การประยุกต์การโปรแกรมตรรกะเชิงอุปนัย เซตวิภัชนัย และข่ายงานประสาทประดิษฐ์ ในการรู้จำตัวพิมพ์อักษรไทย

นาย จุณณ์ ศรีสุธาพรรณ

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2544
ISBN 974-03-0951-8

AN APPLICATION OF INDUCTIVE LOGIC PROGRAMMING, FUZZY SET, AND ARTIFICIAL NEURAL NETWORKS TO THAI PRINTED CHARACTER RECOGNITION

Mr. Jun Srisutapan

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science in Computer Science
Department of Computer Engineering
Faculty of Engineering
Chulalongkorn University
Academic Year 2001
ISBN 974-03-0951-8

Thesis Title	An Application of Inductive Logic Programming, Fuzzy set, and Artificial Neural Networks to Thai Printed Character Recognition
By	Mr. Jun Srisutapan
Field of study	Computer Science
Thesis Advisor	Assistant Professor Boonserm Kijsirikul, Ph.D.

Accepted by the Faculty of Engineering, Chulalongkorn University in Partial Fulfillment of the Requirements for the Master 's Degree

.... Dean of Faculty of Engineering
(Professor Somsak Punyakeow, D.Eng.)

THESIS COMMITTEE

.... Chairman
(Associate Professor Prabhas Chongstitvatana, Ph.D.)

.... Thesis Advisor
(Assistant Professor Boonserm Kijsirikul, Ph.D.)

.... Member
(Suebskul Phiphobmongkol, Ph.D.)

.... Member
(Associate Professor Somchai Prasitjutrakul, Ph.D.)

จุณณ์ ศรีสุธาพรรณ : การประยุกต์การโปรแกรมตรรกะเชิงอุปนัย เซตวิภัชนัย และข่ายงานประสาทประดิษฐ์ ในการรู้จำตัวพิมพ์อักษรไทย. (AN APPLICATION OF INDUCTIVE LOGIC PROGRAMMING, FUZZY SET, AND ARTIFICIAL NEURAL NETWORKS TO THAI PRINTED CHARACTER RECOGNITION) อ. ที่ปรึกษา : ผศ. ดร. บุญเสริม กิจศิริกุล, 50 หน้า. ISBN 974-03-0951-8.

วิทยานิพนธ์นี้เสนอการประยุกต์ใช้ทฤษฎีเซตวิภัชนัย(เอฟเอสที)เพื่อการปรับปรุงประมาณกฎของการโปรแกรมตรรกะเชิงอุปนัย(ไอแอลพี)ซึ่งใช้ข่ายงานประสาทประดิษฐ์(บีเอ็นเอ็น) ด้วยการช่วยเหลือของเอฟเอสที การประมาณค่าความจริงของโปรแกรมตรรกะมีความสมเหตุสมผลมากขึ้นก่อนที่ค่าความจริงเหล่านั้นจะถูกส่งให้กับบีเอ็นเอ็นเพื่อทำการเรียนรู้หรือเพื่อทำการรู้จำ โดยเฉลี่ยแล้วจากหลายๆ การทดลอง อัตราการรู้จำโดยใช้ไอแอลพีอย่างเดียวมีค่าเป็น 83.53% และได้ 88.39% โดยการใช้ไอแอลพีร่วมกับบีเอ็นเอ็น ส่วนวิธีการที่เรานำเสนอนั้นให้ผลการรู้จำดีที่สุดเท่ากับ 90.40%

ภาควิชา วิศวกรรมคอมพิวเตอร์	ลายมือชื่อนิสิต
สาขาวิชา วิทยาศาสตร์คอมพิวเตอร์	ลายมือชื่ออาจารย์ที่ปรึกษา
ปีการศึกษา 2544	ลายมือชื่ออาจารย์ที่ปรึกษาร่วม

4170505321 : MAJOR COMPUTER SCIENCE
KEY WORD: SOFT COMPUTING / INDUCTIVE LOGIC PROGRAMMING / NEURAL NETWORKS / FUZZY LOGIC / ARTIFICIAL INTELLIGENCE

JUN SRISUTAPAN : AN APPLICATION OF INDUCTIVE LOGIC PROGRAMMING, FUZZY SET, AND ARTIFICIAL NEURAL NETWORKS TO THAI PRINTED CHARACTER RECOGNITION. THESIS ADVISOR : ASST. PROF. BOONSERM KIJSIRIKUL, Ph.D., 50 pp. ISBN 974-03-0951-8.

This thesis presents an application of Fuzzy set theory (FST) for improving Backpropagation Neural Network (BNN) based Inductive Logic Programming (ILP) rule approximation. With the help of FST, the approximation of the truth values of logic programs is more reasonable, before the values are sent to the BNN for learning or for recognising. Experimental results show that the recognition accuracies are in average 83.53% and 88.39% for ILP alone and ILP&BNN, respectively. Our proposed method gives the best recognition accuracy of 90.40%.

Department Computer Engineering	Student's signature
Field of study Computer Science	Advisor's signature
Academic year 2001	Co-advisor's signature

ACKNOWLEDGEMENTS

This thesis is dedicated to my family and to all whom I have learned. Moreover, I would like to pay gratitude to my thesis advisor, Asst. Prof. Boonserm Kijsirikul, PhD., for his insight and understanding both technically and personally throughout the period of this thesis study. In addition, I would like to thank to the thesis committee for their reviewing this thesis.

Special thanks to Prof. Chidchanok Lursinsap for comprehensive lectures on both soft and hard computing plus a big warmfully suggestions even if he was busy. My thank also go to ajarn Nuttakorn Thubthong at the Department of Physics for many corrections on manuscript. In final, thanks to the members of Machine Intelligence & Knowledge Discovery Laboratory (MIND-LAB) and the members of Digital System Laboratory (DS-LAB), who supplied me with technical information and nice recreations.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 PROBLEM IDENTIFICATION

For the last ten years, many techniques have been developed to improve recognition accuracy of *Thai–Optical Character Recognition (Thai-OCR)* which is not reasonably high enough so far, especially in the commercial software according to the complexity nature of Thai characters plus the high similarity between some characters, e.g., the characters ฎ, ฏ, ท or ฑ.

[Supanwansa, 2540] has proposed an interesting method that employed *Inductive Logic Programming (ILP)* [Muggleton, 1995] which is a kind of *Machine Learning* [Mitchell, 1997] to generate a rule for recognising characters. The benefit of this technique is that rules produced by ILP can accurately classify characters, and are described in form of *First-Order Predicate Logic* [Mendelson, 1987] which is easy for humans to analysis, to understand or even to modify to meet their requirements. In addition, background knowledge can be recursive, i.e., they are suitable to describe some complex real world data such as Thai characters.

However, for some noisy or unseen character, there can be no rule that perfectly matches the character. In this case, we say that this character is *not ILP-recognisable.* [Kijsirikul and Sinthupinyo, 1999] solved this ILP-unrecognisable problem by sending the binary truth values of literals of the rule to a *Backpropagation Neural Network (BNN).* After BNN is applied, every character is recognisable, and thus, the more correctly recognised characters are obtained.

There is some disadvantage we observed in the line of the above work. The truth values used in this work are quite sharp, i.e., they are possibility values of either 0 or 1. The value 1 is used to show that a given character exactly has some property; otherwise, the value 0 is used. Therefore, the value 0 is used to show that a given character partially has some property or exactly has no some property.

In case that the character partially has the property, we argued that we should not use the value 0 which is not reasonable in some sense, at least, our sense; we should use some *softer* value between 0 and 1 instead, such as 0.9, 0.1, 0.5 or 0.001, respected to the *nature of Thai characters*. These truth values are called *fuzzy* truth values and they can be obtained by using Fuzzy set theory(FST). We believe that such softer truth values can improve the classification accuracy of the Thai-OCR. That is why this research has been emerged.

1.2 OBJECTIVE OF THE RESEARCH

The main objective of this research is to develop a program that finds the fuzzy truth value of background knowledge in [Supanwansa, 2540].

1.3 SCOPE OF THE RESEARCH

1. In this research, we will use fuzzy set theory(FST) to find the truth values of background knowledge in [Supanwansa, 2540].
2. We will compare the results to [Kijsirikul and Sinthupinyo, 1999] and [Supanwansa, 2540], using the same environment.
3. This research will be developed and tested on any suitable operating system and development tools.

1.4 DETAIL SCHEDULES

The detail schedules of this research are the following:

1. Search and study previous works about BNN and ILP.
2. Search and study previous works about FST.
3. Write a program that evaluates truth values of background knowledge and sends them to the BNN.
4. Repeat steps 1-3 as required.
5. Give the conclusion.

1.5 EXPECTED OUTCOME

The usefulness of this work is that the recognition accuracy of the Thai-OCR using our method should be higher than the one using the method of [Kijsirikul and Sinthupinyo, 1999].

1.6 RELATED PUBLICATIONS

This research is accepted as the full paper entitled "Improving Backpropagation Neural Network Based ILP Rule Using Fuzzy Set Theory." in the Fifth National Conference on Computer Science and Engineering (NCSEC2000), Bangkok, Thailand, 2000.

1.7 ORGANISATION

This thesis is organised as follows. First, Chapter 2 gives an overview of ILP&BNN methods. Chapter 3 describes theory of fuzzy set. Chapter 4 discusses the approximation of truth values of the background knowledge. Chapter 5 gives the experiments. Finally, the conclusion of this thesis will be given in Chapter 6.

CHAPTER 2
TRADITIONAL METHOD

In this chapter, we will give an overview of the traditional method of combining ILP with BNN. Section 2.1 describes how to train the BNN to learn the characters. Section 2.2 describes how to recognise characters using the BNN.

2.1 TRAIN THE BNN TO LEARN THE CHARACTERS

For convenience, suppose that there are only three Thai characters, i.e., 'Kai'('ก'), 'Khuad'('ฃ') and 'Tahan'('ฑ'). After ILP is applied, we obtain a *set of character rules* (or *rule set*) that is used to recognise characters. Figure 2.1 shows an example of a rule set generated by ILP.

```
Kai(I):-      HeadZone(I, 3),
              HeadPrimitive(I, 1).

Khuad(I):-    HeadZone(I, 2),
              HeadPrimitive(I, 10),
              EndPointZone(I, 4),
              CountEndPoints(I, 2).

Tahan(I):-    not TopRightTail(I),
              HeadZone(I, 2),
              HeadPrimitive(I, 12),
              EndPointPrimitive(I, 5),
              CountEndPoints(I, 3).
```

Figure 2.1: An example of a rule set generated by ILP.

There are three rules in this rule set, each of which defines the concept of character 'Kai', 'Khuad' or 'Tahan', respectively.

Consider the *predicates* `HeadZone(I,3)` and `HeadPrimitive(I, 1)`. Predicates are statements involving variables. They are neither true nor false when the values of the variables are not specified. Predicates will become *propositions* when all variables in the predicates (in this case, variable *I*) are substituted by some objects in the domain.

These predicates are characteristics of character 'Kai'. `HeadZone(I, 3)` characterises that the head zone of character 'Kai' must be of type 3 (at the bottom left). (See [Supanwansa, 2540] for details). `HeadPrimitive(I, 1)` characterises that the head of character 'Kai' must be of type 1 (in octant 2). Such predicates are called *background knowledge* that must be provided to the ILP. The construction details of background knowledge are described in [Supanwansa, 2540].

A character *X* is represented by the information I_X (data that form the character *X*); *e.g.* I_X = (3, 0.78, [781, 970, 82, 83, 339], [], [], [1]). *X* will be recognised as 'Kai' if when all occurrences of `I` in the rule set are substituted with I_X, the truth values of **all** *literals* (predicates or negation of predicates) in the rule Kai are 1.

Logically, a given character *Y* which is represented by the information I_Y, will be not recognised as 'Khuad' if when all occurrences of `I` in the rule set are substituted with I_Y, the truth value of **some** literal in the rule Khuad is 0.

In order to train the BNN to learn a character of font 'Kai', e.g. *TrainKai* which is represented by the information $I_{TrainKai}$ = (3, 0.78, [781, 970, 82, 83, 339], [], [], [1]). Firstly, we substitute all occurrences of `I` in the rule set with $I_{TrainKai}$ and then compute the truth value of every literal in the rule set. Suppose such the truth values (0's or 1's) are shown on the right hand side of each literal in Figure 2.2 below:

Kai ($I_{TrainKai}$) :-	*HeadZone* ($I_{TrainKai}$, 3),	$\rightarrow$	*1*
	HeadPrimitive ($I_{TrainKai}$, 1).	$\rightarrow$	*1*
Khuad ($I_{TrainKai}$) :-	*HeadZone* ($I_{TrainKai}$, 2),	$\rightarrow$	*0*
	HeadPrimitive ($I_{TrainKai}$, 10),	$\rightarrow$	*0*
	EndPointZone ($I_{TrainKai}$, 4),	$\rightarrow$	*1*
	CountEndPoints ($I_{TrainKai}$, 2).	$\rightarrow$	*0*
Tahan ($I_{TrainKai}$) :-	*not TopRightTail* ($I_{TrainKai}$),	$\rightarrow$	*1*
	HeadZone ($I_{TrainKai}$, 2),	$\rightarrow$	*0*
	HeadPrimitive ($I_{TrainKai}$, 12),	$\rightarrow$	*0*
	EndPointPrimitive ($I_{TrainKai}$, 5),	$\rightarrow$	*0*
	CountEndPoints ($I_{TrainKai}$, 3).	$\rightarrow$	*0*

Figure 2.2: The truth values of all literals when $I_{TrainKai}$ is substituted into the rule set.

Thus, we use the vector $[1\ 1\ 0\ 0\ 1\ 0\ 1\ 0\ 0\ 0\ 0]^T$ for training the BNN (see Figure 2.3). We repeat this step for all training characters.

The links of the BNN are fully connected from the hidden layer to the output layer. The number of nodes in the hidden layer equals to the number of rules, which is three. All input nodes corresponding to predicates in the same rule are connected to one hidden node that represents that rule.

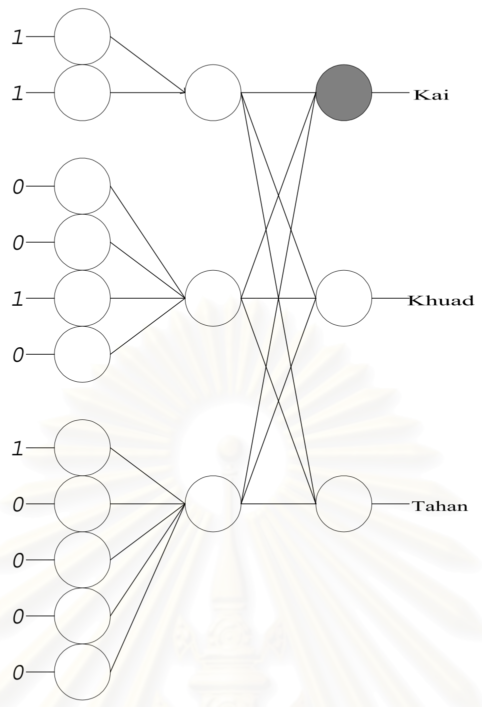

Figure 2.3: Training the BNN with *TrainKai*.

2.2 USE THE BNN TO RECOGNISE CHARACTERS

For a given noisy character of font 'Khuad', e.g. *NoisyKhuad* which is represented by the information $I_{NoisyKhuad}$. Suppose that *NoisyKhuad* is not *ILP-recognisable* (no rule that completely matches this character), as shown in Figure 2.4.

Kai($I_{NoisyKhuad}$) :- *HeadZone*($I_{NoisyKhuad}$, 3), $\rightarrow$ 0
HeadPrimitive($I_{NoisyKhuad}$, 1). $\rightarrow$ 0

Khuad($I_{NoisyKhuad}$) :- *HeadZone*($I_{NoisyKhuad}$, 2), $\rightarrow$ 1
HeadPrimitive($I_{NoisyKhuad}$, 10), $\rightarrow$ 1
EndPointZone($I_{NoisyKhuad}$, 4), $\rightarrow$ 1
CountEndPoints($I_{NoisyKhuad}$, 2). $\rightarrow$ 0

Tahan($I_{NoisyKhuad}$) :- *not TopRightTail*($I_{NoisyKhuad}$), $\rightarrow$ 1
HeadZone($I_{NoisyKhuad}$, 2), $\rightarrow$ 1
HeadPrimitive($I_{NoisyKhuad}$, 12), $\rightarrow$ 0
EndPointPrimitive($I_{NoisyKhuad}$, 5), $\rightarrow$ 0
CountEndPoints($I_{NoisyKhuad}$, 3). $\rightarrow$ 0

Figure 2.4: *NoisyKhuad* is not ILP-recognisable.

Analogous to Section 2.1, we send the vector `[0 0 1 1 1 0 1 1 0 0 0]`^{T} to the BNN. In this case, the prediction from the BNN (the best matching character to *NoisyKhuad*) is 'Khuad', which is correct. (see Figure 2.5)

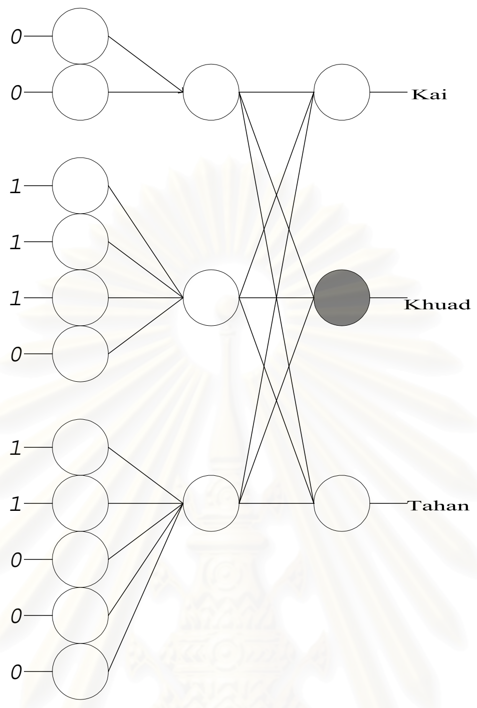

Figure 2.5: *NoisyKhuad* is correctly recognised by BNN.

