

GENEL AMAÇLI BİR BULANIK UZMAN SİSTEM

YÜKSEK LİSANS TEZİ
Mat. Müh. Yusuf Barbaros YONAR
0995Y064

100627 100627

Tezin Enstitüye Verildiği Tarih : 31 Mayıs 1999
Tezin Savunulduğu Tarih : 18 Haziran 1999

Tez Danışmanı : Doç. Dr. Gazanfer ÜNAL
Diğer Jüri Üyeleri Prof. Dr. Galip TEPEHAN
Doç. Dr. Şakir KOCABAŞ

[Handwritten signatures]
G. Ünal
G. Tepehan
Ş. Kocabaş

HAZİRAN 1999

ÖNSÖZ

Bu çalışmanın ortaya çıkmasında en az benim kişisel çabam kadar, yakın çevremdekilerin de büyük katkısı var.

Öncelikle Anneme ve Babama sonsuz minnetlerimi sunuyorum. Herşey için !

Berna ve Yavuz. Sabır ve anlayışınız için sonsuz teşekkürler !

Onur ! Tezimi, yardımların sayesinde tamamlayabildim. Sağol !

Talin ! Bu konudaki tecrüben Bana çok yardımcı oldu. Çok teşekkürler !

Ve Çiğdem. Sen olmasaydın tüm bunların hiçbiri olmazdı.

Bana, lisans ve yüksek lisans öğrenimim boyunca her türlü desteği veren Sayın Hocam Doç. Dr. Gazanfer Ünal'a teşekkürlerimi sunuyorum.

Mayıs 1999

Yusuf Barbaros YONAR

İÇİNDEKİLER

KISALTMALAR	v
TABLO LİSTESİ	vi
ŞEKİL LİSTESİ	vii
ÖZET	viii
SUMMARY	xii
1. BİLGİ VE BELİRSİZLİK	1
1.1. Bilginin Düzeyleri	1
1.2. Bilginin Bileşenleri	3
1.3. Belirsizlik	3
2. BİLGİNİN TEMSİL YÖNTEMLERİ	6
2.1. Gerçekler	6
2.2. Kurallar	7
2.3. Hiyerarşi, Çerçeveler ve Kalıtım	7
2.3.1. Hiyerarşi	7
2.3.2. Çerçeveler	8
2.3.3. Kalıtım	9
2.4. Belirginlik Çarpanları	10
3. UZMAN SİSTEMLER	12
3.1. Uzman Sistemin Öğeleri	13
3.2. Uzman Sistemlerin Yapısı ve Çalışma İlkeleri	15
4. ÇIKARIM YÖNTEMLERİ	17
4.1. Çıkarım Yöntemleri	18
4.1.1. İleri Yönlü Çıkarım	22
4.1.2. Geri Yönlü Çıkarım	23
5. BULANIK KÜMELER	25
5.1. Belirgin Kümeler ve Bulanık Kümeler	26
5.1.1. Belirgin Kümeler	26
5.1.2. Bulanık Kümeler	27
5.1.2.1. Üyelik Fonksiyonunun Özellikleri	29
5.1.2.2. Bulanık Kümelerle İlgili İşlemler	31
5.1.2.3. Bulanık Kümelerin Özellikleri	32

6. BELİRGİN BAĞINTILAR VE BULANIK BAĞINTILAR	35
6.1. Belirgin Bağıntılar	35
6.1.1. Belirgin Bağıntı İşlemleri	36
6.1.2. Belirgin Bağıntılarının Kompozisyonu	37
6.2. Bulanık Bağıntılar	37
6.2.1. Bulanık Bağıntı İşlemleri	38
6.2.2. Bulanık Bağıntılarının Kompozisyonu	38
7. BULANIK MANTIK	39
7.1. Yaklaşım ile Çıkarım	40
7.2. Dilsel Pekiştiriciler	43
8. AĞIRLIKLIL BULANIK ÇIKARIM ALGORİTMASI VE GELİŞTİRİLEN UZMAN SİSTEM	45
8.1. Ağırlıklı Bulanık Çıkarım Algoritması	46
8.1.1. Ağırlıklı Bulanık Çıkarım Algoritması'nda Kullanılan Yamık Bulanık Sayılar	46
8.1.2. Ağırlıklı Bulanık Çıkarım Algoritması'nda Kullanılan Kural Yapısı	47
8.1.3. Ağırlıklı Bulanık Çıkarım Algoritması'nda Kullanılan Kural Matrisi ve Doğruluk Değer Matrisi	49
8.1.4. Ağırlıklı Bulanık Çıkarım Algoritması'nda Kullanılan Çıkarım Yöntemi	50
8.2. Geliştirilen Uzman Sistem	53
8.2.1. Yamuk Bulanık Sayıların Uzman Sistemde Temsil Edilmesi	55
8.2.2. Önergelerin Doğruluk Değerlerinin Uzman Sistemde Temsil Edilmesi	55
8.2.3. Kullanılan Kural Tabanı ve Kural Tabanının Biçimi	57
8.2.4. Önergelerin Ağırlıklarının Uzman Sistemde Temsil Edilmesi	58
8.2.5. Önergelerin Doğruluk Değerlerinin ve Ağırlıklarının Sayısallaştırılması	59
8.2.6. Bileşik Önergelerin Doğruluk Değerlerinin Hesaplanması	61
8.2.7. Kural Matrisinin Oluşturulması	65
8.2.8. Çıkarım İşleminin Gerçekleştirilmesi	66
8.2.9. Çıkarım İşleminin Sonuçlarının Üretilmesi	67
9. UYGULAMA	68
10. SONUÇLAR VE ÖNERİLER	73
KAYNAKLAR	75
EK A	76
EK B	79
ÖZGEÇMİŞ	82

