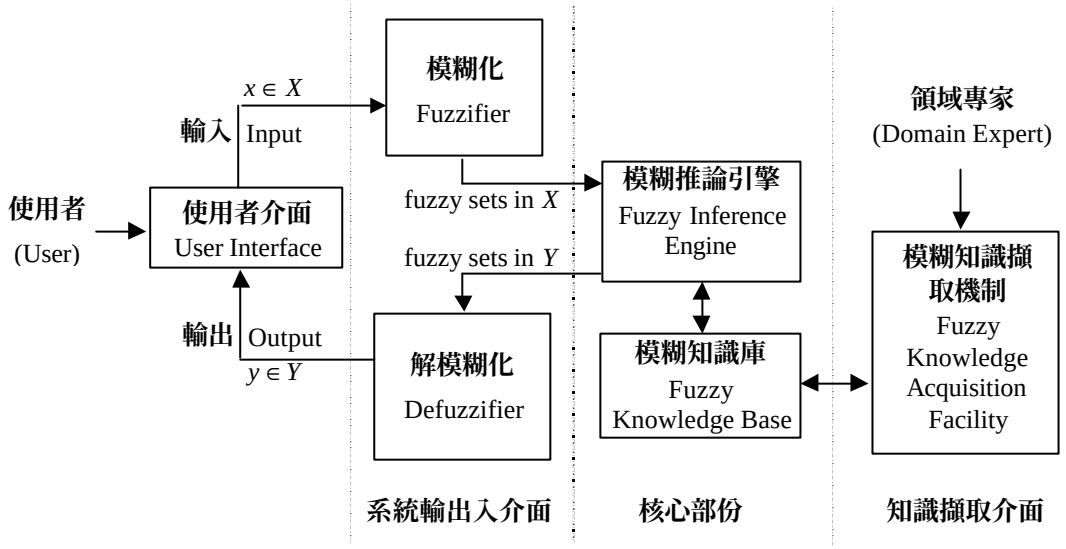


圖 3 模糊推論系統

其中，核心部份具有兩個功能模組：模糊知識庫（Fuzzy Knowledge Base）、模糊推論引擎（Fuzzy Inference Engine）；而介面部份由三個功能模組構成：模糊知識擷取機制（Fuzzy Knowledge Acquisition Facility）模糊化介面（Fuzzifier）、和解模糊化介面（Defuzzifier）。分述如下：

(1) 模糊知識庫

模糊知識庫是模糊推論系統中用以儲存人類知識的寶庫，主要由兩部份構成，其一為記錄專家經驗法則之模糊規則庫（Rule Base），另一為定義模糊語意之資料庫（Data Base）。模糊規則通常是以「若 ~ 則 ~」（「IF ~ THEN ~」）的方式記錄，例如一個典型的教學規則可能如下所示：

IF　學生程度「好」 AND 學習能力「強」
THEN　課程進度「快一點」 AND 課程難度「高一點」

其中，「好」、「強」、「快一點」、「高一點」等語意詞，都被視為模糊集合，其對應的歸屬函數必須在語意資料庫中加以定義。

若以通式表示，一個模糊規則庫可記作 $\boldsymbol{RB} := \bigcup_{j=1}^{N} R_j$，是由 N 條模糊規則所

組成，而每條規則 R_j 可表示為：

$$R_j : IF\ x_1\ is\ A_{1j}\ \text{and}\ x_2\ is\ A_{2j}\ \text{and}\ \ldots x_n\ is\ A_{nj}$$
$$THEN\ y\ is\ B_j \tag{12}$$

其中 x_i $(i = 1, 2, ..., n)$ 為輸入變數，y 為輸出變數；輸入語意值 A_{ij} 和輸出語意值 B_j $(j = 1, 2, \ldots, N)$ 分別是定義於輸入論域 X_i 和輸出論域 Y 的模糊集合。

(2) 模糊推論引擎

模糊推論引擎主要以近似推理為方法，根據知識庫中的先備知識進行模糊推論得出結果。近似推理衍生自模糊邏輯，其主要的邏輯運算雖然也使用 AND、OR、NOT ... 等，但相較於布林邏輯二值運算（Binary Computing）的結果只有 1（邏輯真值 True）、0（邏輯假值 False）兩種情況，模糊邏輯是一種軟性計算（Soft Computing）方式，其結果介於 0~1 之間，代表著不同等級的真假程度，更符合人類思維的特性。

模糊推論過程如下：將模糊規則的前件部各項輸入值與語意值進行模糊匹配，轉換成模糊集合後，規則庫中有關的模糊邏輯運算 AND、OR、NOT，便等效於模糊集合的交集、聯集、補集運算了，所得的結果稱為該條規則的觸發強度（Firing Strength）：

$$m_{R_j}(\underline{x}) = \bigcap_{i=1}^{n} m_{A_{ij}}(x_i) \tag{13}$$

其中，，$m_{R_j}(\underline{x})$ 代表第 j 條規則的觸發強度（$\underline{x} = [x_1\ x_2\ \ldots\ x_n]^{\mathrm{T}}$），交集符號 $\cap$ 代表 AND 運算，可用 T-norm 完成。經由觸發強度可以得出第 j 個模糊基底函數（Fuzzy Basis Function）$r_j(\underline{x})$ ：

$$r_j(\underline{x}) = \frac{m_{R_j}(\underline{x})}{\sum_{j=1}^{N} m_{R_j}(\underline{x})} \tag{14}$$