However, there is some disadvantage of this method. Let us consider the proposition `CountEndPoints(I, 5)`. The truth value of this proposition will be set to the value 1 if *I* contains exactly five endpoints; otherwise, it will be set to the value 0. Suppose *I* contains four endpoints, then the truth value the proposition is set to the value 0, this is not reasonable in some sense.

Therefore, the hypothesis of this research is that the more reasonable truth values may improve the recognition accuracy of the BNN. We will use theory of fuzzy set to find such the values. Theory of fuzzy set will be introduced in the next chapter.

CHAPTER 3
THEORY OF FUZZY SET

In this chapter, we will introduce some basic concepts of fuzzy set theory. Section 3.1 gives the literature surveys on fuzzy set theory. Section 3.2 explains the grand concept of the fuzzy set. Section 3.3 describes the concept of the fuzzy relation. Section 3.4 gives an idea of the fuzzy number. Finally, Section 3.5 describes operations on fuzzy sets.

3.1 LITERATURE SURVEYS ON FUZZY SET THEORY

Fuzzy set theory (FST) is introduced by L.A. Zadeh of the University of California at Berkeley [Zadeh, 1965]. Fuzzy set theory is a mathematical theory dealing with vagueness and uncertainty. FST methodology has proved its soundness and usefulness in many applications, such as Operation Researches, Control Engineering, Economics, Database, Image Recognition, Image Clustering, Psychology and other areas. Below are some applications of FST.

- [Karwowski et. al., 1989] employed FST to improve text editor that can predict user input and change its environment for more user comfortable, e.g., scroll the screen or press function key automatically. The nonfuzzy method predicted correctly 47% while the fuzzy method predicted correctly 76%.

- [Ushida et. al, 1993] employed the conceptual fuzzy set with another techniques to recognise human emotions using facial expressions.

- [Wang et. al, 1997] employed FST in the induction learning process of the Identification Tree and tested the method in the problem of selecting sports under a given weather conditions.

- [Martienne and Quafafou, 1998] built an ILP system namely EAGLE which employed the fuzzy set to partition the input data and then used Rough set theory [Pawlak, 1982] in the inductive learning process. EAGLE was tested on real-world organic chemistry problems.

3.2 FUZZY SET

- **CRISP SET**

In classical(crisp) set theory, the sets are defined as collections of objects having some property; nothing special is considered about the nature of the individual objects.

Formally, let X denote the universal set which contains all possible elements concerning in each particular context. We can define sets within a given universal set X by a *characteristic function* which declares wheter an elements x of X are members of the set or not. Set A is defined by its characteristic function, $A : X \rightarrow \{0, 1\}$ as follow:

$$A(x) = \begin{cases} 1; & x \in A \\ 0; & x \notin A \end{cases} \qquad \text{...............(3.1)}$$

Example 3.1

For a universal set $X = \{1, ..., 10\}$. We define a set *CRISP-FIVE* by using characteristic function *CRISP-FIVE* : $X \rightarrow \{0, 1\}$ as shown in Figure 3.1

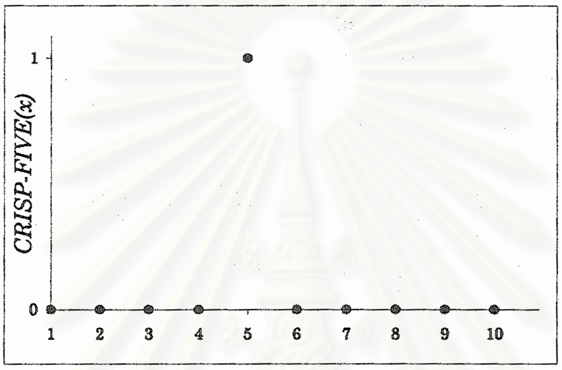

Figure 3.1: Characteristic function *CRISP-FIVE*.

- **FUZZY SET**

Fuzzy set theory is a *generalisation* of classical crisp set theory. In other word, definitions, theorems and proofs of fuzzy set theory always hold for classical crisp set theory. Therefore, fuzzy set theory has a wider scope of applicability than classical crisp set theory in solving problems.

Definition 3.1 (Fuzzy set) *Given the universal set X, in order to define a fuzzy set namely A on X, we define a membership function that maps elements x of X into real numbers in [0, 1]. This membership function is denoted by the same symbol A, that is*

$$A : X \rightarrow [0, 1] \qquad \text{...............(3.2)}$$

We can write fuzzy set A as

$$A = \{ (x, A(x)) \mid x \in X \} \qquad \text{...............(3.3)}$$

Here, A(x) is interpreted as the degree to which x belongs to the fuzzy set A. □

Example 3.2

We define a fuzzy set *FIVE1* = { (x, *FIVE1*(x)) | $x \in [1, 10]$ }, where

$$FIVE1(x) = \frac{1}{1+(x-5)^2}; \; x \in [1, 10] \qquad \text{...............(3.4)}$$

and define another fuzzy set $FIVE2 = \{ (x, FIVE2(x)) \mid x \in [1, 10] \}$, where

$$FIVE2(x) = e^{-|x-5|}; \quad x \in [1, 10] \qquad \text{...............(3.5)}$$

Both membership functions are graphically shown in Figure 3.2.

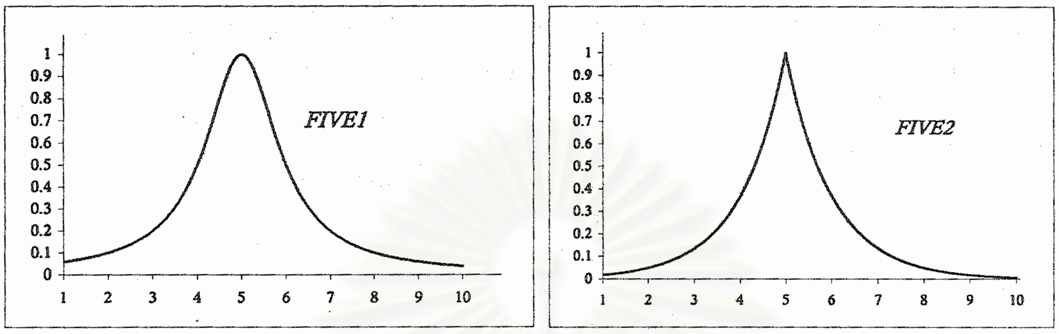

Figure 3.2: Membership functions *FIVE1* and *FIVE2*.

Note that fuzzy sets representing the same concept (in this case, number five) may vary considerably.

3.3 FUZZY RELATION

Definition 3.2 *An n-ary classical(crisp) relation among $A_1, \ldots, A_n$ is a subset of Cartesian product $\prod_{i=1}^{n} A_i$ while an n-ary fuzzy relation among $A_1, \ldots, A_n$ is a fuzzy set defined on $\prod_{i=1}^{n} A_i$.* □

In case of $n = 2$, crisp binary relation indicates whether a given two objects are related or not while fuzzy relation indicates the degree they are related.

Definition 3.3 *Given the universal set X, a crisp binary relation R on X that is reflexive, symmetric and transitive is called an (crisp) equivalence relation.* □

Definition 3.4 *For a fuzzy binary relation R on X,*

R is reflexive if

$$R(x, x) = 1; \quad x \in X \qquad \text{...............(3.6)}$$

R is symmetric if

$$R(x, y) = R(y, x); \quad x, y \in X \qquad \text{...............(3.7)}$$

R is transitive if

$$R(x, y) \geq \max_{t \in X} \min[R(x,t), R(t,y)]; \quad x, y \in X \qquad \text{...............(3.8)}$$

A fuzzy binary relation R on X that is reflexive, symmetric and transitive is called an fuzzy equivalence relation (or similarity relation). □

Example 3.3

Given a universal set X = {'ก', 'ถ', 'ภ', 'ฮ'}, consider a binary crisp equivalence relation C and a binary fuzzy equivalence relation R shown in Table 3.1 and 3.2, respectively.

Table 3.1: Crisp equivalence relation C.

	'ก'	'ถ'	'ภ'	'ฮ'
'ก'	1	0	0	0
'ถ'	0	1	0	0
'ภ'	0	0	1	0
'ฮ'	0	0	0	1

Table 3.2: Fuzzy equivalence relation R.

	'ก'	'ถ'	'ภ'	'ฮ'
'ก'	1	.9	.9	0
'ถ'	.9	1	.81	0
'ภ'	.9	.81	1	0
'ฮ'	0	0	0	1

It is easy to show that R is reflexive and symmetric. To verify that R is transitive, substitute x with 'ก' and y with 'ภ', so R('ก', 'ภ') = .9, then

$$\begin{aligned}\max_{t\in X}\min[R(\text{'ก'}, t), R(t, \text{'ภ'})] &= \max[\min(R(\text{'ก'}, \text{'ก'}), R(\text{'ก'}, \text{'ภ'})),\\ &\qquad \min(R(\text{'ก'}, \text{'ถ'}), R(\text{'ถ'}, \text{'ภ'})),\\ &\qquad \min(R(\text{'ก'}, \text{'ภ'}), R(\text{'ภ'}, \text{'ภ'})),\\ &\qquad \min(R(\text{'ก'}, \text{'ฮ'}), R(\text{'ฮ'}, \text{'ภ'}))\ ]\\ &= \max[\min(1, .9), \min(.9, .81), \min(.9, .81), \min(0, 0)]\\ &= \max[.9, .81, .81, 0]\\ &= 0.9,\ \text{which satisfies condition (3.8).}\end{aligned}$$

This example shows how fuzzy set theory gives us more expressive power than the classical crisp set theory. Let us go to the next basic concept of fuzzy set theory.

3.4 FUZZY NUMBER

Fuzzy number is basic concept of *fuzzy quantifiers*, which are used to evaluate the truth values of quantified fuzzy propositions in Chapter 4.

Definition 3.5 *A fuzzy number, A is a fuzzy set defined on $\Re$ (set of real numbers) and A must capture the intuitive conception of a set of numbers such as follow:*

$$A(x) = \begin{cases} 0; & x < a \\ f(x); & x \in [a,b] \\ 1; & x \in [b,c] \\ g(x); & x \in [c,d] \\ 0; & x > d \end{cases} \qquad \text{..............(3.9)}$$

where $a \le b \le c \le d$ and f is a continuous function that increases to 1 at point b, and g is a continuous function that decreases from 1 at point c. □

It is easy to check that *CRISP-FIVE*, *FIVE1* and *FIVE2* are fuzzy numbers. Let us go to the next important basic concept of fuzzy set theory.

3.5 OPERATIONS ON FUZZY SETS

Operations on fuzzy sets are basic concept of truth value evaluation of fuzzy propositions in Chapter 4. There are three main types of operations: complement, intersection and union.

3.5.1 FUZZY COMPLEMENTS

Definition 3.6 *Let A be a fuzzy set on X. The fuzzy complement of A, is defined by*

$$\overline{A} = \{ (x, c(A(x))) \mid x \in X\} \qquad \text{...............(3.10)}$$

where c is a fuzzy complement function, which is any function $c : [0, 1] \rightarrow [0, 1]$ that satisfies the following axiomatic skeleton for fuzzy complement functions:

Axiom c1.	*$c(0) = 1$ and $c(1) = 0$*	*(crisp boundary conditions)*
Axiom c2.	*$x \le y$ implies $c(x) \ge c(y)$*	*(monotonicity)*

The following axioms are optional:

Axiom c3.	*c is a continuous function*	*(continuity)*
Axiom c4.	*$\forall x \in [0, 1]$: $c(c(x)) = x$*	*(involution)* □

Here are some fuzzy complement functions:

- Standard:

$$c(x) = 1 - x \qquad \text{...............(3.11)}$$

- Cosine:

$$c(x) = \frac{1+\cos(\pi x)}{2} \qquad \text{...............(3.12)}$$

- Sugeno's:

$$c_\lambda(x) = \frac{1-x}{1+\lambda x};\quad \lambda \in (-1,\infty) \qquad \text{...............(3.13)}$$

- Yager's:

$$c_w(x) = (1-x^w)^{1/w}; w \in (0,\infty) \qquad \text{...............(3.14)}$$

3.5.2 FUZZY INTERSECTION

Definition 3.7 *Let A, B be two fuzzy sets on X. The fuzzy intersection of A and B, is defined by*

$$A \cap B = \{ (x, i(A(x), B(x)) \mid x \in X \} \qquad \text{...............(3.15)}$$

where i is a fuzzy intersection function, which is any function i : [0, 1]×[0, 1]→[0, 1] that satisfies the following axiomatic skeleton for fuzzy intersection functions:

Axiom i1.	$i(x, 1) = x$	*(crisp boundary condition)*
Axiom i2.	$y \le y'$ *implies* $i(x, y) \le i(x, y')$	*(monotonicity)*
Axiom i3.	$i(x, y) = i(y, x)$	*(commutativity)*
Axiom i4.	$i(x, i(y, z)) = i(i(x, y), z)$	*(associatively)*

The following axioms are optional:

Axiom i5.	*i is a continuous function*	*(continuity)*
Axiom i6.	$i(x, x) < x$	*(subidempotency)*
Axiom i7.	$x < x'$ *and* $y < y'$ *implies* $i(x, y) < i(x', y')$	*(strict monotonicity)* □

Here are some fuzzy intersection functions:

- Standard:

$$i_{max}(x, y) = \min(x, y) \qquad \text{...............(3.16)}$$

- Algebraic product:

$$i_{ap}(x, y) = xy \qquad \text{...............(3.17)}$$

- Bounded difference:

$$i_{bd} = \max(0, x+y-1) \qquad \text{...............(3.18)}$$

- Drastic:

$$i_{min}(x, y) = \begin{cases} x; & y = 1 \\ y; & x = 1 \\ 0; & otherwise \end{cases} \qquad \text{...............(3.19)}$$

- Yager's:

$$i_w(x, y) = 1 - \min\left(1, \left[(1-x)^w + (1-y)^w\right]^{1/w}\right); \quad w > 0 \qquad \text{............... (3.20)}$$

3.5.3 FUZZY UNION

Definition 3.8 *Let A, B be two fuzzy sets on X. The fuzzy union of A and B, is defined by*

$$A \cup B = \{(x, u(A(x), B(x)) \mid x \in X\} \qquad \text{...............(3.21)}$$

where u is a fuzzy union function, which is any function u : [0, 1]×[0, 1]→[0, 1] that satisfies the following axiomatic skeleton for fuzzy union functions:

Axiom u1.	$u(x, 0) = x$	*(crisp boundary condition)*
Axiom u2.	$y \le y'$ *implies* $u(x, y) \le u(x, y')$	*(monotonicity)*
Axiom u3.	$u(x, y) = u(y, x)$	*(commutativity)*
Axiom u4	$u(x, u(y, z)) = u(u(x, y), z)$	*(associatively)*

The following axioms are optional:

Axiom u5.	*u is a continuous function*	*(continuity)*
Axiom u6.	$u(x, x) > x$	*(superidempotency)*
Axiom u7.	$x < x'$ *and* $y < y'$ *implies* $i(x, y) < i(x', y')$	*(strict monotonicity)*

□

Here are some fuzzy union functions:

- Standard:

$$u_{min}(x, y) = \max(x, y) \qquad \text{...............(3.22)}$$

- Algebraic sum:

$$u_{as}(x, y) = x + y - xy \qquad \text{...............(3.23)}$$

- Bounded sum:

$$u_{bs}(x, y) = \min(1, x+y) \qquad \text{...............(3.24)}$$

- Drastic:

$$u_{max}(x, y) = \begin{cases} x; & y = 0 \\ y; & x = 0 \\ 1; & otherwise \end{cases} \qquad \text{...............(3.25)}$$

- Yager's:

$$u_w(x,y) = \min\left(1, \left[x^w + y^w\right]^{1/w}\right); \quad w > 0 \qquad \text{...............(3.26)}$$

Definition 3.9 *A fuzzy set A defined on X is of type-1 if its membership function A is a mapping from X to [0, 1]. A is of type-k for k = 2, 3, ... if A is a mapping from X to the set of fuzzy set of type-(k-1). For simplicity, it will always be understood that A is of type-1 if it is not specified to be of a higher type.* □

Note that the more comprehensive theoretical treatments on the theory can be found in [Zadeh 1965, 1966, 1968, 1975, 1978, 1988, 1989 and 1996], [Klir and Yuan, 1995], [Kandel, 1986], [Lin and Lee, 1996] and [Yen, 1999]. [Klir, Clair and Yuan, 1997] is well suited for the beginners.

CHAPTER 4
TRUTH VALUES APPROXIMATION OF BACKGROUND KNOWLEDGE

In this chapter, we will explain the truth values evaluation procedures of fuzzy propositions and introduce all fuzzy sets used in experiments in Chapter5. All fuzzy propositions in our research can be divided into three basic types: atomic, compound and quantified [Zadeh, 1988]. Each type will be described in Sections 4.1, 4.2 and 4.3, respectively. Finally, Section 4.4 gives detail about a truth transformation technique.