(3) 模糊知識擷取機制

知識庫中的知識仍需得至專家提供，因此需要一個知識擷取模組來完成。模糊理論發展之初，知識庫的建立大都倚賴領域專家（Domain experts）提供經驗，再由知識工程師將其轉換成對應的模糊規則及歸屬函數。其優點是設計原理簡單且直觀；而缺點則是費時耗力。因為知識的擷取並非易事，且隨著所要解決的問題愈趨繁瑣，推論規則也相對愈趨複雜而更難於歸納整理。幸好拜科技之賜，今日的電腦運算能力相較於十年前，可謂一日千里，加以眾多學習理論日趨成熟，為模糊推論系統的建立另外提供了出路。目前已有許多研究成果朝著模糊系統自我學習（Self-learning）或自我組織（Self-organizing）的方向努力[11]，期望藉由各式各樣人性化的自學方案，讓模糊推論系統具備學習能力，達到自我建立知識庫

[11] 林信成、陳永耀，「自我學習法則在模糊推論系統上之應用」，模糊系統學刊，第一卷，頁 7-26，民 84（*Journal of Chinese Fuzzy Systems Association*, Vol.1, pp7-26, 1995）。

的目的，免去人工建立之苦，並且希望經由此種方式所架構出來的模糊推論系統能超越人工建立者，甚至超越原來的領域專家。

(4) 模糊化介面

用以將明確輸入值 $x_0 \in X$ 轉換成定義於 X 的模糊集合 A_x。特殊地，若 A_x 是一個支點（Support）在 x_0 的模糊單點集（Fuzzy Singleton），則稱此種模糊化方式為單點集模糊化（Singleton Fuzzification），是一種最常見的模糊化方式。

(5) 解模糊化介面

相對於模糊化的反向動作，解模糊化主要將定義於 Y 的模糊集合 B 轉換為精確的輸出值 $y \in Y$。最常用的解模糊化運算為「加權平均解模糊化法」（Weighted Average Defuzzification）：

$$y = \frac{\sum_{j=1}^{N} j_j m_{R_j}(\underline{x})}{\sum_{j=1}^{N} m_{R_j}(\underline{x})} := \underline{j}^{\mathrm{T}} \underline{r}(\underline{x}) \tag{15}$$

其中，$\underline{r} = [r_1\ r_2\ \ldots\ r_N]^{\mathrm{T}}$ 為模糊基底函數向量，$\underline{j} = [j_1\ j_2\ \ldots\ j_N]^{\mathrm{T}}$，$j_j$ 為輸出語意值 B_j 的支點。

三、分散式模糊推論與知識管理

（一）系統模式

本研究基於分散式知識管理概念，提出一個基於 Internet/Intranet 架構之模糊推論與知識管理系統模式，如圖 4 所示。

(1) FKB #1 ~ FKB #*n*

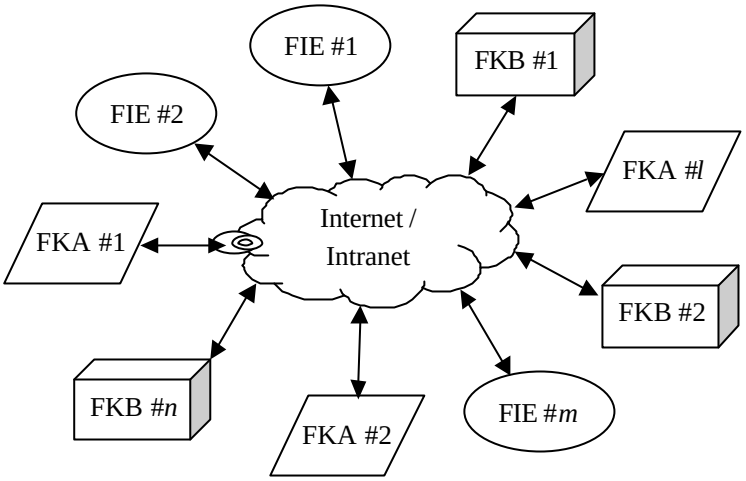
這是分散於網路上的 n 個模糊知識庫，又可分為相同領域的同質性知識庫（Homogeneous Knowledge Base）及不同領域的異質性知識庫（Heterogeneous Knowledge Base）。

(2) FKA #1 ~ FKA #*l*

這是分散於網路上的 l 個模糊知識擷取機制，主要收集及整理不同領域專家之知識，轉化為適當的模糊知識表示法（如模糊規則），並傳送至對應的模糊知識庫中儲存備用。

(3) FIE #1 ~ FIE #*m*

這是分散於網路上的 m 個模糊推論引擎，可根據不同的問題連結至適當的知識庫尋求解答。此外，為了與使用者有良好互動，此部份應依需要與使用者介面、模糊化和解模糊化介面合併。



FKB: Fuzzy Knowledge Base
FIE: Fuzzy Inference Engine
FKA: Fuzzy Knowledge Acquisition agent