4.1 ATOMIC FUZZY PROPOSITION

Definition 4.1 *An atomic fuzzy proposition is a sentence of the form*

$$p \equiv \text{"}x \text{ is } A\text{"} \qquad \text{..........(4.1)}$$

where "A" is some concept such as young, old, large or left-bottom while x refers to an object in the domain. The truth value of p, which is denoted by T(p), is defined to be the membership degree to which x belongs to the fuzzy set that is used to represent concept "A", that is,

$$T(p) = T(\text{"}x \text{ is } A\text{"}) = A(x) \qquad \text{..........(4.2)}$$

□

Example 4.1

Consider the fuzzy sets *FIVE1* and *FIVE2* in Chapter 3. Let p be the proposition "4 is around 5".

- If we use fuzzy set *FIVE1* to represent the concept "around 5", then $T(p) = FIVE1(4) = \dfrac{1}{1+(4-5)^2} = 0.5$ (see Figure 4.1 left).

- If we use fuzzy set *FIVE2* to represent the same concept, then $T(p) = FIVE2(4) = e^{-|4-5|} = 0.3678$ (see Figure 4.1 right).

Generally, in this research, we define a fuzzy set namely *FuzzyAround:k:x₀:* to represent the concept "around x_0" as

$$FuzzyAround : k : x_0 := \{\, (x,\ e^{-k|x-x_0|}) \mid x \in \Re \,\} \qquad \text{..........(4.3)}$$

where k is a constant factor of the exponential function. Many propositions in our background knowledge employ this fuzzy set with their own k. It is easy to check that *FuzzyAround:k:x₀* is a fuzzy number.

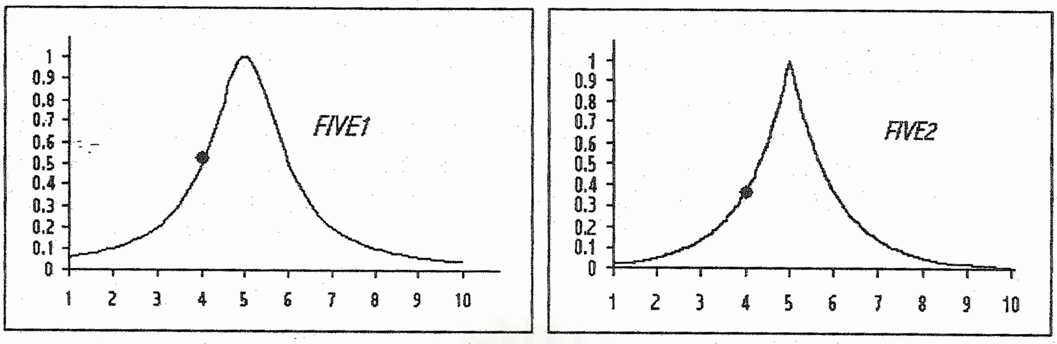

Figure 4.1: Truth values of the proposition "4 is around 5" using *FIVE1* and *FIVE2*.

Example 4.2

In [Supanwansa, 2540], each character image is composed of *primitive vectors*. There are two types of primitive vectors: line vectors and circle vectors. (see Figure 4.2).

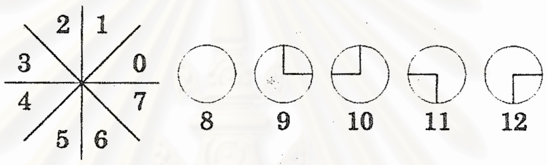

Figure 4.2: Primitive Vectors

The primitive vectors of type 0 to type 7 are used to represent lines while the primitive vectors of type 8 to type 12 are used to represent circles. The primitive vectors of type 0 are the lines whose its angle lie in octant 1, and so on. The primitive vector of type 8 is the circle that does not connect to any line. The primitive vector of type 9 is the circle that connects to a line at quadrant 1, and so on.

Next, the similarity on primitive vectors are defined in the fuzzy relation namely *PrimitiveVectorsRelation* which is a fuzzy set defined on the set of primitive vector types {0, 1, ..., 12} (see Table 4.1). In Table 4.1, *P_S* is the similarity value for line vectors and *P_Z* is for circle vectors. Both *P_S* and *P_Z* $\in$ [0, 1] and they are obtained by the experiments while the blank cells indicate the value 0.

Table 4.1: Fuzzy relation *PrimitiveVectorsRelation*.

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	1	*P_S*						*P_S*					
1	*P_S*	1	*P_S*										
2		*P_S*	1	*P_S*									
3			*P_S*	1	*P_S**								
4				*P_S*	1	*P_S*							
5					*P_S*	1	*P_S*						
6						*P_S*	1	*P_S*					
7	*P_S*						*P_S*	1					
8									1	*P_Z*	*P_Z*	*P_Z*	*P_Z*
9									*P_Z*	1	*P_Z*		*P_Z*
10									*P_Z*	*P_Z*	1	*P_Z*	
11									*P_Z*		*P_Z*	1	*P_Z*
12									*P_Z*	*P_Z*		*P_Z*	1

Let x denote a primitive vector, and suppose we have a proposition $p \equiv$ "Primitive type of x is 4". The truth value of this proposition can be computed as follows:

STEP 1: Find the actually primitive type of x, say 3.
STEP 2: The truth value of p is assigned to be the membership grade to which the 2-tuple (3, 4) belongs to the fuzzy relation *PrimitiveVectorsRelation*, which is *P_S*. (notice * in Table 4.1)

Example 4.3
In [Supanwansa, 2540], *primitive vector zones* (or *vector zones*) are used to address the position of vectors. We address the position of a vector using its *starting zone* and its *ending zone*. The vector zones consist of eight zones: zone 0, zone 1, ..., zone 7 (see Figure 4.3). Next, the similarity on vector zones are defined by the fuzzy relation namely *VectorZonesRelation* which is a fuzzy set defined on the set of vector zones {0, 1, ..., 7}. (see Table 4.2) The value $Z\_S \in [0, 1]$ is obtained by the experiments while the blank cells indicate the value 0.

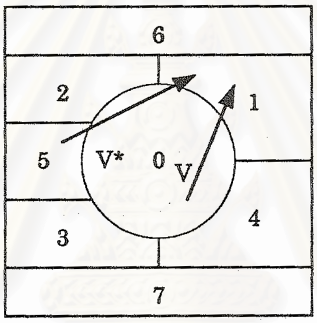

Figure 4.3: Vector zones

Table 4.2: Fuzzy relation *VectorZonesRelation*.

	0	1	2	3	4	5	6	7
0	1	*Z_S*	*Z_S*	*Z_S*	*Z_S*	*Z_S*		
1	*Z_S*	1	*Z_S*		*Z_S*		*Z_S*	
2	*Z_S*	*Z_S*	1			*Z_S*	*Z_S*	
3	*Z_S*			1	*Z_S*	*Z_S*		*Z_S*
4	*Z_S*	*Z_S*		*Z_S*	1			*Z_S*
5	*Z_S*		*Z_S*	*Z_S*	0	1		
6		*Z_S*	*Z_S*				1	
7				*Z_S*	*Z_S*			1

To use this fuzzy relation, let V and V^* be two primitive vectors shown in Figure 4.3; V starts in zone 0 and ends in zone 1 while V^* starts in zone 5 and ends in zone 1.

The truth value of the proposition $p_1 \equiv$ "V starts in zone 4" is computed as follows:

STEP1: Find the actually starting zone of V, which is 0.
STEP2: Analogous to Example 4.2, $T(p_1)$ is assigned to be the membership grade to which the 2-tuple (0, 4) belongs to the fuzzy relation *VectorZonesRelation*, which is *Z_S*, e.g., 0.25.

Next, the truth value of the proposition $p_2 \equiv$ "V^* starts in zone 4" is computed as follows:

STEP1: Find the actually starting zone of V^*, which is 5.
STEP2: Analogously, $T(p_2)$ is assigned to be the membership grade to which the 2-tuple (5, 4) belongs to the fuzzy relation *VectorZonesRelation*, which is 0.

Example 4.4

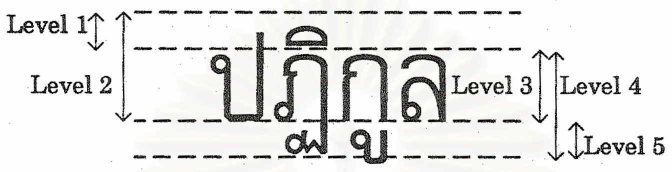

Figure 4.4: Five levels of Thai writing system.

Let us consider Figure 4.4. Thai characters are written in five levels, that are, level 1 to level 5. Thus, the level of each character in the string "ปฏิกูล" are 2, 4, 1, 3, 5 and 3, respectively. We define a fuzzy set on {1, 2, 3, 4, 5}×{1, 2, 3, 4, 5}, namely *FuzzyLevelsRelation* to express the matter of degree these levels are related. (see Table 4.3). The value $L\_S \in [0, 1]$ is obtained by the experiments while the blank cells indicate the value 0. The method of using this fuzzy set is analogous to the previous examples.

Table 4.3: Fuzzy relation *FuzzyLevelsRelation*.

	1	2	3	4	5
1	1				
2		1	*L_S*		
3		*L_S*	1	*L_S*	
4			*L_S*	1	
5					1

We will introduce another type of fuzzy proposition in the next section.

4.2 COMPOUND FUZZY PROPOSITION

A *compound fuzzy proposition* is composed of propositions connected with *logical connectives*: negation connective, conjunction connective, disjunction connective and implication connective. Each of these connectives will be described in subsection 4.2.1, 4.2.2, 4.2.3 and 4.2.4, respectively.

4.2.1 NEGATED FUZZY PROPOSITION

Definition 4.2 *A negated fuzzy proposition is a fuzzy proposition of the form*

$$p \equiv \text{"}\neg\Phi\text{"} \qquad \text{..........(4.4)}$$

where Φ is any fuzzy proposition. The truth value of p is defined as

$$T(p) = T(\text{"}\neg\Phi\text{"}) = c(T(\Phi)) \qquad \text{..........(4.5)}$$

where c is any fuzzy complement function. □

Here, we employ the standard fuzzy complement function, $c(x) = 1-x$.

Example 4.5

Let V be a primitive vector and p be the proposition "V does not start in zone 4". Clearly, p can be converted into the equivalent proposition "¬(V starts in zone 4)". In order to find $T(p)$, we do the following steps:

STEP1: Find T("V starts in zone 4"), which is 0.25 by Example 4.3.

STEP2: T("V does not starts in zone 4") = T("¬(V starts in zone 4)")= $c(T$("V starts in zone 4")) = 1-0.25 = 0.75.

4.2.2 CONJUNCTION OF FUZZY PROPOSITIONS

Definition 4.3 *A conjunction of fuzzy propositions is a proposition of the form*

$$p \equiv \text{"}\Phi\wedge\Gamma\text{"} \qquad \text{..........(4.6)}$$

where Φ and Γ are any fuzzy propositions. The truth value of p is defined as

$$T(p) = T(\text{"}\Phi\wedge\Gamma\text{"}) = i(T(\Phi), T(\Gamma)) \qquad \text{..........(4.7)}$$

where i is any fuzzy intersection function. □

Here, we employ the standard fuzzy intersection function, $i(x, y) = \min(x, y)$.

Example 4.6

Let p be the proposition "$(\neg\Phi\wedge\Gamma)\wedge\neg(\Theta\wedge\Psi)$" where Φ, Γ, Θ and Ψ are some fuzzy propositions. Using standard fuzzy complement function and algebraic products, then

$$\begin{aligned}
T(p) &= T(\text{"}(\neg\Phi\wedge\Gamma)\wedge\neg(\Theta\wedge\Psi)\text{"})\\
&= i(T(\text{"}\neg\Phi\wedge\Gamma\text{"}), T(\text{"}\neg(\Theta\wedge\Psi)\text{"}))\\
&= i(i(T(\text{"}\neg\Phi\text{"}), T(\Gamma)), c(T(\text{"}\Theta\wedge\Psi\text{"})))\\
&= i(i(T(\text{"}\neg\Phi\text{"}), T(\Gamma)), c(i(T(\Theta), T(\Psi))))\\
&= i(i(c(\mathrm{T}(\Phi)), T(\Gamma)), c(i(T(\Theta), T(\Psi))))\\
&= ([1-T(\Phi)]\times T(\Gamma))\times(1-[T(\Theta)\times T(\Psi]))
\end{aligned}$$

Suppose $T(\Phi) = 0.5$, $T(\Gamma) = 0.6$, $T(\Theta) = 0.7$ and $T(\Psi) = 0.8$, then

$$\begin{aligned}
T(p) &= ([1-0.5]\times0.6)\times(1-[0.7\times0.8])\\
&= 0.3\times0.44\\
&= 0.1452
\end{aligned}$$

Next, let V be a primitive vector and p be the proposition "V starts in zone 4 *and* V does not start in zone 4". p can be rewritten as the proposition "$\Phi\wedge\neg\Phi$" where Φ stands for the proposition "V starts in zone 4". By Example 4.5, we got $T(\Phi) = 0.25$ and $T(\neg\Phi) = 0.75$. Therefore, we have the following.

- If we use the standard fuzzy intersection function, then $T(p) = i(T(\Phi), T(\neg\Phi)) = \min(0.25, 0.75) = 0.25$.

- If we use algebraic product, then $T(p) = i(T(\Phi), T(\neg\Phi)) = 0.25 \times 0.75 = 0.1875$.

Here, $T(p)$ *is not equal to 0* even p is of the form "$\Phi \wedge \neg\Phi$".

Let us go to the next type of the compound fuzzy proposition.

4.2.3 DISJUNCTION OF FUZZY PROPOSITIONS

Definition 4.4 *A disjunction of fuzzy propositions is a proposition of the form*

$$p \equiv \text{"}\Phi \vee \Gamma\text{"} \qquad \text{..........(4.8)}$$

where Φ and Γ are any fuzzy propositions. The truth value of p is defined as

$$T(p) = T(\text{"}\Phi \vee \Gamma\text{"}) = u(T(\Phi), T(\Gamma)) \qquad \text{..........(4.9)}$$

where u is any fuzzy union function. □

In this work, we employ standard fuzzy union function, $u(x, y) = \max(x, y)$.

Example 4.7
Let V be a primitive vector and p be the proposition "V starts in zone 4 *or* V does not start in zone 4". p can be rewritten as the proposition "$\Phi \vee \neg\Phi$" where Φ stands for the proposition "V starts in zone 4".
By Example 4.4, we got $T(\Phi) = 0.25$ and $T(\neg\Phi) = 0.75$. Therefore, we have the following.

- If we use the standard fuzzy union function, then $T(p) = u(T(\Phi), T(\neg\Phi)) = \max(0.25, 0.75) = 0.75$.

- If we use algebraic sum, then $T(p) = u(T(\Phi), T(\neg\Phi)) = 0.25 + 0.75 - 0.25 \times 0.75 = 1 - 0.1875 = 0.8125$.

Here, $T(p)$ *is not equal to 1* even p is of the form "$\Phi \vee \neg\Phi$".

Let us go to the last type of the compound fuzzy proposition.

4.2.4 IMPLICATION OF FUZZY PROPOSITIONS

Definition 4.5 *An implication of fuzzy propositions is a proposition of the form*

$$p \equiv \text{"}\Phi \rightarrow \Gamma\text{"} \qquad \text{..........(4.10)}$$

where Φ and Γ are any fuzzy propositions. The truth value of p is defined as

$$T(p) = T(''\Phi \to \Gamma'') = I(T(\Phi), T(\Gamma)) \qquad \text{..........(4.11)}$$

where I is a fuzzy implication function. I should be an extension of the classical implication from the restricted domain {0, 1} into the full domain [0, 1] of the truth values in fuzzy sense. The following properties may be viewed as optional reasonable axioms of fuzzy implication functions.

Axiom I1. *$a \le b$ implies $I(a, x) \ge (b, x)$* *(monotonicity in first argument)*
This means that the truth value of fuzzy implications increases as the truth value of the antecedent decreases.

Axiom I2. *$a \le b$ implies $I(x, a) \le (x, b)$* *(monotonicity in second argument)*
This means that the truth value of fuzzy implications increases as the truth value of the consequent increases.

Axiom I3. *$I(0, x) = 1$* *(dominance of falsity)*
This means that the falsity implies everything.

Axiom I4. *$I(1, x)= x$* *(neutrality of truth)*
This means that the truth does not imply anything.

Axiom I5. *$I(x, x)=1$* *(identity)*
This means that the fuzzy implications are true whenever the truth values of the antecedent and consequent are equal.

Axiom I6. *$I(a, I(b, x))= I(b, I(x, a))$* *(exchange property)*
This is a generalisation of the equivalence of $a\to(b\to x)$ and $b\to(a\to x)$ that holds for the classical implication.

Axiom I7. *$I(x, y) = 1$ iff $x \le y$* *(boundary condition)*
This means that fuzzy implications are true if and only if the consequent is at least true as the antecedent.

Axiom I8. *$I(a, b) = I(c(b), c(a))$ for a fuzzy complement function c.*
(contraposition)
This means that fuzzy implications are equally true when the antecedent and consequent are exchanged and negated.

Axiom I9. *I is a continuous function* *(continuity)*
This property ensures that small changes in the truth values of the antecedent or consequent do not produce a large (discontinuous) changes in the truth values of fuzzy implications. □

Here are some fuzzy implication functions:

- Zadeh's:

$$I(x, y) = \max[1-x, \min(x, y)] \qquad \text{...............(4.12)}$$

- Gaines-Rescher's:

$$I(x,y) = \begin{cases} 1; & x \leq y \\ 0; & x > y \end{cases} \qquad \text{...............(4.13)}$$

- Gödel's:

$$I(x,y) = \begin{cases} 1; & x \leq y \\ y; & x > y \end{cases} \qquad \text{...............(4.14)}$$

- Lukasiecwicz's:

$$I(x, y) = \min(1, 1-x+y) \qquad \text{...............(4.15)}$$

It should be note that there is no proposition of the implication form in our background knowledge.