圖 4 分散式模糊推論與知識管理系統架構

基於此一架構，可以發展出「同質性分散式知識管理系統」和「異質性分散式知識管理系統」：

(1) **同質性分散式知識管理系統：**

此類型系統指的是同一個專業領域可以跨越空間限制，在網路上建立分散式的同質性知識庫，它們能獨立作業完成區域性的應用，也能彼此關連完成整體性的推論。

(2) **異質性分散式知識管理系統：**

此類型系統指的是不同專業的知識可以跨界合作，使得原本各自獨立的異質性知識庫得以跨領域的完成整合推論，以達成整體知識共享的目標。

（二）基於 XML 之模糊知識庫

一般認為，網路資源若能有效整合，輔以具備知識思維能力的智慧型系統，將建構出人類有史以來最大的知識管理系統，輔助人類進行推理與決策。本文針對模糊推論系統提出以 XML 為基礎之分散式知識管理模式，以因應網路時代的需求。

XML 是由 W3C（World Wide Web Consortium，全球資訊網協會）所倡議的新一代的網路資源整合技術[12]，其所具有的結構性、可擴展性和資料樣式分離原則，不但在資料的管理、交換上擁有極為卓越之性能，良好的自我描述性，更能夠有效的表達網路上的各種知識，為知識管理提供新的機制。

其主要特色略述如下：

(1) **可擴展性**

XML 既可視為是一種在 Web 上建立結構化資料的通用格式（Universal Format），也可視為發展其他應用語言的低階語法（Low-Level Syntax）[13]，這就是 XML 被稱為 Meta-Language 的原因，也是 XML 最引以為傲的可擴展性

[12] 1996 年 7 月「XML 工作小組」（XML Working Group）在 W3C 的贊助下成立，當年 11 月提交 XML 初稿，並於 1998 年 1 月 10 日正式通過 XML 1.0 規範，成為 W3C 的一個建議標準（Recommendation）「XML 工作小組」。

[13] 同註21。

（Extensibility）。目前已有許多經由 XML 所定義並使用於不同領域的應用語言，例如應用於數學方面的 MathML（Mathematical Markup Language）[14]、應用於向量圖的 SVG（Scalable Vector Graphics）[15]、應用於多媒體領域的 SMIL（Synchronized Multimedia Integration Language）[16]、應用於描述網路資源的 RDF（Resource Description Framework）[17]、應用於網路推播頻道的 CDF（Channel Definition Format）[18]... 等。

(2) 自我描述性

XML 的標籤可根據不同的用途來定義，因此在語意層次上具備一定程度的自我描述（Self-Description）特性，有助於提昇處理程式解讀資料內容的能力與進行自動處理的效率。因此，藉由 XML 可以將資料由機器可讀取（Machine Readable）層次提昇至機器可理解（Machine Understandable）層次，對於在網路上建立智慧型系統有著莫大的幫助。

(3) 結構性

XML 具有嚴格的規範以適應廣泛的應用，因而造就了 XML 文件強烈的結構性，在資料處理和機器理解方面具備了先天的優勢，這也是促使 XML 迅速成為重要機讀格式的主因之一。

XML 文件可根據其結構性概分為兩類[19]：

(a) 完備的（Well-Formed）XML 文件。

(b) 有效的（Valid）XML 文件。

一個 XML 文件只要合於 XML 規範中所制訂的語法規則，便可以很容易的具備「完備性」（Well-Formedness）；不過，要達到「有效性」（Validity），則除了滿足語法規則外，尚必須遵循某些額外定義的語意才行。XML 處理器會根據 XML 規範中所定義的完備性和有效性來查核文件內容，一旦發現不合規定的文件便會拒絕處理並告知對方，這對於在網路上進行資料交換與資訊共享是非常必要的—不符合公認標準的資料可以不被接受。「文件類型定義」（Document Type Definition，簡稱 DTD）和「XML 綱要」（XML Schema）是 XML 處理器據以確定文件有效性的兩大重要機制。

(4) 資料和樣式分離原則

XML 強調的是如何以適當的結構來組織資料，對於外在的表現則必須透過其他顯示機制才能達成，這就是 XML 文件的內在、外貌（即資料、樣式）分離原則，這使得資料交換時只要傳遞資料與結構即可，合乎機器運作及處理原則。

XML 和 HTML 一樣都是從 SGML 演變而來的，只不過 HTML 是 SGML 的一個應用語言（Application），而 XML 卻是 SGML 的一個精簡子集（Subset）。XML 將 SGML 去蕪存菁，捨棄約百分之二十複雜罕用的部份，承襲了其他百分之八十的特點，是以具備了 SGML 所沒有的簡易性與靈活性，又有著 HTML 所欠缺的擴展性與結構性。因此，稱 XML 為主導「第二代 Web」（Second-Generation Web）的重要技術實不為過[20]。XML 的發展將使得許多理想得以實現[21]：

---

[14] "Mathematical Markup Language (MathML)", available at <http://www.w3.org/Math/>.

[15] "W3C Scalable Vector Graphics (SVG)", available at <http://www.w3.org/Graphics/SVG/>.

[16] "Synchronized Multimedia", available at <http://www.w3.org/AudioVideo/#SMIL>.

[17] "Resource Description Framework (RDF)", available at <http://www.w3.org/RDF/>.

[18] "Channel Definition Format (CDF)", available at <http://www.w3.org/TR/NOTE-CDFsubmit.html>.

[19] XML 1.0 規範中對於完備性（Well-Formedness）和有效性（Validity）有嚴謹的定義，可由以下 URL 取得詳細說明 <http://www.w3.org/TR/1998/REC-xml-19980210>.