So far, all of compound fuzzy propositions are described. The next section will introduce another type of fuzzy proposition.

4.3 QUANTIFIED FUZZY PROPOSITION

In this research, three types of fuzzy quantifiers are used to quantify a fuzzy proposition: existentially fuzzy quantifier, universally fuzzy quantifier and fuzzy-number based fuzzy quantifier. They will be described in subSections 4.3.1, 4.4.2 and 4.4.3 respectively.

4.3.1 FUZZY PROPOSITION QUANTIFIED WITH EXISTENTIALLY FUZZY QUANTIFIER

Definition 4.6 *A fuzzy proposition quantified with existentially fuzzy quantifier is a proposition of the form*

$$p \equiv \text{"}\exists v \in L\text{: } A(v)\text{"} \qquad \text{...........(4.16)}$$

where A(v) is some predicate involving variable v. L is a finite set of objects. This proposition reads "There is some v in L that is A", and the truth value of p is defined as

$$T(p) = T(\text{"}\exists v \in L\text{: } A(v)\text{"}) = \max_{x \in L}\left[T\left(A(v \leftarrow x)\right)\right] \qquad \text{...........(4.17)}$$

where $A(v \leftarrow x)$ is the proposition which is obtained by substituting all occurrences of variable v with x, in A. □

Example 4.8

In [Supanwansa, 2540], there is a proposition *EndPointTopRight* which is defined by the following PROLOG code [Clocksin and Mellish, 1984]. L stands for a finite list of primitive vectors.

```
EndPointTopRight(L):- member(V, L), endpoint(V, -1),
                      startzone(V,1), endzone(V, 1).
EndPointTopRight(L):- member(V, L), endpoint(V, -1),
                      startzone(V,2), endzone(V,1).
```

Clearly, *EndPointTopRight* is equivalent to the proposition q defined by,

$$q \equiv \text{"}\exists v\in L\text{: } (v \text{ is EndPoint})\wedge(v \text{ starts in zone 1})\wedge(v \text{ ends in zone 1}) \vee \exists v\in L\text{: } (v \text{ is EndPoint})\wedge(v \text{ starts in zone 2})\wedge(v \text{ ends in zone 1})\text{"}$$

Mathematically, q is equivalent to the proposition p defined by,

$$p \equiv \text{"}\exists v\in L\text{: } (v \text{ is EndPoint})\wedge(v \text{ starts in zone 1}\vee v \text{ starts in zone 2})\wedge(v \text{ ends in zone 1})\text{"}$$

Since L is finite, suppose L is $\{x_1, x_2, \ldots, x_6\}$ and suppose the truth values of each atomic fuzzy propositions in p are shown in Table 4.4 below:

Table 4.4: The truth values of atomic propositions in Example 4.8.

substitution	*T("v is EndPoint")*	*T("v starts in zone 1")*	*T("v starts in zone 2")*	*T("v ends in zone 1")*
$v \leftarrow x_1$	1	1	0.25	0.25
$v \leftarrow x_2$	1	0.25	0.25	0.25
$v \leftarrow x_3$	1	0	0	1
$v \leftarrow x_4$	1	0	1	0.25
$v \leftarrow x_5$	0	0.25	0.25	0
$v \leftarrow x_6$	1	0	0.25	1

Note that, in this research, the concept "is EndPoint" is crisp, because there is no information that we can use them to form a fuzzy set that represents concept "is EndPoint".

If we use $i(x, y) = x\cdot y$ and $u(x, y) = \max(x, y)$, then the truth values T("(v starts in zone 1$\vee$ v starts in zone 2)") will be shown in Table 4.5. Next, the truth values T("(v is EndPoint)$\wedge$(v starts in zone 1$\vee v$ starts in zone 2)$\wedge$(v ends in zone 1)") will be given in Table 4.6.

Table 4.5: T("v starts in zone 1$\vee v$ starts in zone 2").

substitution	*T("v starts in zone 1∨v starts in zone 2")*
$v \leftarrow x_1$	max(1, 0.25) = 1
$v \leftarrow x_2$	max(0.25, 0.25) = 0.25
$v \leftarrow x_3$	max(0,0) = 0
$v \leftarrow x_4$	max(0,1) = 1
$v \leftarrow x_5$	max(0.25, 0.25) = 0.25
$v \leftarrow x_6$	max(0,0.25) = 0.25

Table 4.6: T("(v is EndPoint)∧(v starts in zone 1∨v starts in zone 2) ∧(v ends in zone 1)").

substitution	*T("(v is EndPoint)∧(v starts in zone 1∨v starts in zone 2) ∧ (v ends in zone 1)")*
$v \leftarrow x_1$	1×1×0.25 = 0.25
$v \leftarrow x_2$	1×0.25×0.25 = 0.0625
$v \leftarrow x_3$	1×0×1 = 0
$v \leftarrow x_4$	1×1×0.25 = 0.25
$v \leftarrow x_5$	0×0.25×0 = 0
$v \leftarrow x_6$	1×0.25×1 = 0.25

Finally, $T(EndPointTopRight)$ = max[0.25, 0.0626, 0, 0.25, 0, 0.25] = 0.25.

We now ready to go to the next type of fuzzy quantifier.

4.3.2 FUZZY PROPOSITION QUANTIFIED WITH UNIVERSALLY FUZZY QUANTIFIER

Definition 4.7 *A fuzzy proposition quantified with universally fuzzy quantifier is a proposition of the form*

$$p \equiv \text{"}\forall v \in L\text{: } A(v)\text{"} \qquad \text{........... (4.18)}$$

where A(v) is some predicate involving variable v. L is a finite set of objects. This proposition reads "All v in L are A", and the truth value of p is defined as

$$T(p) = T(\text{"}\forall v \in L\text{: } A(v)\text{"}) = \min_{x \in L}\left[T\left(A(v \leftarrow x)\right)\right] \qquad \text{........... (4.19)}$$

where A(v ← x) is the proposition which is obtained by substituting all occurrences of variable v with x, in A. □

Note that there is no proposition of this form in our background knowledge.

Let us go to the last type of fuzzy quantifier.

4.3.3 FUZZY PROPOSITION QUANTIFIED WITH FUZZY-NUMBER BASED FUZZY QUANTIFIER

Definition 4.7 *A fuzzy proposition quantified with fuzzy-number based fuzzy quantifier is a proposition of the form*

$$p \equiv \text{"}Qv \in L\text{: } A(v)\text{"} \qquad \text{...........(4.18)}$$

where A(v) is some predicate involving variable v. L is a finite set of objects. Q is fuzzy-number based fuzzy quantifier which is a fuzzy number used to represent the concept of some quantity, such as "Around five", "More than nine" or "Most".

This proposition reads "Q v in L are A" and the truth value of p is defined to be the membership grade to which the sigma count, $\sum_{x \in L} T\left(A(v \leftarrow x)\right)$ *belongs to the fuzzy quantifier Q, that is,*

$$T(p) = T(\text{"}Qv \in L\text{: } A(v)\text{"}) = Q\left(\sum_{x \in L} T(A(v \leftarrow x))\right) \quad \text{......... (4.19)}$$

where $A(v \leftarrow x)$ is the proposition which is obtained by substituting all of occurrences of variable v with x, in A. □

Example 4.9

Let us consider another proposition used in this research. Let $x \equiv (x_0, \dots, x_7)$ and $y \equiv (y_0, \dots, y_7)$ be two eight-tuples of natural numbers. x_i or y_i is the total number of the i^{th} vector zone used by some primitive vector in a given character (e.g., if some vector starts in zone 2, we increase x_2 by 1; if some vector ends in zone 3, we also increase x_3 by 1).

Let U be the universal set of vector zones {0, 1, ..., 7}, then the similarity between x and y can be viewed as the truth value of the fuzzy proposition p which is quantified with fuzzy-number based fuzzy quantifier:

$$p \equiv \text{"Most } i \in U\text{: } x_i \text{ is closed to } y_i\text{"} \quad \text{..........(4.20)}$$

If we employ $FuzzyAround{:}k_1{:}|\ U\ |{:}$ to represent concept "Most", then

$$T(p) = FuzzyAround : k_1 : |\,U\,| : \left(\sum_{i=0}^{7} T(\text{"}x_i \textit{ is closed to } y_i\text{"})\right) \quad \text{..........(4.21)}$$

For each $i \in \{0, \dots, 7\}$, if we use fuzzy set $FuzzyAround{:}k_2{:}y_i{:}$ to represent concept "is closed to y_i", then $T(p)$ become

$$T(p) = FuzzyAround : k_1 : |\,U\,| : \left(\sum_{i=0}^{7} FuzzyAround : k_2 : y_i : (x_i)\right) \quad \text{..........(4.22)}$$

Suppose $FuzzyAround\text{: } k_2 : y_0 : (x_0) = 0.9$, $FuzzyAround\text{: } k_2 : y_1 : (x_1) = 0.8$, $FuzzyAround\text{: } k_2 : y_2 : (x_2) = 0.7$, $FuzzyAround\text{: } k_2 : y_3 : (x_3) = 0$, $FuzzyAround\text{: } k_2 : y_4 : (x_4) = 0.1$, $FuzzyAround\text{: } k_2 : y_5 : (x_5) = 0.2$, $FuzzyAround\text{: } k_2 : y_6 : (x_6) = 0.4$ and $FuzzyAround\text{: } k_2 : y_7 : (x_7) = 1$.

Then $\sum_{i=0}^{7} FuzzyAround : k_2 : y_i : (x_i) = .9{+}.8{+}.7{+}0{+}.1{+}.2{+}.4{+}1 = 4.1.$

Suppose $k_1 = 1$, then finally, $T(p) = FuzzyAround : k_1 : 8 : (4.1) = e^{-|8-4.1|} = e^{-3.9} = 0.02.$

From this point, let us verify our definition. Assume that x is identical to y, then

$$\begin{aligned} T(p) &= FuzzyAround : k_1 : 8 : \left(\sum_{i=0}^{7} FuzzyAround : k_2 : x_i : (x_i)\right) \\ &= FuzzyAround : k_1 : 8 : \left(\sum_{i=0}^{7} e^0\right) \\ &= FuzzyAround : k_1 : 8 : \left(\sum_{i=0}^{7} 1\right) \end{aligned}$$

I 20297427

$$= FuzzyAround:k_1:8:(8)$$

$=$ 1, which is correct because x is maximally similar to itself.

Example 4.10

In [Supanwansa, 2540], there is a proposition p of the form "There are around n primitive vectors in the list L that are of type a". This is a fuzzy proposition quantified with a fuzzy-number based fuzzy quantifier, and the truth value of p is defined as

$$T(p) = FuzzyAround{:}k{:}n{:}\left(\sum_{x\in L} T(\text{"}x \textit{ is primitive of type } a\text{"})\right) \quad \text{..........(4.23)}$$

The computation procedure is analogous to Example 4.8.

Suppose $n = 5$, $L = \{x_1, x_2, x_3, x_4, x_5, x_6\}$, $k = 1$ and

T("x_1 is primitive of type a") = 0.1, T("x_2 is primitive of type a") = 0.3,
T("x_3 is primitive of type a") = 0.5, T("x_4 is primitive of type a") = 0.4,
T("x_5 is primitive of type a") = 0 and T("x_6 is primitive of type a") = 1.

So, $\sum_{x\in L} T(\text{"}x \textit{ is primitive of type } a\text{"}) = 0.1+0.3+0.5+0.4+0+1 = 2.3$

Finally, $T(p) = FuzzyAround{:}1{:}5{:}(2.3) = e^{-|5-2.3|} = e^{-2.7} = 0.07$.

Example 4.11

In [Supanwansa, 2540], there is a proposition p of the form "There are more than n primitive vectors in the list L that are of type a". This is a fuzzy proposition quantified with a fuzzy-number based fuzzy quantifier, and the truth value of p is defined as

$$T(p) = FuzzyGreaterThan{:}k{:}x_0{:}\left(\sum_{x\in L} T(\text{"}x \textit{ is primitive of type } a\text{"})\right) \quad \text{..........(4.24)}$$

where

$$FuzzyGreaterThan{:}k{:}x_0{:}(x) = \begin{cases} 1; & x > x_0 \\ k; & x = x_0 \\ 0; & otherwise \end{cases} \quad \text{..........(4.25)}$$

Suppose $n = 5$, $L = \{x_1, x_2, x_3, x_4, x_5, x_6\}$, $k = 1$ and

T("x_1 is primitive of type a") = 0.1, T("x_2 is primitive of type a") = 0,
T("x_3 is primitive of type a") = 0.5, T("x_4 is primitive of type a") = 0.4,
T("x_5 is primitive of type a") = 0 and T("x_6 is primitive of type a") = 1.

So, $\sum_{x\in L} T(\text{"}x \textit{ is primitive of type } a\text{"}) = 0.1+0+0.5+0.4+0+1 = 2.0$

Finally, $T(p) = FuzzyGreaterThan{:}0.5{:}2.0{:}(2.0) = 0.5$.

We also provide the fuzzy set namely, *FuzzyLessThan:x₀:n:* defined as

$$FuzzyLessThan{:}k{:}x_0{:}(x) = \begin{cases} 1; & x < x_0 \\ k; & x = x_0 \\ 0; & otherwise \end{cases} \qquad \text{..........(4.26)}$$

Sofar, we are always able to find the truth value of every proposition in the background knowledge because they must be one of the basic three types: atomic, compound or quantified. Figure 4.5 shows the conclusion of the fuzzy truth values evaluation procedure.

The full list of fuzzy propositions used in this research can be found in Appendix B.

```
function T(p : fuzzy proposition) : [0, 1];
{
    case p of
    {
        "x is A":          $T := A(x)$;
        "¬Φ":              $T := c(T(\Phi))$;
        "Φ∧Γ":             $T := i(T(\Phi), T(\Gamma))$;
        "Φ∨Γ":             $T := u(T(\Phi), T(\Gamma))$;
        "Φ→Γ":             $T := I(T(\Phi), T(\Gamma))$;
        "∃v∈L: A(v)":      $T := \max_{x \in L}[T(A(v \leftarrow x))]$;
        "∀v∈L: A(v)":      $T := \min_{x \in L}[T(A(v \leftarrow x))]$;
        "Qv∈L: A(v)":      $T := Q\left(\sum_{x \in L} T(A(v \leftarrow x))\right)$;
    }
}
```

Figure 4.5: Conclusion of the truth values evaluation procedure.

4.4 TRANSFORMATION OF TRUTH VALUES

Some technique can dramatically increase recognition accuracy while decreases the amount of time required for training the BNN. Before the fuzzy(also for crisp) truth values are sent to the BNN for learning or for recognising, we transform these truth values which range in [0, 1] into another range [–1, +1] by the mapping *f* defined by

$$f(x) = 2 \cdot x - 1 \qquad \text{..........(4.27)}$$

It follows that, $f(0) = -1$, $f(1) = +1$ and $f(0.5) = 0$. For example, applying (4.27) to a vector $[1\ \ 0\ \ 0.5\ \ 0.1\ \ 0.9]^T$ gives $[f(1)\ \ f(0)\ \ f(0.5)\ \ f(0.1)\ \ f(0.9)]^T$ which is $[+1\ \ -1\ \ 0\ \ -0.8\ \ +0.8]^T$.

The experimental result using these truth values will be given in the next chapter.

CHAPTER 5
EXPERIMENTAL RESULTS

We run experiments to test our hypothesis. First of all, before FST is applied, we determine by experiments the most optimised configuration of the BNN, i.e., the learning rate, the momentum and the convergence condition. This is important in the same sense we optimise the serial algorithm before we parallelise it. The best BNN configuration ensures us the worth of our method, if any worth.

5.1 DATA SET

All training and test characters in our experiments are printed by a 300-dpi laser printer and scanned into the computer with the same resolution scanner.

Let E and C stand for the sets of Eucrosia and Cordia fonts, respectively. Let E^{darker} and C^{darker} denote noised Eucrosia and noised Cordia fonts, which are obtained by copying the images by a photocopy machine with darker setting, respectively.

In the same way, let $E^{lighter}$ and $C^{lighter}$ denote noised Eucrosia and noised Cordia fonts, which are obtained by copying the images by a photocopy machine with lighter setting, respectively. These images consists of 77 different Thai characters ('ก', ... , '๙') and 7 sizes (20, 22, 24, 28, 32, 36 and 48 points).

Therefore, the size of each character set ($|E|$, $|C|$, $|E^{darker}|$, $|C^{darker}|$, $|E^{lighter}|$, or $|C^{lighter}|$) is equal to $77 \times 7 = 539$.

5.2 EXPERIMENTAL RESULTS

There are three cases of experiments for testing, i.e., noisy images, unseen noise-free images and unseen noisy images. We compare our proposed method with the exactly match ILP [Supanwansa, 2540] and the optimised ILP&BNN [Kijsirikul and Sinthupinyo, 1999].

Experiment 1:
The training set is $E \cup C$. The test set is $E^{darker} \cup C^{darker} \cup E^{lighter} \cup C^{lighter}$. The size of the training set is $|E \cup C| = 2 \times 539 = 1{,}078$ while the size of the test set is $|E^{darker} \cup C^{darker} \cup E^{lighter} \cup C^{lighter}| = 4 \times 539 = 2{,}158$.

Experiment 2:
The training set is E. The test set is C. The size of the training set is $|E| = 539$ while the size of the test set is $|C| = 539$.

Experiment 3:
The training set is E. The test set is $E^{darker} \cup C^{darker} \cup E^{lighter} \cup C^{lighter}$. The size of the training set is $|E| = 539$ while the size of the test set is $|E^{darker} \cup C^{darker} \cup E^{lighter} \cup C^{lighter}| = 4 \times 539 = 2{,}158$.

The recognition accuracy of all experiments are shown in Table 5.1 below:

Table 5.1: Recognition accuracy of all experiments.
(The maximum value in each row is printed in italic face.)

Experiment	*#TRAIN*	*#TEST*	*ILP*	*ILP&BNN*	*ILP&BNN&FST*
1	1,078	2,158	84.23%	94.77%	*96.03%*
2	539	539	*87.69%*	79.48%	82.65%
3	539	2,158	81.80%	84.23%	*86.70%*
avg.			83.53%	88.39%	*90.40%*

ILP&BNN yields the improvement only for noisy and unseen noisy images compared to the exactly match ILP. In the case of unseen noise-free images, ILP provides the highest accuracy among the three methods. However, ILP&BNN&FST always gives better recognition accuracy than ILP&BNN.

Next, Tables 5.2, 5.3 and 5.4 show the recognition accuracy of Experiment 1, 2 and 3 reported separately by each character. Table 5.5 shows the average recognition accuracy of Experiment 1, 2 and 3 reported separately by each character. Figures 5.1, 5.2, 5.3 and 5.4 are graphical representations of Tables 5.2, 5.3, 5.4 and 5.5, respectively.

Table 5.2: Recognition accuracy of Experiment 1 reported separately by each character.

	ILP	*ILP&BNN*	*ILP&BNN&FST*
ก	89.29	92.86	100
ข	85.71	85.71	92.86
ฃ	82.14	82.14	82.14
ค	82.14	89.29	89.29
ฅ	0	92.86	92.86
ฆ	82.14	89.29	89.29
ง	92.86	96.43	96.43
จ	100	100	100
ฉ	96.43	96.43	100
ช	71.43	96.43	96.43
ซ	76	80	88
ฌ	78.57	92.86	100
ญ	85.71	92.86	96.43
ฎ	92.86	89.29	96.43
ฏ	85.19	85.19	81.48
ฐ	89.29	92.86	89.29
ฑ	57.14	85.71	82.14
ฒ	82.14	92.86	96.43
ณ	75	100	100
ด	64.29	82.14	92.86
ต	76.79	91.07	96.43
ถ	100	100	100
ท	92.86	92.86	92.86
ธ	75	100	100
น	89.29	92.86	89.29
บ	85.71	92.86	92.86
ป	85.71	96.43	89.29
ผ	92.86	89.29	89.29
ฝ	92.86	96.43	96.43
พ	85.71	100	100
ฟ	92.86	100	100
ภ	96.43	100	100
ม	89.29	100	100
ย	60.71	92.86	89.29
ร	0	100	100
ฤ	89.29	96.43	96.43
ล	82.14	100	100
ว	96.43	100	100
ศ	94.74	100	100
ษ	78.57	92.86	92.86
ส	89.29	92.86	100
ห	60.71	96.43	96.43
ฬ	89.29	100	100
อ	85.71	98.21	98.21
ฮ	75	89.29	92.86
ฯ	85.71	96.43	96.43
ะ	89.29	100	100
ั	85.71	92.86	92.86
า	75	100	100
ิ	92.86	92.86	100
ี	85.71	85.71	85.71
ึ	92.86	92.86	92.86
ื	85.71	89.29	92.86
ุ	85.71	89.29	92.86
ู	89.29	96.43	100
เ	91.67	99.40	99.40
โ	89.29	100	100
ใ	96.43	100	100
ไ	92.86	100	100
ๆ	64.29	96.43	100
็	78.57	100	100
่	100	100	100
้	75	92.86	92.86
๊	85.71	89.29	100
๋	85.71	92.86	100
์	82.14	82.14	82.14
ํ	96.43	96.43	96.43
๐	100	100	100
๑	92.86	92.86	96.43
๒	100	100	100
๓	96.43	100	100
๔	85.71	96.43	96.43
๕	100	100	100
๖	78.57	100	100
๗	96.43	100	100
๘	85.71	100	100
๙	100	100	100

Table 5.3: Recognition accuracy of Experiment 2 reported separately by each character.

	ILP	*ILP&BNN*	*ILP&BNN&FST*
ก	100	100	100
ข	0	28.57	42.86
ฃ	14.29	14.29	14.29
ค	100	100	100
ฅ	100	100	100
ฆ	100	100	100
ง	100	100	100
จ	100	85.71	71.43
ฉ	100	71.43	71.43
ช	100	28.57	42.86
ซ	85.71	85.71	71.43
ฌ	85.71	100	85.71
ญ	100	71.43	100
ฎ	100	0	71.43
ฏ	100	100	100
ฐ	28.57	28.57	42.86
ฑ	100	100	100
ฒ	100	100	100
ณ	0	0	0
ด	100	100	100
ต	85.71	85.71	85.71
ถ	100	85.71	85.71
ท	100	14.29	85.71
ธ	100	0	0
น	100	71.43	100
บ	100	14.29	28.57
ป	71.43	100	100
ผ	28.57	100	100
ฝ	100	92.86	92.86
พ	100	100	100
ฟ	100	85.71	85.71
ภ	100	100	100
ม	57.14	85.71	85.71
ย	100	100	100
ร	0	100	100
ฤ	100	100	100
ล	100	57.14	100
ว	100	100	100
ศ	16.67	16.67	16.67
ษ	100	100	100
ส	100	100	50
ห	100	100	0
ฬ	100	85.71	85.71
อ	100	64.29	71.43
ฮ	100	100	71.43
ฯ	100	100	85.71
ะ	100	100	100
ั	85.71	85.71	85.71
า	100	71.43	71.43
ิ	100	100	100
ี	100	100	100
ึ	28.57	57.14	100
ื	57.14	64.29	85.71
ุ	100	100	100
ู	100	85.71	28.57
เ	100	95.24	95.24
โ	85.71	85.71	100
ใ	100	100	100
ไ	100	100	100
ๆ	57.14	57.14	100
็	100	0	0
่	100	100	100
้	100	100	100
๊	100	57.14	57.14
๋	100	100	100
์	100	100	100
ํ	100	100	100
๐	100	100	100
๑	100	100	85.71
๒	85.71	100	100
๓	100	100	100
๔	100	14.29	100
๕	100	57.14	85.71
๖	71.43	100	100
๗	100	85.71	85.71
๘	100	85.71	100
๙	85.71	100	100

Table 5.4: Recognition accuracy of Experiment 3 reported separately by each character.

	ILP	*ILP&BNN*	*ILP&BNN&FST*
ก	89.29	92.86	100
ข	53.57	60.71	92.86
ฃ	39.29	42.86	82.14
ค	82.14	85.71	89.29
ฅ	92.86	96.43	92.86
ฆ	82.14	89.29	89.29
ง	92.86	96.43	96.43
จ	100	100	100
ฉ	96.43	75	100
ช	75	60.71	96.43
ซ	84	88	88
ฌ	78.57	92.86	100
ญ	85.71	92.86	96.43
ฎ	92.86	7.14	96.43
ฏ	88.89	100	81.48
ฐ	39.29	57.14	89.29
ฑ	57.14	82.14	82.14
ฒ	89.29	96.43	96.43
ณ	10.71	39.29	100
ด	75	85.71	92.86
ต	82.14	85.71	96.43
ถ	100	100	100
ท	92.86	60.71	92.86
ธ	92.86	50	100
น	89.29	82.14	89.29
บ	89.29	53.57	92.86
ป	60.71	78.57	89.29
ผ	42.86	85.71	89.29
ฝ	92.86	96.43	96.43
พ	89.29	92.86	100
ฟ	92.86	100	100
ภ	100	100	100
ม	78.57	96.43	100
ย	64.29	89.29	89.29
ร	0	100	100
ฤ	89.29	92.86	96.43
ล	100	82.14	100
ว	96.43	100	100
ศ	42.11	57.89	100
ษ	78.57	85.71	92.86
ส	96.43	100	100
ห	60.71	85.71	96.43
ฬ	92.86	89.29	100
อ	87.5	80.36	98.21
ฮ	75	100	92.86
ฯ	85.71	89.29	96.43
ะ	89.29	96.43	100
ั	85.71	85.71	92.86
า	75	64.29	100
ิ	92.86	92.86	100
ี	85.71	78.57	85.71
ึ	53.57	85.71	92.86
ื	67.86	73.21	92.86
ุ	96.43	89.29	92.86
ู	89.29	89.29	100
เ	92.26	92.86	99.40
โ	85.71	96.43	100
ใ	100	100	100
ไ	92.86	100	100
ๅ	64.29	67.86	100
ๆ	82.14	46.43	100
่	100	100	100
้	75	92.86	92.86
๊	85.71	67.86	100
๋	92.86	92.86	100
์	82.14	82.14	82.14
ํ	96.43	96.43	96.43
๐	100	100	100
๑	92.86	92.86	96.43
๒	64.29	100	100
๓	96.43	100	100
๔	85.71	60.71	96.43
๕	100	60.71	100
๖	67.86	100	100
๗	96.43	96.43	100
๘	85.71	96.43	100
๙	96.43	96.43	100

Table 5.5: Average recognition accuracy reported separately by each character.

	ILP	*ILP&BNN*	*ILP&BNN&FST*
ก	92.86	95.24	100.00
ข	46.43	58.33	76.19
ฃ	45.24	46.43	59.52
ค	88.09	91.67	92.86
ฅ	64.29	96.43	95.24
ฆ	88.09	92.86	92.86
ง	95.24	97.62	97.62
จ	100.00	95.24	90.48
ฉ	97.62	80.95	90.48
ช	82.14	61.90	78.57
ซ	81.90	84.57	82.48
ฌ	80.95	95.24	95.24
ญ	90.47	85.72	97.62
ฎ	95.24	32.14	88.10
ฏ	91.36	95.06	87.65
ฐ	52.38	59.52	73.81
ฑ	71.43	89.28	88.09
ฒ	90.48	96.43	97.62
ณ	28.57	46.43	66.67
ด	79.76	89.28	95.24
ต	81.55	87.50	92.86
ถ	100.00	95.24	95.24
ท	95.24	55.95	90.48
ธ	89.29	50.00	66.67
น	92.86	82.14	92.86
บ	91.67	53.57	71.43
ป	72.62	91.67	92.86
ผ	54.76	91.67	92.86
ฝ	95.24	95.24	95.24
พ	91.67	97.62	100.00
ฟ	95.24	95.24	95.24
ภ	98.81	100.00	100.00
ม	75.00	94.05	95.24
ย	75.00	94.05	92.86
ร	0.00	100.00	100.00
ฤ	92.86	96.43	97.62
ล	94.05	79.76	100.00
ว	97.62	100.00	100.00
ศ	51.17	58.19	72.22
ษ	85.71	92.86	95.24
ส	95.24	97.62	83.33
ห	73.81	94.05	64.29
ฬ	94.05	91.67	95.24
อ	91.07	80.95	89.28
ฮ	83.33	96.43	85.72
ฯ	90.47	95.24	92.86
ะ	92.86	98.81	100.00
ั	85.71	88.09	90.48
า	83.33	78.57	90.48
ิ	95.24	95.24	100.00
ี	90.47	88.09	90.47
ึ	58.33	78.57	95.24
ื	70.24	75.60	90.48
ุ	94.05	92.86	95.24
ู	92.86	90.48	76.19
เ	94.64	95.83	98.01
โ	86.90	94.05	100.00
ใ	98.81	100.00	100.00
ไ	95.24	100.00	100.00
ๆ	61.91	73.81	100.00
็	86.90	48.81	66.67
่	100.00	100.00	100.00
้	83.33	95.24	95.24
๊	90.47	71.43	85.71
๋	92.86	95.24	100.00
์	88.09	88.09	88.09
ํ	97.62	97.62	97.62
๐	100.00	100.00	100.00
๑	95.33	95.24	92.86
๒	83.24	100.00	100.00
๓	97.33	100.00	100.00
๔	90.67	57.14	97.62
๕	100.00	72.62	95.24
๖	72.81	100.00	100.00
๗	97.33	94.05	95.24
๘	90.67	94.05	100.00
๙	93.90	98.81	100.00

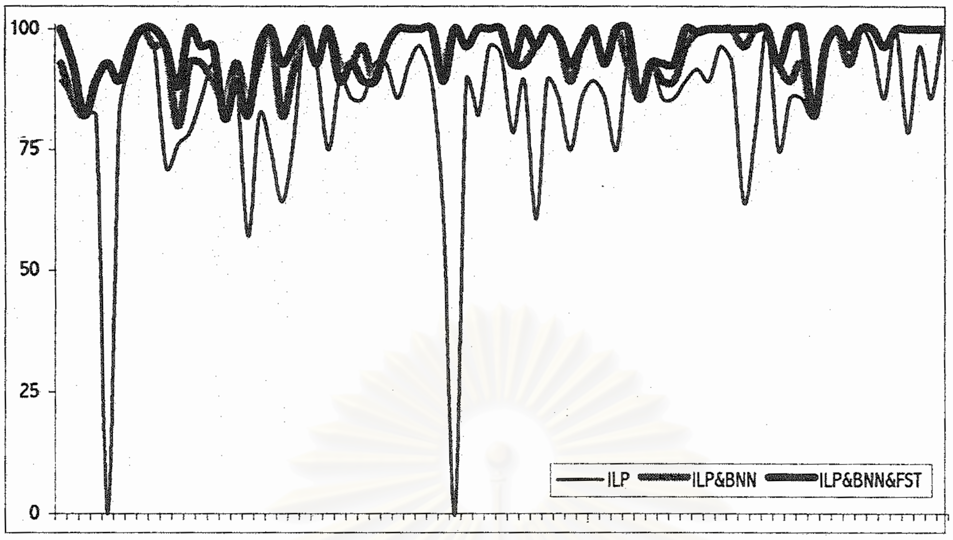

Figure 5.1: Graphical representation of Table 5.2.

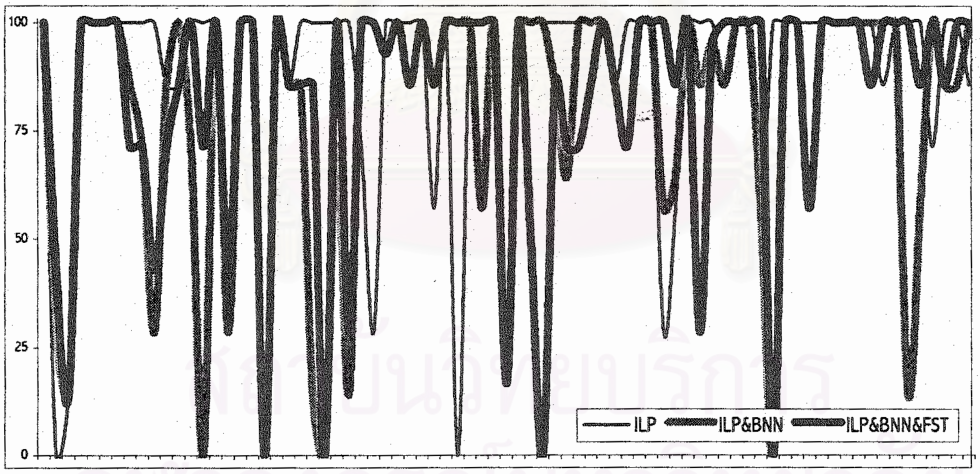

Figure 5.2: Graphical representation of Table 5.3.

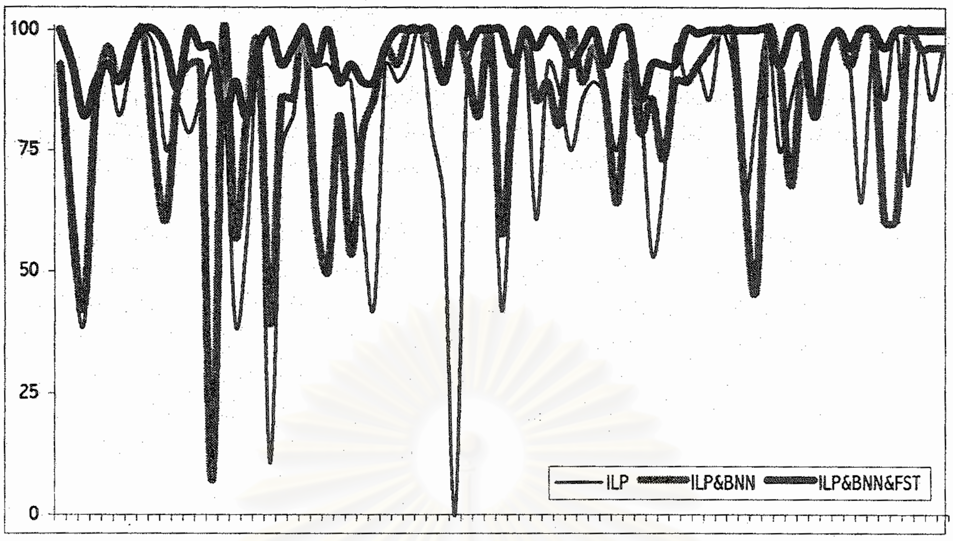

Figure 5.3: Graphical representation of Table 5.4.