[20] Jon Bosak and Tim Bray, "XML and the Second-Generation Web", Scientific American, May 1999,

(1) 促進國際化媒體獨立（Media-Independent）的電子出版。
(2) 允許產業間定義平台獨立（Platform-Independent）的協定來進行資料交換，尤其是在電子商務方面。
(3) 以某種形式傳送資訊給使用者代理程式（User Agents），使其能在接收之後自動處理。
(4) 讓人們很容易的便能使用平價軟體來處理資料。
(5) 允許人們以自己想要的方式顯示資訊。
(6) 提供 Metadata，不但能幫助人們找到所需的資訊，更能幫助資訊生產者與消費者找到對方。

四、模糊知識庫之結構分析與 DTD 設計

「文件類型定義」（Document Type Definition，簡稱 DTD）是 XML 處理器據以確定文件有效性的重要機制。DTD 是沿襲自 SGML 的技術，主要用途在於定義文件中的元素型態、結構、範圍、順序等，以作為撰寫 XML 文件的藍本。在網路上所建構的模糊知識庫，最基本的要求便是要能在不同的知識管理系統間進行資料交換、資源共享，才能達成分散處理、推論的目的，因此遵循一定的規範作為建立知識庫的標準是有其必要性的。因此，以下基於前述的模糊理論，進行模糊知識庫之結構分析與 DTD 設計，以作為建構分散式模糊知識庫之依據。

（一）知識庫結構

一個典型的模糊知識庫基本上包含兩大部份：規則庫和資料庫。因此，DTD 的定義應反應此一結構，如下所示：

```
<!ELEMENT knowledgebase (rulebase, database)>
```

（二）規則庫結構

根據研究，一個「多輸入多輸出」（Multi-Input-Multi-Output，簡稱 MIMO）的模糊推論系統通常可被等效的分割為若干個「多輸入單輸出」（Multi-Input-Single-Output，簡稱 MISO）的模糊推論系統[22]，這使得問題得以簡化，只把焦點放在 MISO 系統即可。

由於一個模糊規則庫可記作 $\boldsymbol{RB} := \bigcup_{j=1}^{N} R_j$，而每條規則 R_j 可表示為如下的 MISO 型式：

$$R_j : IF\ x_1\ is\ A_{1j}\ \text{and}\ x_2\ is\ A_{2j}\ \text{and}\ \ldots x_n\ is\ A_{nj}$$
$$THEN\ y\ is\ B_j$$

因此，規則庫的結構可歸納如下：

(1) 規則庫是由一條以上的規則所組成；
(2) 每條規則分為輸入的 *IF* 和輸出的 *THEN* 兩部份；

---

also available at <http://www.sciam.com/1999/0599issue/0599bosak.html>.
[21] "Extensible Markup Language (XML) Activity", available at <http://www.w3.org/XML/Activity.html>.
[22] L. X. Wang, *Adaptive Fuzzy Systems and Control*, Englewood Cliffs, NJ: Printice Hall, 1994.

(3) 輸入的 *IF* 部份由一個以上的輸入條件合成；
(4) 輸入條件是由輸入變數名稱及其對應的輸入語意值構成的；
(5) 輸入合成運算為模糊邏輯 AND 或 OR 運算，預設為 AND；
(6) 輸出的 *THEN* 部份只有一個變數；
(7) 輸出結論是由輸出變數名稱及其對應的輸出語意值構成的；

基於以上分析，一個可行的 DTD 定義中，滿足規則庫結構之部份應如下所示：

```
<!ELEMENT rulebase (rule+)>
<!ELEMENT rule (if, then)>
<!ELEMENT if (input+)>
  <!ATTLIST if
            op  (and|or)  "and">
<!ELEMENT then (output)>
<!ELEMENT input EMPTY>
  <!ATTLIST input
            name  CDATA  #REQUIRED
            value  CDATA  #REQUIRED>
<!ELEMENT output EMPTY>
  <!ATTLIST output
            name  CDATA  #REQUIRED
            value  CDATA  #REQUIRED>
```

（三）資料庫結構

資料庫記錄各項輸出入語意變數的範圍及定義，作為界定規則庫中所使用到的各個模糊量的換算依據。若依上述的規則結構，歸納輸入部份的及輸出部份所應含括的的資料定義為：

(1) 輸入部份必須定義一個以上的輸入變數資料；
(2) 輸入變數的定義包含輸入論域的範圍和輸入語意的歸屬函數；
(3) 輸出部份只要定義一個輸出變數資料；
(4) 輸出變數的定義包含輸出論域的範圍和輸出語意的歸屬函數；

基於以上分析，一個可行的 DTD 定義中，滿足資料庫結構之部份應如下所示：

```
<!ELEMENT database (inpDefs, outDefs)>
<!ELEMENT inpDefs (inpVar+)>
<!ELEMENT inpVar (range, mfDefs+)>
  <!ATTLIST inpVar
            name  CDATA  #REQUIRED>
<!ELEMENT outDefs (outVar)>
<!ELEMENT outVar (range, mfDefs+)>
  <!ATTLIST outVar
            name  CDATA  #REQUIRED>
```