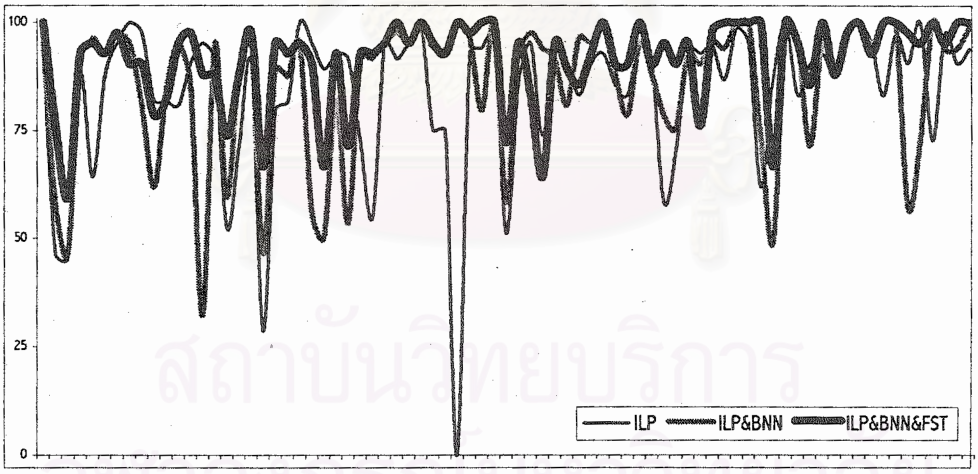

Figure 5.4: Graphical representation of Table 5.5.

5.3 ANALYISIS ON THE IMPROVEMENT

As shown in the experiments, although our method does not improve recognition accuracy of all characters, i.e., some are improved, some are not, in average, around 2% performance up is obtained via our proposed method.

The improved classification accuracy in our experiments are exactly based on the two things. First, the powerful of combination of ILP and BNN, i.e., the high accuracy of the rule generated by ILP plus the learning ability of the BNN that makes the rule more flexible for detecting future imperfect characters. However, the data sent to the BNN is based on crisp set theory, i.e., nothing special is considered about the *nature* of the characters, e.g., the level of the character, the quantity or the angle of the vectors. This prevents the learning ability of the BNN.

Second, FST extends the learning ability of the BNN by considering the nature of the character using several defined fuzzy sets. Thus, FST contributes to the bridge of the gap between the hard binary truth values 0 and 1. More precisely, FST affects the changing of synaptic weights, and this make BNN more flexible for recognising future imperfect characters.

REFERENCES

Clocksin, W. F., and Mellish, C. S. Programming in Prolog. Springer-Verlag, 1984.

Goldberg, D. E. Genetic Algorithms in Search, Optimization & Machine Learning. Addison-Wesley, 1989.

Kandel, A. Fuzzy Mathematical Techniques with Applications. Addison-Wesley, 1986.

Karwowski, W. et al. A Framework for Development of Fuzzy GOMS Model for Human-Computer Interaction. International Human-Computer Interaction., (1990):287-305.

Kijsirikul, B., and Sintupinyo, S. Approximate ILP Rules by Backpropagation Neural Networks: A Result on Thai Character Recognition. Proceeding of the Ninth International Workshop on Inductive Logic Programming., (1999): 53-60.

Klir, G., and Yuan, B. Fuzzy Set and Fuzzy Logic. Prentice-Hall, 1995.

Klir, G., Clair, U., and Yuan, B. Fuzzy Set Theory and Applications. Prentice-Hall, 1997.

Lin, C. T., and Lee, C. S. Neural Fuzzy Systems. Prentice-Hall, 1996.

Martiene, E., and Quafafou, M. Learning Fuzzy Relational Descriptions Using the Logical Framework and Rough Set Theory. IEEE Transaction on Computers., (1998):939-944.

Mendelson, E. Introduction to Mathematical Logic. Wadsworth/Cole, 1987.

Mitchell, T. M. Machine Learning. McGraw-Hill, 1997.

Muggleton, S. Inverse Entailment and PROGOL. New Generation Computing., 3, (1995):245-286.

Pawlak, Z. Rough sets. International Journal of Computer and Information Science., 11, (1982):341-356.

Supanwansa, A. An Application of Inductive Logic Programming to Thai Printed Character Recognition., Master's Thesis, Department of Computer Engineering, Chulalongkorn University, 2540.

Ushida, H. et al. Recognition of Facial Expressions using Conceptual Fuzzy Sets. IEEE Conference on Fuzzy Systems., 1, (1993):594-599.

Wang, C.H. et al. FILSMR: A Fuzzy Induction Learning Strategy for Modular Rules. IEEE Conference on Fuzzy Systems., 3, (1997):1289-1294.

Yen, J. Fuzzy Logic - A Modern Perspective. IEEE Transactions on Knowledge and Data Engineering., 11, 11, (1999):153-165.

Zadeh, L. A. Fuzzy Sets. Information and Control., 8, 3, (1965):338-353.

Zadeh, L. A. Shadows of Fuzzy Sets. Problem in Transmission of Information., 2, (1996):37-44.

Zadeh, L. A. Probability Measures of Fuzzy Events. Journal of Mathematical Analysis and Application., 23, (1968):421-427.

Zadeh, L. A. The Concept of a Linguistic Variable and its Application to Approximate Reasoning. Information Sciences., 8, (1975):199-249.

Zadeh, L. A. Fuzzy Sets as a Basis for a Theory of Possibility. Fuzzy Sets and Systems., 1, 1, (1978):3-28.

Zadeh, L. A. Fuzzy Logic. IEEE Transaction on Computers., 21, 4, (1988):83-93.

Zadeh, L. A. Knowledge Representation in Fuzzy Logic. IEEE Transaction on Knowledge and Data Engineering., 1, (1989):89-100.

Zadeh, L. A. Fuzzy Logic = Computing with Words. IEEE Transactions on Fuzzy Systems., 2, (1996):103-111.

APPENDICES

APPENDIX A
STRUCTURE OF THE BNN USED.

```
#define NPATTERNS   77
#define N_HIDDEN    NPATTERNS
#define IPU         11
#define N_INPUTS    847

DefineBlackBox nn
{
  OutputLayer->  Output_Layer
  InputSize->    N_INPUTS
  Components->
  {
      PdpNode Output_Layer [NPATTERNS]
      {
        InputsFrom-> Hidden_Layer
      }
      PdpNode Hidden_Layer [N_HIDDEN]
      {
        InputsFrom-> $INPUTS (with a [IPUx1] Tessellation)
      }
  }
}
```

APPENDIX B
FULL LIST OF PROPOSITIONS IN C.

```
// Borland C++ 3.1
extern unsigned _stklen=16000;

#include <object.h>
#include <fstream.h>
#include <dos.h>
#include <string.h>
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <ctype.h>
#include <math.h>
#include "dlistimp.h"
#include "shddel.h"
#include "ldate.h"
#include "ltime.h"

#define PIE    22.0/7.0
#define YES    (Truth) 1.00
#define NO     (Truth) 0.00

#define ListType     BI_DoubleListImp
#define IteratorType   BI_DoubleListIteratorImp

typedef  char  String[1024];
typedef  char  SmallString[64];
typedef  float Truth;
typedef  int   SectionIDType;
typedef  int   LevelType;
typedef  int   ZoneType;
typedef  int   PrimitiveType;
typedef  float   NaturalNO;
Truth    DEBUG_CHAR_INFO_MODE=NO;

Truth DecisionList[249-161+1];
SmallString CharsAssoc[249-161+1];

ListType<char>          List;
IteratorType<char>      I(List);

SmallString             CharName;
LevelType               Level;
float                   Ratio;
SmallString             RatioString;
ZoneType                Zone;
ListType<SectionIDType> C;
ListType<ZoneType>      J,L,U;
NaturalNO CountZone_J[8];
NaturalNO CountZone_L[8];
NaturalNO CountZone_U[8];

SectionIDType           HeadSectionID;
String                  input;

Truth Max(Truth x,Truth y)
{
  if (x >=y)
    return x;
  else
    return y;
}

Truth Min(Truth x,Truth y)
{
  if (x < y)
    return x;
  else
    return y;
}
Truth DrasticIntersection(Truth a,Truth b)
{
  if (b==1)
    return a;
  else
    if (a==1)
      return b;
    else
      return 0;
}
```

```
Truth CosinicComplement(Truth a)
{
//  return (1.0+cos(PIE*a))/2.0;
}
Truth StandardComplement(Truth a)
{
  return 1-a;
}

Truth NOT(Truth x,Truth p=-0.5)
{
  return 1.0-x;
//  return (1.0-x)/(1.0+p*x);//  p in (-1,infinity)
//  return pow(1.0-pow(x,p),1.0/p);//   p in (0,infinity)
//  return x <=p ? 1 : 0;
//  return CosinicComplement(x);
}

Truth AND(Truth x1,Truth x2)
{
//  return DrasticIntersection(x1,x2);
  return Min(x1,x2);
}

Truth AND(Truth x1,Truth x2,Truth x3)
{
  return AND(AND(x1,x2),x3);
}

Truth AND(Truth x1,Truth x2,Truth x3,Truth x4)
{
  return AND(AND(x1,x2,x3),x4);
}

Truth AND(Truth x1,Truth x2,Truth x3,Truth x4,Truth
x5)
{
  return AND(AND(x1,x2,x3,x4),x5);
}

Truth AND(Truth x1,Truth x2,Truth x3,Truth x4,Truth
x5,Truth x6)
{
  return AND(AND(x1,x2,x3,x4,x5),x6);
}

Truth AND(Truth x1,Truth x2,Truth x3,Truth x4,Truth
x5,Truth x6,Truth x7)
{
  return AND(AND(x1,x2,x3,x4,x5,x6),x7);
}

Truth AND(Truth x1,Truth x2,Truth x3,Truth x4,Truth
x5,Truth x6,Truth x7,Truth x8)
{
  return AND(AND(x1,x2,x3,x4,x5,x6,x7),x8);
}
Truth OR(Truth x1,Truth x2)
{
  return Max(x1,x2);
}

Truth OR(Truth x1,Truth x2,Truth x3)
{
  return OR(OR(x1,x2),x3);
}

Truth AlphaCut(Truth Input,Truth thrs)
{
  if (Input >=thrs)
    return 1;
  else
    return 0;
}

Truth Binary2Bipolar(Truth t)
{
  return 2*t-1;
}
```

```
// Input range {0,1} -> final output value !
Truth TotalTruth(Truth t,Truth t0)   // determine fuzzy
mode,and cut
{
  return Binary2Bipolar(t);
}

#define  ASPIRIN              1
#define  NUMBER_OF_CLASSES      77
#define  PRINT_COMMENT          1

//  not chage value here,change in *.CFG.
int     FUZZY_MODE;
Truth   ALPHA_CUT;
Truth   P_S;// primitive vectors similartity for line
Truth   P_Z;// primitive vectors similartity for circle
Truth   Z_S;// zone sim.
Truth   L_S;// level sim.
Truth   FuzzyGreaterThan_Simirality;
Truth   FuzzyLessThan_Simirality;
Truth   FuzzyCountCircleEndPoints_Constant;
Truth   FuzzyCountEndPoints_Constant;
Truth   FuzzyCountPrimitives_OuterConstant;
Truth   FuzzyCountSections_Constant;
Truth   FuzzyCountStartZone_Constant;
Truth   InnerFuzzyListEquality_Constant;
Truth   OuterFuzzyListEquality_Constant;

Truth   LevelsRelation[5][5]=
{
/*    1,2,3,4,5  */
1,0,0,0,0,
0,1,-1,0,0,
0,-1,1,-1,0,
0,0,-1,1,0,
0,0,0,0,1
};

Truth VectorZonesRelation[8][8]=//
VectorZoneRelation,HeadVectorZoneRelation,EndVe
ctorZoneRelation use the same fuzzy relation.
{
1,-1,-1,-1,-1,-1,0,0,
 -1,1,-1,0,-1,0,-1,0,
 -1,-1,1,0,0,-1,-1,0,
 -1,0,0,1,-1,-1,0,-1,
 -1,-1,0,-1,1,0,0,-1,
 -1,0,-1,-1,0,1,0,0,
 0,-1,-1,0,0,0,1,0,
 0,0,0,-1,-1,0,0,1
};

Truth VectorTypesRelation[13][13]=
{
1,-1,0,0,0,0,0,-1,0,0,0,0,0,
 -1,1,-1,0,0,0,0,0,0,0,0,0,0,
 0,-1,1,-1,0,0,0,0,0,0,0,0,0,
 0,0,-1,1,-1,0,0,0,0,0,0,0,0,
 0,0,0,-1,1,-1,0,0,0,0,0,0,0,
 0,0,0,0,-1,1,-1,0,0,0,0,0,0,
 0,0,0,0,0,-1,1,-1,0,0,0,0,0,
 -1,0,0,0,0,0,-1,1,0,0,0,0,0,
 0,0,0,0,0,0,0,0,1,-2,-2,-2,-2,
 0,0,0,0,0,0,0,0,-2,1,-2,0,-2,
 0,0,0,0,0,0,0,0,-2,-2,1,-2,0,
 0,0,0,0,0,0,0,0,-2,0,-2,1,-2,
 0,0,0,0,0,0,0,0,-2,-2,0,-2,1
};

NaturalNO MAXIndex;

#define TRIVIAL

TRIVIAL char *B4(char *input0,char *token0)
{
  String input,token;

  strcpy(input,input0);
  strcpy(token,token0);
  String p;
  strcpy(p,strtok(input,token));

  return p;
}

TRIVIAL char *After(char *input,char token)
{
  char *p=&(input[0]);
  while (*p !=token)
    p++;
  return p+1;
}

TRIVIAL void SEND2BNN(Truth t)
{
  if (ASPIRIN)
  {
    cout<<TotalTruth(t,ALPHA_CUT);
    cout<<" ";
  }
  else  // To be used with exactly match ILP
  {
    Truth xxx=TotalTruth(t,ALPHA_CUT);
    if (xxx==1)
      cout<<"+";
    else
      cout<<"-";
  }
}

TRIVIAL void NULL_LINKS(NaturalNO n)
{
  for (int i=1;i <=n;i++)
    if (ASPIRIN)
      cout<<0<<" ";
    else
      cout<<" ";
}

Truth FuzzyAround(Truth Exact,Truth Desired,float
Const)
{
//  if (Exact==Desired)
//    return 1;
//  else
    return exp(-1*Const*abs(Desired-Exact));
}

Truth FuzzyGreaterThan(Truth Exact,Truth Desired)
{
  if (Exact > Desired)
    return 1.00;
  else
    if (Exact==Desired)
      return FuzzyGreaterThan_Simirality;
    else
      return 0;
}

Truth FuzzyGreaterThanOrEqual(Truth Exact,Truth
Desired)
{
  if (Exact >=Desired)
    return 1.00;
  else
    return 0;
}

Truth FuzzyLessThan(Truth Exact,Truth Desired)
{
  if (Exact < Desired)
    return 1;
  else
    if (Exact==Desired)
      return FuzzyLessThan_Simirality;
    else
      return 0;
}

Truth FuzzyLessThanOrEqual(Truth Exact,Truth
Desired)
{
  if (Exact <=Desired)
    return 1.00;
  else
    return 0.00;
}

Truth IsEndPoint(int SectionID)
{
  Truth xxx=-1*NOT(((SectionID-1) / 64) % 2);
  if (xxx < 0)
    return 1;
  else
    return 0;
}

Truth FuzzyLevel(LevelType Desired)
{
  return LevelsRelation[Level-1][Desired-1];
}

TRIVIAL template <class T> NaturalNO
ListSize(ListType<T> &aList)
{
  IteratorType<T> &I(aList);
  NaturalNO Size=0;
  while (I)
  {
    Size++;
    I++;
  }
  return Size;
}

NaturalNO CountSections()
{
//  cntsection([],0).
//  cntsection([A],1).
//  cntsection([A|B],C) :- cntsection(B,D),inc(D,C).
  return ListSize(C);
}

Truth FuzzyCountSections(NaturalNO Desired,Truth
Const=FuzzyCountSections_Constant)
{
  return FuzzyAround(CountSections(),Desired,Const);
}

NaturalNO CountZone(ListType<ZoneType>
&ZoneList)
{
  return ListSize(ZoneList);
}

TRIVIAL Truth IntervalMember(int x,int x1,int x2)
{
  return (x >=x1) && (x <=x2);
}

TRIVIAL NaturalNO H1(int SectionID)   // Sectionid
<=1024
{
  if (IntervalMember(SectionID,1,8))
    return 11;
  if (IntervalMember(SectionID,9,16))
    return 14;
  if (IntervalMember(SectionID,17,24))
    return 12;
  if (IntervalMember(SectionID,25,32))
    return 10;
  if (IntervalMember(SectionID,33,40))
    return 13;
  if (IntervalMember(SectionID,41,48))
    return 9;
  if (IntervalMember(SectionID,49,56))
    return 15;
  if (IntervalMember(SectionID,57,64))
    return 16;
  if (IntervalMember(SectionID,65,128))
    return 17;
  return H1(SectionID-(128-1+1));
}

TRIVIAL NaturalNO H2(int SectionID)   // 1025
<=SectionID <=1664
{
  if (IntervalMember(SectionID,1025,1032))
    return 4;
  if (IntervalMember(SectionID,1033,1040))
    return 6;
  if (IntervalMember(SectionID,1041,1048))
    return 2;
  if (IntervalMember(SectionID,1049,1056))
    return 3;
  if (IntervalMember(SectionID,1057,1064))
    return 5;
  if (IntervalMember(SectionID,1065,1072))
    return 1;
  if (IntervalMember(SectionID,1073,1080))
    return 7;
  if (IntervalMember(SectionID,1081,1088))
    return 8;
  if (IntervalMember(SectionID,1089,1152))
    return 17;
  return H2(SectionID-(1152-1025+1));
}

TRIVIAL NaturalNO HRank(int SectionID)
{
  if (SectionID <=1024)
    return H1(SectionID);
  else
    return H2(SectionID);
}

TRIVIAL char *List2Char()
{
  String s,inc;
  IteratorType<char>   I(List);

  I.restart();
  memset(s,0,sizeof(s));
  while (I)
  {
    sprintf(inc,"%c",I.current());
    strcat(s,inc);
    I++;
  }
  return s;
}

TRIVIAL template <class T> void
PrintList(ListType<T> &aList,char sep='!')
{
  IteratorType<T>   I(aList);
  I.restart();
  while (I)
  {
    cout<<I.current();
    if (sep !='!')
      cout<<sep;
    I++;
  }
  cout<<endl;
}

TRIVIAL void DeleteChar(char ToDel)
{
  for (int i=0;i < strlen(input);i++)
    List.detach(ToDel,TShouldDelete::Delete);
}

TRIVIAL void FindCharName()
{
  String inc;
  I.restart();
  memset(CharName,0,sizeof(CharName));
  while (I.current() !='(')
  {
    sprintf(inc,"%c",I.current());
    strcat(CharName,inc);
    I++;
  }
}


TRIVIAL void FindLevel()
{
  I++;
  Level=(I.current() - '0');
}

TRIVIAL void FindRatio()
{
  I++;
  I++;
  String inc;
  memset(RatioString,0,sizeof(RatioString));
  while (I.current() !=',')
  {
    sprintf(inc,"%c",I.current());
    strcat(RatioString,inc);
    I++;
  }
  Ratio=atof(RatioString);
}

TRIVIAL void FindComponents()
{
  C.flush();
  I++;
  I++;
  int   n=0;
  int   done=1;

  while (I.current() !=']')
  {
    if (I.current()==',')
    {
      C.addAtTail(n);
      n=0;
      done=1;
    }
    if (isdigit(I.current()))
    {
      n *=10;
      n +=I.current() - '0';
      done=0;
    }
    I++;
  }
  if (done==0)
    C.addAtTail(n);
}

TRIVIAL void FindJunctions()
{
  J.flush();
  I++;
  I++;
  int   n=0;
  int   done=1;

  while (I.current() !=']')
  {
    if (I.current()==',')
    {
      J.addAtTail(n);
      n=0;
      done=1;
    }
    if (isdigit(I.current()))
    {
      n *=10;
      n +=I.current()  - '0';
      done=0;
    }
    I++;
  }
  if (done==0)
    J.addAtTail(n);
}

TRIVIAL void FindUpperYuks()
{
  U.flush();
  I++;
  I++;
  int   n=0;
  int   done=1;

  while (I.current() !=']')
  {
    if (I.current()==',')
    {
      U.addAtTail(n);
      n=0;
      done=1;
    }
    if (isdigit(I.current()))
    {
      n *=10;
      n +=I.current()  - '0';
      done=0;
    }
    I++;
  }
  if (done==0)
    U.addAtTail(n);
}

TRIVIAL void FindLowerYuks()
{
  L.flush();
  I++;
  I++;
  int   n=0;
  int   done=1;

  while (I.current() !=']')
  {
    if (I.current()==',')
    {
      L.addAtTail(n);
      n=0;
      done=1;
    }
    if (isdigit(I.current()))
    {
      n *=10;
      n +=I.current()-'0';
      done=0;
    }
    I++;
  }
  if (done==0)
    L.addAtTail(n);
}

PrimitiveType CrispPrimitive(SectionIDType
SectionID)
{
  return ((SectionID-1) / 128) % 13;
}

Truth FuzzyPrimitive(SectionIDType
SectionID,PrimitiveType  Desired)
{
  return
VectorTypesRelation[CrispPrimitive(SectionID)][Desir
ed];
}

ZoneType CrispStartZone(SectionIDType SectionID)
{
  return ((SectionID-1) / 8) % 8;
}

Truth FuzzyZone(ZoneType X,ZoneType Y)
{
  return VectorZonesRelation[X][Y];
}

Truth FuzzyStartZone(SectionIDType
SectionID,ZoneType Desired)
{
  return
FuzzyZone(CrispStartZone(SectionID),Desired);
}

ZoneType CrispEndZone(SectionIDType SectionID)
{
  return ((SectionID-1) / 1) % 8;
}

Truth FuzzyEndZone(SectionIDType
SectionID,ZoneType Desired)
{
  return
FuzzyZone(CrispEndZone(SectionID),Desired);
}

TRIVIAL char YESNO(Truth yn)
{
  return yn==YES ? 'Y' : 'N';
}

Truth IsCircle(SectionIDType x)
{
  if ((8 <=CrispPrimitive(x)) && (CrispPrimitive(x)
<=12))
    return 1;
  else
    return 0;
}

TRIVIAL void BuildList(char *t)
{
  List.flush();
  for (char *p=t;*p !=NULL;p++)
    List.addAtTail(*p);
}

TRIVIAL void Split()
{
  IteratorType<int>   I(C);
  I.restart();

cout<<"primitive:"<<"\t"<<"endpoint:"<<"\t"<<"startzon
e:"<<"\t"<<"endzone:"<<"\t"<<"hrank:"<<endl;
  cout<<"--------- "<<"\t"<<"-------- "<<"\t"<<"---------
"<<"\t"<<"------- "<<"\t"<<"------"<<endl;
  int ComponentsCount=0;
  while (I)
  {
    ComponentsCount++;
    cout<<"["<<ComponentsCount<<"]";
    cout<<CrispPrimitive(I.current())<<"\t\t";
    cout<<YESNO(IsEndPoint(I.current()))<<"\t\t";
    cout<<CrispStartZone(I.current())<<"\t\t";
    cout<<CrispEndZone(I.current())<<"\t\t";
    cout<<HRank(I.current());
    cout<<endl;
    I++;
  }
}

TRIVIAL void FindHead()
{
  IteratorType<SectionIDType> I(C);

  I.restart();
  HeadSectionID=I.current();
  while (I)
  {
    I++;
    if (HRank(I.current()) < HRank(HeadSectionID))
      HeadSectionID=I.current();
  }
}

Truth FuzzyHeadZone(ZoneType Desired)
{
  return
FuzzyZone(CrispStartZone(HeadSectionID),Desired);
}

Truth FuzzyHeadPrimitive(PrimitiveType Desired)
{
  return
VectorTypesRelation[CrispPrimitive(HeadSectionID)][
Desired];
}

Truth FuzzyEndPointTopRight()
{
//  enpt_topright(A) :- member(B,A),endpoint(B,-
1),(startzone(B,1) or startzone(B,2)),endzone(B,1).
  IteratorType<SectionIDType> I(C);
  Truth MaxTruth=0;
  I.restart();
  while (I)
  {
    Truth each=AND(IsEndPoint(I.current()),
                  FuzzyEndZone(I.current(),1),

OR(FuzzyStartZone(I.current(),1),FuzzyStartZone(I.cu
rrent(),2))
);
```

```
    if (each > MaxTruth)
      MaxTruth=each;
    I++;
  }
  return MaxTruth;
}

Truth FuzzyTopRightTail()
{
//  topright_tail(A) :- member(B,A),endpoint(B,-
1),startzone(B,1),endzone(B,1),<primitive(B,0) OR
primitive(B,1)>.
  IteratorType<SectionIDType> I(C);
  Truth MaxTruth=0;
  I.restart();
  while (I)
  {
    Truth each=AND(IsEndPoint(I.current()),
                   FuzzyStartZone(I.current(),1),
                   FuzzyEndZone(I.current(),1),

OR(FuzzyPrimitive(I.current(),0),FuzzyPrimitive(I.curre
nt(),1))
);
    if (each > MaxTruth)
      MaxTruth=each;
    I++;
  }
  return MaxTruth;
}

Truth FuzzyEndPointPrimitive(ZoneType Desired)
//  enptprim(A,B) :- member(C,A),endpoint(C,-
1),primitive(C,B).
{
  IteratorType<SectionIDType> I(C);
  Truth MaxTruth=0;

  I.restart();
  while (I)
  {
    Truth each=AND(IsEndPoint(I.current()),
                   FuzzyPrimitive(I.current(),Desired)
);
    if (each > MaxTruth)
      MaxTruth=each;
    I++;
  }
  return MaxTruth;
}

Truth FuzzyEndPointZone(ZoneType Desired)
{
//  EndPointZone(L,Z) :- member(x,L),endpoint(x,-
1),startzone(x,Z).

  IteratorType<SectionIDType> I(C);
  Truth MaxTruth=0;

  I.restart();
  while (I)
  {
    Truth
each=AND(FuzzyStartZone(I.current(),Desired),
                   IsEndPoint(I.current())
);

    if (each > MaxTruth)
      MaxTruth=each;
    I++;
  }
  return MaxTruth;
}

Truth FuzzyRightLine()
{
//right_line(L) :- member(x,L),endpoint(x,-
1),endzone(x,1),(primitive(x,1) OR primitive(x,2)).

  IteratorType<SectionIDType> I(C);
  Truth MaxTruth=0;

  I.restart();
  while (I)
  {
    Truth each=AND(IsEndPoint(I.current()),
                   FuzzyEndZone(I.current(),1),
                   OR(FuzzyPrimitive(I.current(),1),
                      FuzzyPrimitive(I.current(),2)
)
);
    if (each > MaxTruth)
      MaxTruth=each;
    I++;
  }
  return MaxTruth;
}

TRIVIAL void FlushAll()
{
  List.flush();
  C.flush();
  J.flush();
  L.flush();
  U.flush();
}

NaturalNO CountEndPoints()
{
//  cntenpt([],0).
//  cntenpt([A|B],C) :- endpoint(A,-
1),cntenpt(B,D),inc(D,C).
//  cntenpt([A|B],C) :- endpoint(A,0),cntenpt(B,C).

  IteratorType<SectionIDType> I(C);
  NaturalNO SUM=0;

  I.restart();
  while (I)
  {
    if (IsEndPoint(I.current()))
      SUM++;
    I++;
  }
  return SUM;
}

Truth FuzzyCountEndPoints(NaturalNO Desired,Truth
Const=FuzzyCountEndPoints_Constant)
{
  return
FuzzyAround(CountEndPoints(),Desired,Const);
}

Truth IsCircleEndPoint(SectionIDType Section)
{
//  iscircenpt(A) :- primitive(A,B),endpoint(A,-
1),iscircle(B).
  return AND(IsCircle(Section),IsEndPoint(Section));
}

NaturalNO CountCircleEndPoints()
{
//  cntcircenpt([],0).
//  cntcircenpt([A|B],C) :-
iscircenpt(A),cntcircenpt(B,D),inc(D,C).
//  cntcircenpt([A|B],C) :- not
iscircenpt(A),cntcircenpt(B,C).
  IteratorType<SectionIDType> I(C);
  int EndPoints=0;

  I.restart();
  while (I)
  {
    if (IsCircleEndPoint(I.current()))
      EndPoints++;
    I++;
  }
  return EndPoints;
}

Truth FuzzyCountCircleEndPoints(NaturalNO
Desired,Truth
Const=FuzzyCountCircleEndPoints_Constant)
{
  return
FuzzyAround(CountCircleEndPoints(),Desired,Const);
}

template <class T> Truth CrispMember(ListType<T>
&aList,T toFind)
{
//  member(A,[A|B]).
//  member(A,[B|C]) :- member(A,C).

  IteratorType<T>  I(aList);

  I.restart();
  while (I)
  {
    if (I.current()==toFind)
      return YES;
    else
      I++;
  }
  return 0;// include empty list !
}

template <class T> T LastMember(ListType<T>
&aList)
{
//  lastmember([A],A).
//  lastmember([A|B],C) :- lastmember(B,C).
  if (ListSize(aList)==0)
    return NULL;
  else
  {
    IteratorType<T> I(aList);
    I.restart();
    for (int i=1;i <=ListSize(aList)-1;i++)
      I++;
    return I.current();
  }
}

Truth FuzzyBeginEndZone(ZoneType
DesiredBeginZone,ZoneType DesiredEndZone)
//  begendzone([A|B],C,D) :-
lastmember(B,E),startzone(E,C),endzone(A,D).
{
  IteratorType<SectionIDType>     I(C);
  I.restart();
  return
AND(FuzzyEndZone(I.current(),DesiredEndZone),

FuzzyStartZone(LastMember(C),DesiredBeginZone)
);
}

Truth FuzzyCircleEndPointPrimitive(PrimitiveType
Desired)
//    circle_endpoint_prim(List,Desired) :-
member(C,List),iscircenpt(C),primitive(C,Desired).
{
  IteratorType<SectionIDType> I(C);
  Truth MaxTruth=0;

  I.restart();
  while (I)
  {
    Truth each=AND(IsCircleEndPoint(I.current()),
                   FuzzyPrimitive(I.current(),Desired)
);
    if (each > MaxTruth)
      MaxTruth=each;
    I++;
  }
  return MaxTruth;
}

Truth FuzzyCircleEndPointZone(ZoneType Desired)
//    circenptzone(A,B) :-
member(C,A),iscircenpt(C),startzone(C,B).
{
  IteratorType<SectionIDType> I(C);
  Truth MaxTruth=0;

  I.restart();
  while (I)
  {
    Truth
each=AND(FuzzyStartZone(I.current(),Desired),
                   IsCircleEndPoint(I.current())
);
    if (each > MaxTruth)
      MaxTruth=each;
    I++;
  }
  return MaxTruth;
}

Truth FuzzyListEquality(NaturalNO X[8],NaturalNO
Y[8],
                    Truth
Const1=InnerFuzzyListEquality_Constant,
                    Truth
Const2=OuterFuzzyListEquality_Constant)
{
  if ((X[0]==Y[0]) &&
       (X[1]==Y[1]) &&
       (X[2]==Y[2]) &&
       (X[3]==Y[3]) &&
       (X[4]==Y[4]) &&
       (X[5]==Y[5]) &&
       (X[6]==Y[6]) &&
       (X[7]==Y[7])
)
    return 1;
  else
  {
    Truth SigmaCount=FuzzyAround(X[0],Y[0],Const1)
+
                   FuzzyAround(X[1],Y[1],Const1) +
                   FuzzyAround(X[2],Y[2],Const1) +
                   FuzzyAround(X[3],Y[3],Const1) +
                   FuzzyAround(X[4],Y[4],Const1) +
                   FuzzyAround(X[5],Y[5],Const1) +
                   FuzzyAround(X[6],Y[6],Const1) +
                   FuzzyAround(X[7],Y[7],Const1);
    return FuzzyAround(SigmaCount,8.00,Const2);
  }
}


Truth J_(NaturalNO z0,NaturalNO z1,NaturalNO
z2,NaturalNO z3,
        NaturalNO z4,NaturalNO z5,NaturalNO
z6,NaturalNO z7)
{
  NaturalNO   Temp[8];

  Temp[0]=z0;
  Temp[1]=z1;
  Temp[2]=z2;
  Temp[3]=z3;
  Temp[4]=z4;
  Temp[5]=z5;
  Temp[6]=z6;
  Temp[7]=z7;

  return FuzzyListEquality(Temp,CountZone_J);
}


Truth L_(NaturalNO z0,NaturalNO z1,NaturalNO
z2,NaturalNO z3,
        NaturalNO z4,NaturalNO z5,NaturalNO
z6,NaturalNO z7)
{
  NaturalNO   Temp[8];

  Temp[0]=z0;
  Temp[1]=z1;
  Temp[2]=z2;
  Temp[3]=z3;
  Temp[4]=z4;
  Temp[5]=z5;
  Temp[6]=z6;
  Temp[7]=z7;

  return FuzzyListEquality(Temp,CountZone_L);
}

Truth U_(NaturalNO z0,NaturalNO z1,NaturalNO
z2,NaturalNO z3,
        NaturalNO z4,NaturalNO z5,NaturalNO
z6,NaturalNO z7)
{
  NaturalNO   Temp[8];

  Temp[0]=z0;
  Temp[1]=z1;
  Temp[2]=z2;
  Temp[3]=z3;
```

```
  Temp[4]=z4;
  Temp[5]=z5;
  Temp[6]=z6;
  Temp[7]=z7;

  return FuzzyListEquality(Temp,CountZone_U);
}

Truth FuzzyHaveXXXX(PrimitiveType
DesiredPrimitive,NaturalNO dummy,ZoneType
DesiredStartZone,ZoneType DesiredEndZone)
{
// havemember(A,B,C,D) :-
member(E,A),primitive(E,B),endpoint(B,0),startzone(E
,C),endzone(E,D).
  IteratorType<SectionIDType> I(C);
  Truth MaxTruth=0;

  I.restart();
  while (I)
  {
    Truth each=AND(NOT(IsEndPoint(I.current())),

FuzzyPrimitive(I.current(),DesiredPrimitive),

FuzzyStartZone(I.current(),DesiredStartZone),

FuzzyEndZone(I.current(),DesiredEndZone)
);
    if (each > MaxTruth)
      MaxTruth=each;
    I++;
  }
  return MaxTruth;
}

Truth FuzzyHaveMember(PrimitiveType
DesiredPrimitive,ZoneType
DesiredStartZone,ZoneType DesiredEndZone)
{
// havemember(A,B,C,D) :-
member(E,A),primitive(E,B),startzone(E,C),endzone(E
,D).
  IteratorType<SectionIDType> I(C);
  Truth MaxTruth=0;

  I.restart();
  while (I)
  {
    Truth
each=AND(FuzzyPrimitive(I.current(),DesiredPrimitive
),

FuzzyStartZone(I.current(),DesiredStartZone),

FuzzyEndZone(I.current(),DesiredEndZone)
);

    if (each > MaxTruth)
      MaxTruth=each;
    I++;
  }
  return MaxTruth;
}

Truth FuzzyMemberZone(ZoneType
DesiredStartZone,ZoneType DesiredEndZone)
//    memberzone(A,B,C) :-
member(E,A),startzone(E,B),endzone(E,C).
{
  IteratorType<SectionIDType> I(C);
  Truth MaxTruth=0;

  I.restart();
  while (I)
  {
    Truth
each=AND(FuzzyStartZone(I.current(),DesiredStartZo
ne),

FuzzyEndZone(I.current(),DesiredEndZone)
);

    if (each > MaxTruth)
      MaxTruth=each;
    I++;
```

```
  }
  return MaxTruth;
}

template <class T> Truth FuzzyNoZone(ListType<T>
&aList)
{
//  nozone([]).
  IteratorType<T>   I(aList);

  I.restart();
  if (I)
    return 0;
  else
    return 1.00;
}


template <class T> Truth FuzzyNoZone(ListType<T>
&aList,ZoneType X)
{
//  nozoneX(A) :- not havezone(X,z0).
  return NOT(FuzzyHaveZone(aList,X));
}

Truth FuzzyBottomRightTail()
{
// bottomright_tail(A) :-  member(B,A),endpoint(B,-
1),endzone(B,4),
//                    primitive(B,5).
// bottomright_tail(A) :-  member(B,A),endpoint(B,-
1),endzone(B,4),
//                    primitive(B,6).
  IteratorType<SectionIDType> I(C);
  Truth MaxTruth=0;

  I.restart();
  while (I)
  {
    Truth each=AND(IsEndPoint(I.current()),
                   FuzzyEndZone(I.current(),4),

OR(FuzzyPrimitive(I.current(),5),FuzzyPrimitive(I.curre
nt(),6))
);
    if (each > MaxTruth)
      MaxTruth=each;
    I++;
  }
  return MaxTruth;
}

Truth FuzzyCountPrimitives(PrimitiveType
P,NaturalNO Desired,Truth
OuterConst=FuzzyCountPrimitives_OuterConstant)
{
// cntprimX([],0).
// cntprimX([A|B],C) :-
primitive(A,X),cntprimX(B,D),inc(D,C).
// cntprimX([A|B],C) :- not
primitive(A,X),cntprimX(B,C).

  IteratorType<SectionIDType> I(C);
  NaturalNO SigmaCount=0;

  I.restart();
  while (I)
  {
    SigmaCount +=FuzzyPrimitive(I.current(),P);
    I++;
  }
  return
FuzzyAround(SigmaCount,Desired,OuterConst);
}

Truth FuzzyCountStartZone(ZoneType Z,NaturalNO
Desired,Truth
Const=FuzzyCountStartZone_Constant)
{
//  cntstzoneX([],0).
//  cntstzoneX([A|B],C) :-
startzone(A,X),cntstzoneX(B,D),inc(D,C).
//  cntstzoneX([A|B],C) :- not
startzone(A,X),cntstzoneX(B,C).

  IteratorType<SectionIDType> I(C);
  NaturalNO SUM=0;

  I.restart();
  while (I)
  {
    SUM +=FuzzyStartZone(I.current(),Z);
    I++;
  }
  return FuzzyAround(SUM,Desired,Const);
}

Truth FuzzyTopLeftTail()
{
// topleft_tail(A) :- member(B,A),endpoint(B,-
1),endzone(B,2),primitive(B,4).
// topleft_tail(A) :- member(B,A),endpoint(B,-
1),endzone(B,2),primitive(B,5).
  IteratorType<SectionIDType> I(C);
  Truth MaxTruth=0;

  I.restart();
  while (I)
  {
    Truth each=AND(IsEndPoint(I.current()),
                  FuzzyEndZone(I.current(),2),

OR(FuzzyPrimitive(I.current(),4),FuzzyPrimitive(I.curre
nt(),5))
);
    if (each > MaxTruth)
      MaxTruth=each;
    I++;
  }
  return MaxTruth;
}

Truth FuzzyHaveZone(ListType<ZoneType>
&aList,ZoneType Z)
{
//  havezone(A,B) :- member(B,A).

  IteratorType<ZoneType>  I(aList);
  I.restart();
  if (!I)
    return 0;// empty list
  else
  {
    Truth MAX=FuzzyZone(I.current(),Z);
    while (I)
    {
      Truth each=FuzzyZone(I.current(),Z);
      if (each > MAX)
        MAX=each;
      I++;
    }
    return MAX;
  }
}

Truth FuzzyHeadYuk(ListType<ZoneType> &aList)
{
// headyuk(A) :- member(z2,A).
  return FuzzyHaveZone(aList,2);
}

Truth FuzzyUpperYuk(ListType<ZoneType> &aList)
{
//    upperyuk(A) :- member(z1,A).
//    upperyuk(A) :- member(z2,A).
  return
OR(FuzzyHaveZone(aList,1),FuzzyHaveZone(aList,2)
);
}

// topline(A) :-
member(B,A),endpoint(B,0),primitive(B,0),endzone(B,
1),startzone(B,1)
// topline(A) :-
member(B,A),endpoint(B,0),primitive(B,0),endzone(B,
1),startzone(B,2)

Truth FuzzyTopLine()
{
/*
topline(A) :-
member(B,A),primitive(B,0),endpoint(B,0),startzone(B,
1),
            endzone(B,1).
topline(A) :-
member(B,A),primitive(B,0),endpoint(B,0),startzone(B,
2),
            endzone(B,1).
*/
  IteratorType<SectionIDType> I(C);
  Truth MaxTruth=0;
  I.restart();
  while (I)
  {
    Truth each=AND(FuzzyPrimitive(I.current(),0),
                  NOT(IsEndPoint(I.current())),

OR(FuzzyStartZone(I.current(),1),FuzzyStartZone(I.cu
rrent(),2)),
                  FuzzyEndZone(I.current(),1)
);
    if (each > MaxTruth)
      MaxTruth=each;
    I++;
  }
  return MaxTruth;
}


#include "i.knb" // i can be e.knb,z.knb

void BuildDecisionList()
{
  kai();
  khai();
  khud();
  khay();
  khon();
  rakung();
  ngoo();
  jan();
  ching();
  chang();
  soo();
  kacher();
  ying();
  chada();
  patak();
  than();
  monto();
  putoa();
  nen();
  deg();
  toa();
  tung();
  tahan();
  tong();
  nuu();
  bimai();
  pla();
  peung();
  fa();
  pan();
  fan();
  sampoa();
  mar();
  yag();
  rua();
  rue();
  ling();
  van();
  sala();
  ruesi();
  sua();
  heep();
  chula();
  ang();
  huug();
  paiyan();
  sra_at();
  mi_hanka();
  sra_ar();
  sra_ei();
  sra_ee();
  sra_eut();
  sra_eu();
  sra_ut();
  sra_uu();
  sra_a();
  sra_o();
  sra_aio();
  sra_aim();
  yamok();
  mi_tikoo();
  mi_ek();
  mi_to();
  mi_tee();
  mi_jatva();
  karan();
  sra_am();
  num_0();
  num_1();
  num_2();
  num_3();
  num_4();
  num_5();
  num_6();
  num_7();
  num_8();
  num_9();
}

void BuildCharsAssoc()
{
  NaturalNO n=0;
  strcpy(CharsAssoc[n++],"kai");
  strcpy(CharsAssoc[n++],"khai");
  .
  .
  .
  strcpy(CharsAssoc[n++],"num_9");
}

int Decision()
{
  NaturalNO MAXValue=0;

  for (NaturalNO i=0;i <=249-161;i++)
    if (DecisionList[i] > MAXValue)
      MAXValue=DecisionList[i];
  for (MAXIndex=0;MAXIndex <=249-
161;MAXIndex++)
    if (DecisionList[MAXIndex]==MAXValue)
      break;
  if (strcmp(CharsAssoc[MAXIndex],CharName)==0)
    return 1;
  else
    return 0;
}

void DebugCharInfo()
{
  cout<<"CHARACTER NAME : "<<CharName<<endl;
  cout<<"LEVEL     : "<<Level<<endl;
  cout<<"RATIO     : "<<Ratio<<endl;
  cout<<"COMPONENTS     : ";PrintList(C,' ');
// Split();
  cout<<"JUNCTIONS      : ";PrintList(J,' ');
  cout<<"LOWERYUKS      : ";PrintList(L,' ');
  cout<<"UPPERYUKS      : ";PrintList(U,' ');
  cout<<"HEAD ZONE      :
"<<CrispStartZone(HeadSectionID)<<endl;
  cout<<"HEAD PRIM      :
"<<CrispPrimitive(HeadSectionID)<<endl;
}

void FindBasicComponents()
{
  FlushAll();
  BuildList(input);
  DeleteChar(' ');
  DeleteChar('z');
  FindCharName();
  FindLevel();
  FindRatio();
  FindComponents();
  FindJunctions();
  FindLowerYuks();
  FindUpperYuks();

  memset(CountZone_J,0,sizeof(CountZone_J));
  memset(CountZone_L,0,sizeof(CountZone_L));
```

```
 memset(CountZone_U,0,sizeof(CountZone_U));

 FindHead();
 {
   IteratorType<ZoneType> I(J);
   I.restart();
   while (I)
   {
     CountZone_J[I.current()]++;
     I++;
   }
 }
 {
   IteratorType<ZoneType> I(L);
   I.restart();
   while (I)
   {
     CountZone_L[I.current()]++;
     I++;
   }
 }
 {
   IteratorType<ZoneType> I(U);
   I.restart();
   while (I)
   {
     CountZone_U[I.current()]++;
     I++;
   }
 }
}

NaturalNO RecogFact(const char *ToRecog)
{
 strcpy(input,ToRecog);
 FindBasicComponents();
 if (DEBUG_CHAR_INFO_MODE==YES)
 {
   DebugCharInfo();
   PrintList(List);
 }
 if (PRINT_COMMENT)
   cout<<"/*"<<CharName<<"*/";
 BuildDecisionList();
 NaturalNO n=Decision();
 FlushAll();

 return n;
}

Truth IsCharacterName(char *s)
{
 for (NaturalNO i=0;i <=sizeof(CharsAssoc[i]);i++)
   if (strcmp(s,CharsAssoc[i]) !=0)
     return YES;
 return NO;
}

void GenerateRule(SmallString C)
{
 if (ASPIRIN)
   cout<<endl;
 else
   cout<<",";
 for (NaturalNO i=0;i <=NUMBER_OF_CLASSES-
1;i++)  // don't forget must -1 !
   if (strcmp(CharsAssoc[i],C)==0)
     break;
 int ClassBar[NUMBER_OF_CLASSES];
 memset(ClassBar,0,sizeof(ClassBar));
 ClassBar[i]=1;
 if (PRINT_COMMENT)
   cout<<"/*"<<CharName<<"*/";
 for (int k=0;k <=NUMBER_OF_CLASSES-1;k++)
 {
   cout<<ClassBar[k];
   if (k !=NUMBER_OF_CLASSES-1)
     if (ASPIRIN)
       cout<<" ";
 }
}

int main(int argc,char **argv)
{
  if (argc < 4)
 {
   cout<<"Using p77.exe e.knb e.ilp 1.0 c"<<endl;
   exit(-1);
 }
 DEBUG_CHAR_INFO_MODE=NO;

 cout<<"/*"<<endl;
 Date d;
 Time t;
 cout<<d<<endl;
 cout<<t<<endl;

 ifstream CFGFILE("P77.CFG");
 String CFG_Line;

 CFGFILE.getline(CFG_Line,sizeof(CFG_Line));
 FUZZY_MODE=atof(B4(After(CFG_Line,'='),";"));

 CFGFILE.getline(CFG_Line,sizeof(CFG_Line));
 ALPHA_CUT=atof(B4(After(CFG_Line,'='),";"));

 CFGFILE.getline(CFG_Line,sizeof(CFG_Line));
 P_S=atof(B4(After(CFG_Line,'='),";"));

 CFGFILE.getline(CFG_Line,sizeof(CFG_Line));
 P_Z=atof(B4(After(CFG_Line,'='),";"));

 CFGFILE.getline(CFG_Line,sizeof(CFG_Line));
 Z_S=atof(B4(After(CFG_Line,'='),";"));

 CFGFILE.getline(CFG_Line,sizeof(CFG_Line));
 L_S=atof(B4(After(CFG_Line,'='),";"));

 CFGFILE.getline(CFG_Line,sizeof(CFG_Line));

FuzzyGreaterThan_Simirality=atof(B4(After(CFG_Line
,'='),";"));

 CFGFILE.getline(CFG_Line,sizeof(CFG_Line));

FuzzyLessThan_Simirality=atof(B4(After(CFG_Line,'='
),";"));

 CFGFILE.getline(CFG_Line,sizeof(CFG_Line));

FuzzyCountCircleEndPoints_Constant=atof(B4(After(
CFG_Line,'='),";"));

 CFGFILE.getline(CFG_Line,sizeof(CFG_Line));

FuzzyCountEndPoints_Constant=atof(B4(After(CFG_
Line,'='),";"));

 CFGFILE.getline(CFG_Line,sizeof(CFG_Line));

FuzzyCountPrimitives_OuterConstant=atof(B4(After(C
FG_Line,'='),";"));

 CFGFILE.getline(CFG_Line,sizeof(CFG_Line));

FuzzyCountSections_Constant=atof(B4(After(CFG_Li
ne,'='),";"));

 CFGFILE.getline(CFG_Line,sizeof(CFG_Line));

FuzzyCountStartZone_Constant=atof(B4(After(CFG_L
ine,'='),";"));

 CFGFILE.getline(CFG_Line,sizeof(CFG_Line));

InnerFuzzyListEquality_Constant=atof(B4(After(CFG_
Line,'='),";"));

 CFGFILE.getline(CFG_Line,sizeof(CFG_Line));

OuterFuzzyListEquality_Constant=atof(B4(After(CFG_
Line,'='),";"));

 system("type p77.cfg");

 CFGFILE.close();

 cout<<"*/"<<endl;

 int i,j;
 for (i=0;i <=4;i++)
 {
   for (j=0;j <=4;j++)
     if (i !=j)
       if (LevelsRelation[i][j]==-1)
         LevelsRelation[i][j]=L_S;
 }
 for (i=0;i <=7;i++)
 {
   for (j=0;j <=7;j++)
     if (i !=j)
       if (VectorZonesRelation[i][j]==-1)
         VectorZonesRelation[i][j]=Z_S;
 }
for (i=0;i <=12;i++)
 {
   for (j=0;j <=12;j++)
     if (i !=j)
     {
       if (VectorTypesRelation[i][j]==-1)
         VectorTypesRelation[i][j]=P_S;
       if (VectorTypesRelation[i][j]==-2)
         VectorTypesRelation[i][j]=P_Z;
     }
 }
 BuildCharsAssoc();
 ifstream ifs(argv[1]);
 String TestData;

 while (!ifs.eof())
 {
   ifs.getline(TestData,sizeof(TestData));
   if ((strcmp(TestData,"") !=0) && (TestData[0] !='\%')
&& (TestData[0] !='\\'))
   {
     RecogFact(TestData);
     GenerateRule(CharName);
     cout<<endl;
   }
 }
 ifs.close();
 return 0;
}
```