接著必須對於輸出、輸入變數之進一步資料，如論域上下限、歸屬函數各點參數值 ... 等，加以分析及定義：

(1) 論域範圍包含上下限，以最小值 min 及最大值 max 來界定；

(2) 歸屬函數除像 Z 型、S 型、π型、三角型、梯型、高斯型 ... 等常用的、定型的之外，對於無法以函數型式表示的歸屬函數，應以條列方式表達；

(3) Z 型、S 型歸屬函數，可由左端點（left）和右端點（right）兩個參數決定，相當於公式(2)、(3) 中的 l 和 r；

(4) 三角型歸屬函數可由左端點（left）、中點（center）、右端點（right）三個參數決定，相當於公式 (5) 中的 l, m 和 r；

(5) π型、高斯型歸屬函數，可由中點（center）和擴散度（breadth）兩個參數決定，相當於公式(4)、(7) 中的 m 和 b；

(6) 梯型歸屬函數可由左點（left）、左肩點（left-shoulder）、右肩點（right-shoulder）和右點（right）共四個參數決定，相當於公式(6)中的 l, m, n, r；

(7) 離散型的歸屬函數則以條列 x_i（member）和 m_i（grade）的方式表示。

以下即為所提之 DTD 定義中滿足上述分析之部份：

```
<!ELEMENT range EMPTY>
  <!ATTLIST range
            min CDATA #REQUIRED
            max CDATA #REQUIRED>
<!ELEMENT mfDefs (mf+)>
<!ELEMENT mf (zmf*|pimf*|smf*|trimf*|trapmf*|gaussmf*|pwmf*)>
  <!ATTLIST mf
            name CDATA #REQUIRED>
<!ELEMENT zmf (left, right)>
<!ELEMENT pimf (center, breadth)>
<!ELEMENT smf (left, right)>
<!ELEMENT trimf (left, center, right)>
<!ELEMENT trapmf (left, lshoulder, rshoulder, right)>
<!ELEMENT gaussmf (center, breadth)>
<!ELEMENT pwmf (li+)>
<!ELEMENT left (#PCDATA)>
<!ELEMENT center (#PCDATA)>
<!ELEMENT right (#PCDATA)>
<!ELEMENT lshoulder (#PCDATA)>
<!ELEMENT rshoulder (#PCDATA)>
<!ELEMENT breadth (#PCDATA)>
<!ELEMENT li EMPTY>
  <!ATTLIST li
            member CDATA #REQUIRED
            grade CDATA #REQUIRED>
```

完整的 DTD 請參見本文附錄。

五、XML 模糊知識庫建置範例

本節僅以一個點型的雙輸入單輸出之模糊推論系統為例，遵循上述之 DTD 結構，以 XML 建構模糊知識庫。

假設這個模糊推論系統是一個與教學有關的智慧型系統，其規則庫內容為經

過歸納整理的教學法則。例如：

如果　學息誤差　大　而且　學習速度　慢　則　教學進度　很慢

首先，將此模糊推論系統加以模式化，如圖 5 所示。

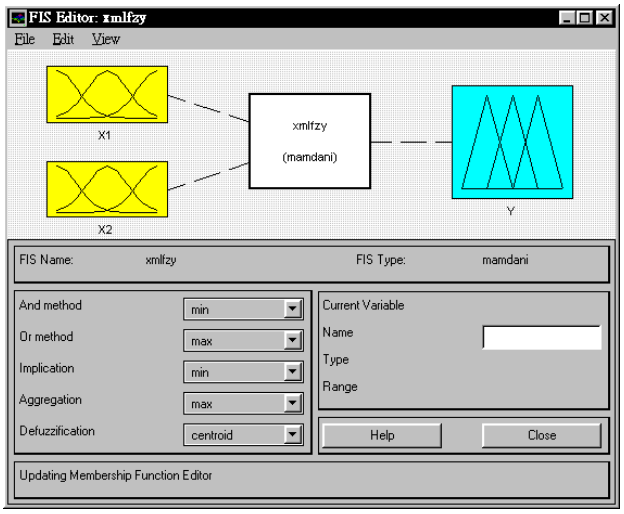


圖 5 一個典型的雙輸入單輸出模糊系統

其中，有兩個輸入變數 x_1、x_2 和一個輸出變數 y。x_1 代表學習者之學習誤差，是學習成果與學習目標之間的差距，可用來評量學習者目前對學習主題的認知程度；而 x_2 代表學習速度，是衡量學習者的吸收力、學習能力、認真程度 ... 等的綜合指標；至於輸出 y 則代表教學進度，由教學者依據學習者的學習狀況而彈性決定。以此觀之，上述的教學規則其實反映了教學者在教學過程中的一個複雜的思維及決策過程，這個過程的含意是「如果學習者對目前這個學習主題的認知不足（學息誤差大），而且如果學習者的吸收力、學習能力、認真程度都不好（導致學習速度緩慢），那麼，教學進度就要放慢一點」。若將此思維過程以模糊規則表示，則為：

If x_1 is L and x_2 is S Then y is VS

其中，L 代表模糊語意「大」（Large），S 代表模糊語意「慢」（Slow），而 VS 則代表模糊語意「很慢」（Very Slow），可以分別用三個不同的模糊集合表示。

綜合上述，現假設在此模糊推論系統中，其完整的規則庫如下所示：

IF x_1 is X1.L and x_2 is X2.S THEN y is Y.VS
IF x_1 is X1.M and x_2 is X2.S THEN y is Y.VS
IF x_1 is X1.S and x_2 is X2.S THEN y is Y.S
IF x_1 is X1.L and x_2 is X2.M THEN y is Y.S
IF x_1 is X1.M and x_2 is X2.M THEN y is Y.M
IF x_1 is X1.S and x_2 is X2.M THEN y is Y.F
IF x_1 is X1.L and x_2 is X2.F THEN y is Y.F

IF x_1 is X1.M and x_2 is X2.F THEN y is Y.VF

IF x_1 is X1.S and x_2 is X2.F THEN y is Y.VF

此處為了區分不同變數所屬之模糊語意值，特地在模糊集合代號前加上 X1、X2. **和** Y. **等前導符號以資識別。各模糊語意之含意如下表所示：**

輸入模糊語意值	含意	輸出模糊語意值	含意
X1.L (Large)	(學習誤差) 大	Y.VS (Very Slow)	(課程進度) 很慢
X1.M (Medium)	(學習誤差) 中等	Y.S (Slow)	(課程進度) 慢
X1.S (Small)	(學習誤差) 小	Y.M (Medium)	(課程進度) 中等
X2.S (Slow)	(學習速度) 慢	Y.F (Fast)	(課程進度) 快
X2.M (Medium)	(學習速度) 中等	Y.VF (Very Fast)	(課程進度) 很快
X2.F (Fast)	(學習速度) 快		

表 1 **輸出入模糊語意值之含意**

接著，有了規則庫之後，為了順利讓推論引擎進行推論，必須在資料庫中定義上述規則中若干模糊語意，以完成完整的知識庫。

首先，定義輸出入變數 x_1, x_2, y **之論域皆為** [0, 1]**，這意味著所有評估指標必須量化至** 0 ~ 1 **之間。**

再者，定義 x_1 **之歸屬函數如圖** 6 **所示：**

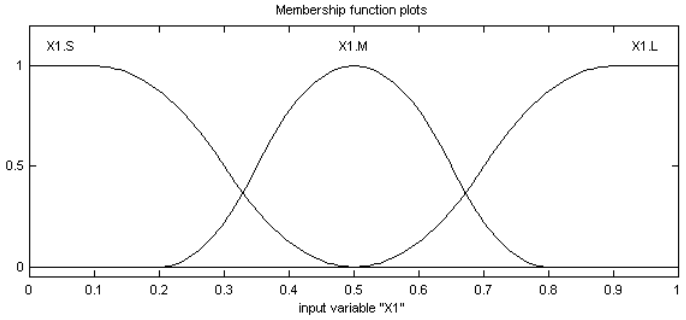


圖 6 **輸入變數** x1 **之歸屬函數**

若以方程式表示，則為：

$$X1.S = Z(x_1; 0.1, 0.5)$$
$$X1.M = p(x_1; 0.5, 0.6)$$
$$X1.L = S(x_1; 0.5, 0.9)$$

其次，定義 x_2 **之歸屬函數如圖** 7 **所示：**

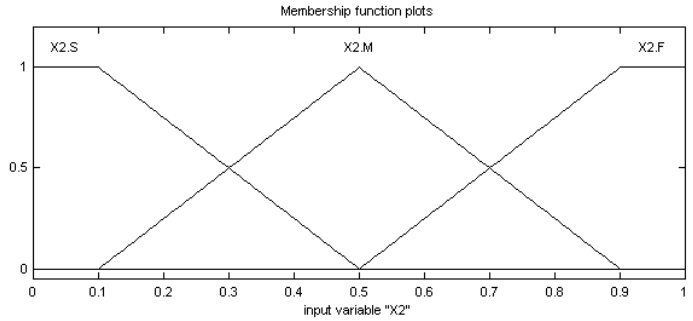


圖 7 輸入變數 x2 之歸屬函數

若以方程式表示，則為：

$$X2.S = Trap(x_2; 0, 0, 0.1, 0.5)$$
$$X2.M = Tri(x_2; 0.1, 0.5, 0.9)$$
$$X2.F = Trap(x_2; 0.5, 0.9, 1, 1)$$

最後，定義 *y* 之歸屬函數如圖 8 所示：

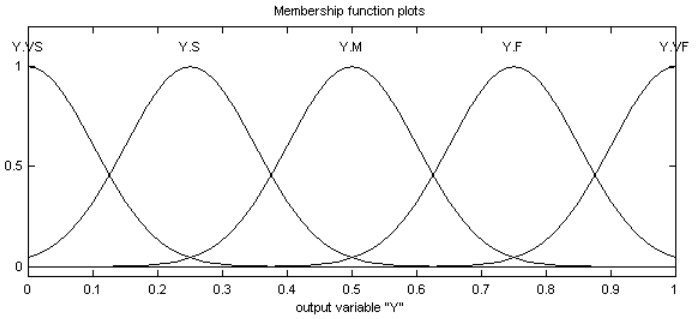


圖 8 輸出變數 y 之歸屬函數

若以方程式表示，則為：

$$Y.VS = G(y; 0, 0.1)$$
$$Y.S = G(y; 0.25, 0.1)$$
$$Y.M = G(y; 0.5, 0.1)$$
$$Y.F = G(y; 0.75, 0.1)$$
$$Y.VF = G(y; 1, 0.1)$$