APPENDIX C
CHARACTERS OF FONTS EUCROSIA AND CORDIA.

ASCII CODE	*Eucroria*	*Cordia*
161	ก	ก
162	ข	ข
163	ฃ	ฃ
164	ค	ค
165	ฅ	ฅ
166	ฆ	ฆ
167	ง	ง
168	จ	จ
169	ฉ	ฉ
170	ช	ช
171	ซ	ซ
172	ฌ	ฌ
173	ญ	ญ
174	ฎ	ฎ
175	ฏ	ฏ
176	ฐ	ฐ
177	ฑ	ฑ
178	ฒ	ฒ
179	ณ	ณ
180	ด	ด
181	ต	ต
182	ถ	ถ
183	ท	ท
184	ธ	ธ
185	น	น
186	บ	บ
187	ป	ป
188	ผ	ผ
189	ฝ	ฝ
190	พ	พ
191	ฟ	ฟ
192	ภ	ภ
193	ม	ม
194	ย	ย
195	ร	ร
196	ฤ	ฤ
197	ล	ล
199	ว	ว
200	ศ	ศ
201	ษ	ษ
202	ส	ส
203	ห	ห
204	ฬ	ฬ
205	อ	อ
206	ฮ	ฮ
207	ฯ	ฯ
208	ะ	ะ
209	ั	ั
210	า	า
212	ิ	ิ
213	ี	ี
214	ึ	ึ
215	ื	ื
216	ุ	ุ
217	ู	ู
224	เ	เ
226	โ	โ
227	ใ	ใ
228	ไ	ไ
230	ๆ	ๆ
231	็	็
232	่	่
233	้	้
234	๊	๊
235	๋	๋
236	์	์
237	ํ	ํ
240	๐	๐
241	๑	๑
242	๒	๒
243	๓	๓
244	๔	๔
245	๕	๕
246	๖	๖
247	๗	๗
248	๘	๘
249	๙	๙

BIOGRAPHY

Mr.Jun Srisutapan was born on the 21th of August 2520. His oldname is วสันต์. He had his Mattayom education at WadSuthiWararam School. He graduated with the Bachelor Degree in Mathematics from the Department of Mathematics, Faculty of Science, King Mongkut's Institute of Technology - Thonburi.