基於上一節之 DTD 定義，以 XML 所建立之模糊知識庫如下所示：

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE knowledgebase SYSTEM "fuzzy.dtd">
<knowledgebase>
    <rulebase>
        <rule>
            <if op="and">
                <input name="x1" value="X1.L" />
                <input name="x2" value="X2.S" />
```

```
        </if>
        <then>
            <output name="y" value="Y.VS" />
        </then>
</rule>
<rule>
        <if op="and">
            <input name="x1" value="X1.M" />
            <input name="x2" value="X2.S" />
        </if>
        <then>
            <output name="y" value="Y.VS" />
        </then>
</rule>
<rule>
        <if op="and">
            <input name="x1" value="X1.S" />
            <input name="x2" value="X2.S" />
        </if>
        <then>
            <output name="y" value="Y. S" />
        </then>
</rule>
<rule>
        <if op="and">
            <input name="x1" value="X1.L" />
            <input name="x2" value="X2.M" />
        </if>
        <then>
            <output name="y" value="Y. S" />
        </then>
</rule>
<rule>
        <if op="and">
            <input name="x1" value="X1.M" />
            <input name="x2" value="X2.M" />
        </if>
        <then>
            <output name="y" value="Y. M" />
        </then>
</rule>
<rule>
        <if op="and">
            <input name="x1" value="X1.S" />
            <input name="x2" value="X2.M" />
        </if>
        <then>
            <output name="y" value="Y. F" />
        </then>
</rule>
```

```
        <rule>
            <if op="and">
                <input name="x1" value="X1.L" />
                <input name="x2" value="X2.F" />
            </if>
            <then>
                <output name="y" value="Y. F" />
            </then>
        </rule>
        <rule>
            <if op="and">
                <input name="x1" value="X1.M" />
                <input name="x2" value="X2.F" />
            </if>
            <then>
                <output name="y" value="Y.VF" />
            </then>
        </rule>
        <rule>
            <if op="and">
                <input name="x1" value="X1.S" />
                <input name="x2" value="X2.F" />
            </if>
            <then>
                <output name="y" value="Y.VF" />
            </then>
        </rule>
    </rulebase>

    <database>
        <inpDefs>
            <inpVar name="x1">
                <range min="0" max="1" />
                <mfDefs>
                    <mf name="X1.S">
                        <zmf>
                            <left>0.1</left>
                            <right>0.5</right>
                        </zmf>
                    </mf>
                    <mf name="X1.M">
                        <pimf>
                            <center>0.5</center>
                            <breadth>0.6</breadth>
                        </pimf>
                    </mf>
                    <mf name="X1.L">
                        <smf>
                            <left>0.5</left>
                            <right>0.9</right>
```

```
                    </smf>
                </mf>
            </mfDefs>
        </inpVar>
        <inpVar name="x2">
            <range min="0" max="1" />
            <mfDefs>
                <mf name="X2.S">
                    <trapmf>
                        <left>0</left>
                        <lshoulder>0</lshoulder>
                        <rshoulder>0.1</rshoulder>
                        <right>0.5</right>
                    </trapmf>
                </mf>
                <mf name="X2.M">
                    <trimf>
                        <left>0.1</left>
                        <center>0.5</center>
                        <right>0.9</right>
                    </trimf>
                </mf>
                <mf name="X2.F">
                    <trapmf>
                        <left>0.5</left>
                        <lshoulder>0.9</lshoulder>
                        <rshoulder>1</rshoulder>
                        <right>1</right>
                    </trapmf>
                </mf>
            </mfDefs>
        </inpVar>
    </inpDefs>
    <outDefs>
        <outVar name="y">
            <range min="0" max="1" />
            <mfDefs>
                <mf name="Y.VS">
                    <gaussmf>
                        <center>0</center>
                        <breadth>0.1</breadth>
                    </gaussmf>
                </mf>
                <mf name="Y.S">
                    <gaussmf>
                        <center>0.25</center>
                        <breadth>0.1</breadth>
                    </gaussmf>
                </mf>
                <mf name="Y.M">
```

```
                    <gaussmf>
                        <center>0.5</center>
                        <breadth>0.1</breadth>
                    </gaussmf>
                </mf>
                <mf name="Y.F">
                    <gaussmf>
                        <center>0.75</center>
                        <breadth>0.1</breadth>
                    </gaussmf>
                </mf>
                <mf name="Y.VF">
                    <gaussmf>
                        <center>1</center>
                        <breadth>0.1</breadth>
                    </gaussmf>
                </mf>
            </mfDefs>
        </outVar>
    </outDefs>
</database>
</knowledgebase>
```

接著我們使用微軟所發展的 MSXML **剖析器**（Parser）[23]**，來驗證這個** XML **模糊知識庫之完備性及有效性，結果如圖** 9 **所示。此一通過驗證之** XML **模糊知識庫便可在網路上傳遞，供分散各處的模糊推論引擎所使用。**

[23]**目前已有許多軟體廠商致力於開發** XML **剖析器，**MSXML **是由微軟所發展的有效性剖析器**（Validity Parser)**，可得自**<http://msdn.microsoft.com/downloads/webtechnology/XML/msxml.asp>。

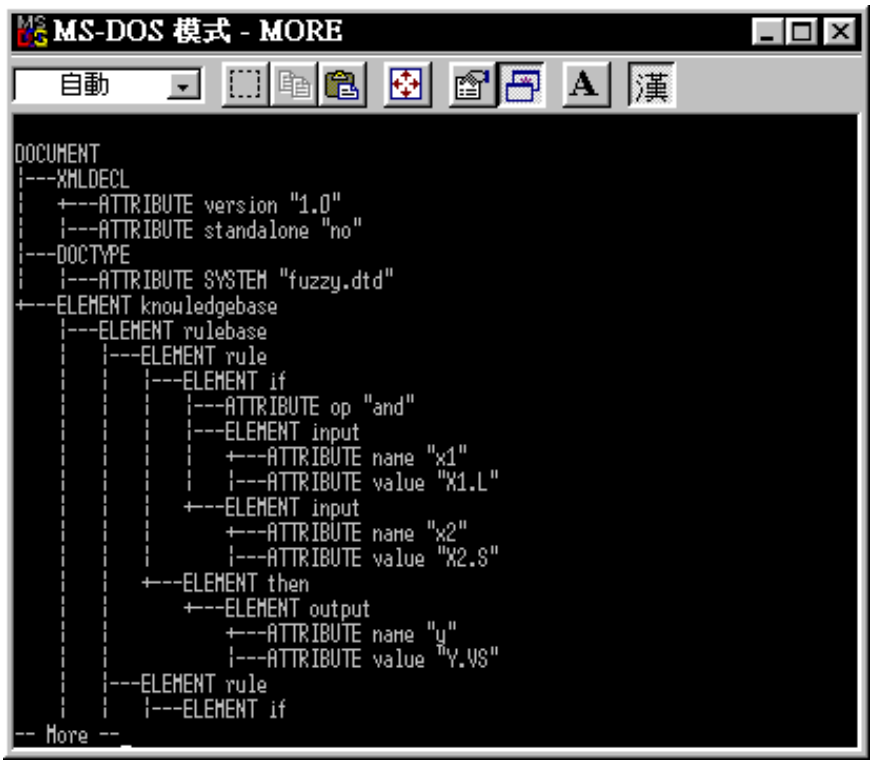


圖 9 使用 MSXML 剖析 XML 模糊知識庫

六、結論

本研究針對知識工程中之模糊推論系統，利用 Web 分散處理的特性，提出一個基於 XML 之分散式模糊知識庫管理模式，作為建構分散式智慧型系統的立論基礎。此外，藉由對模糊知識庫之結構分析，得知模糊知識庫是由模糊規則庫和模糊資料庫所組成；再深入剖析模糊規則庫和模糊資料庫之後，我們定義了一個適用於網路分散環境下的模糊知識庫 DTD，作為以 XML 建構分散式模糊知識庫之依據。

模糊推論系統中的推論引擎和知識擷取機制，是除了知識庫之外非常值得研究的主題，在本論文中並沒有針對這兩部份提出深入的結構分析，也沒有定義其相關的 DTD，此部份將作為未來進一步研究的重點，以期作為建構一個完整的分散式模糊推論系統之基石。

附錄、完整的模糊知識庫 DTD

```
<?xml version="1.0"?>
<!ELEMENT knowledgebase (rulebase, database)>
<!ELEMENT rulebase (rule+)>
<!ELEMENT rule (if, then)>
<!ELEMENT if (input+)>
  <!ATTLIST  if
                op  (and|or)  "and">
<!ELEMENT then (output)>
<!ELEMENT input EMPTY>
  <!ATTLIST  input
                name  CDATA #REQUIRED
                value  CDATA  #REQUIRED>
```

```
<!ELEMENT output EMPTY>
  <!ATTLIST  output
                name  CDATA  #REQUIRED
                value  CDATA  #REQUIRED>
<!ELEMENT database (inpDefs, outDefs)>
<!ELEMENT inpDefs (inpVar+)>
<!ELEMENT inpVar (range, mfDefs+)>
  <!ATTLIST  inpVar
                name  CDATA  #REQUIRED>
<!ELEMENT outDefs (outVar)>
<!ELEMENT outVar (range, mfDefs+)>
  <!ATTLIST  outVar
                name  CDATA  #REQUIRED>
<!ELEMENT range EMPTY>
  <!ATTLIST  range
                min  CDATA  #REQUIRED
                max  CDATA  #REQUIRED>
<!ELEMENT mfDefs (mf+)>
<!ELEMENT mf (zmf*|pimf*|smf*|trimf*|trapmf*|gaussmf*|pwmf*)>
  <!ATTLIST  mf
                name  CDATA  #REQUIRED>
<!ELEMENT zmf (left, right)>
<!ELEMENT pimf (center, breadth)>
<!ELEMENT smf (left, right)>
<!ELEMENT trimf (left, center, right)>
<!ELEMENT trapmf (left, lshoulder, rshoulder, right)>
<!ELEMENT gaussmf (center, breadth)>
<!ELEMENT pwmf (li+)>
<!ELEMENT left (#PCDATA)>
<!ELEMENT center (#PCDATA)>
<!ELEMENT right (#PCDATA)>
<!ELEMENT lshoulder (#PCDATA)>
<!ELEMENT rshoulder (#PCDATA)>
<!ELEMENT breadth (#PCDATA)>
<!ELEMENT li EMPTY>
  <!ATTLIST  li
                member  CDATA  #REQUIRED
                grade  CDATA  #REQUIRED>
```