KISALTMALAR

YZ	: Yapay Zeka
US	: Uzman Sistem
ÇM	: Çıkarım Mekanizması
BT	: Bilgi Tabanı
İYÇ	: İleri Yönde Çıkarım
GYÇ	: Geri Yönde Çıkarım



TABLO LİSTESİ

	<u>Sayfa No</u>
Tablo 5.1. X’de tanımlı bazı bulanık kümeler.....	29
Tablo 6.1. R ilişkisi matrisi.....	36
Tablo 8.1. Yamuk bulanık sayıların bellekteki yerleşimi.....	55
Tablo 8.2. Bulanık doğruluk değerlerinin bellekteki yerleşimi.....	56
Tablo 8.3. Bulanık doğruluk değerlerinin ve ağırlıklarının bellekteki Yerleşimi.....	59
Tablo 8.4. Bellekteki sayısallaştırılmış değer matrisi.....	61
Tablo 8.5. dcmp_list yapısındaki yerleşim ve işlem akışı.....	65
Tablo 8.6. Bellekteki F kural matrisi.....	66



ŞEKİL LİSTESİ

	<u>Sayfa No</u>
Şekil 1.1 : Akıl ve bilgi arasındaki ilişki.....	2
Şekil 1.2 : Sığ ve derin bilgi.....	2
Şekil 2.1 : Eldeki gerçeklerden yeni gerçeklerin üretilmesi.....	7
Şekil 2.2 : Bir çerçeve örneği.....	9
Şekil 3.1 : Takım elemanları ve etkileşim içinde oldukları US parçaları.....	14
Şekil 3.2 : Uzmanların ve US'lerin benzer çalışma biçimleri.....	14
Şekil 3.3 : Uzman sistemin yapısı.....	16
Şekil 4.1 : Kurallardan oluşmuş bir ağ yapısı.....	19
Şekil 4.2 : Türkiye'deki bir uçuş ağı.....	21
Şekil 4.3 : Ankara-Diyarbakır uçuş ağı.....	21
Şekil 4.4 : Sinop-Bolu uçuş ağı.....	22
Şekil 5.1 : A belirgin kümesi.....	26
Şekil 5.2 : $\underline{A}$ bulanık kümesi.....	26
Şekil 5.3 : $\overline{A}$ bulanık kümesinin üyelik fonksiyonu.....	28
Şekil 5.4 : Bir bulanık kümenin çekirdek, destek ve sınır bölgesi.....	30
Şekil 5.5 : $\underline{A}$ bulanık kümesinin tümleyeni.....	31
Şekil 5.6 : $\underline{A} \cup \underline{B}$ bulanık kümesi.....	32
Şekil 5.7 : $\underline{A} \cap \underline{B}$ bulanık kümesi.....	32
Şekil 8.1 : Bir yamuk bulanık sayı.....	46
Şekil 8.2 : Çıkarım sürecinin akış şeması.....	54

GENEL AMAÇLI BİR BULANIK UZMAN SİSTEM

ÖZET

Uzman sistemler; bilgi tabanı, çıkarım mekanizması ve kullanıcı arayüzünden oluşan, günlük yaşamda kendi alanlarında uzmanlaşmış kişilerce yürütülen işlemleri gerçekleştiren bilgisayar programları olarak tanımlanabilir. Genellikle doğal konuşma diliyle ifade edilen bilgi belirsizlik içerir. Belirsizlik; karmaşıklık, ihmal, ölçümlerde yapılan hata veya duyarsızlık gibi pekçok nedenden kaynaklanabilir ve uzman kişilerin, karar verme sürecinde bu belirsizlik içeren bilgiyle çıkarım yapması gerekir. Dolayısıyla, uzman sistemlerin de belirsizlik içeren bilgiyi kullanabilmesi bir zorunluluktur.

Bu yüksek lisans tez çalışmasının amacı; belirsizlik içeren bilgiyi kullanarak çıkarım yapabilen genel amaçlı bir bulanık uzman sistem geliştirmektir. Uzman sistemi geliştirme sürecinde Shyi Ming Chen tarafından ortaya atılan Ağırlıklı Bulanık Çıkarım Algoritması kullanılmıştır.

Bu algoritmada, kuralların koşul kısmında yer alan önermelerin doğruluk değerleri ve ağırlıkları (çıkarım sürecinde sahip oldukları önem), kuralların belirginlik çarpanları (uzman kişinin kuraldan emin olma derecesi) belirsizlik içerebilir ve bu belirsizlik, dilsel terimler ve bu dilsel terimlere karşılık gelen yamuk bulanık sayılar (yamuk şeklindeki normal, konveks bulanık kümeler) aracılığıyla temsil edilir. Belirsizliğin temsil edilmesinde kullanılan dilsel terimler ve bu terimlere karşılık gelen yamuk bulanık sayılar aşağıda listelenmiştir:

unknown	(0,0,0,0)
absolutely-false	(0,0,0,0)
very-low	(0,0,0.02,0.07)
low	(0.04,0.1,0.18,0.23)
medium-low	(0.17,0.22,0.36,0.42)
medium	(0.32,0.41,0.58,0.65)
medium-high	(0.58,0.63,0.80,0.86)
high	(0.72,0.78,0.92,0.97)
very-high	(0.975,0.98,1,1)
absolutely-high	(1,1,1,1)

Kural tabanında yer alan bir kuralın genel yapısı (1.1) denkleminde görülmektedir:

$$R_i: \text{IF } C_j \text{ THEN } C_k \text{ (CF=f}_{kj}\text{)} \quad (1.1)$$

(1.1) denkleminde, C_j doğruluk değeri bilinen önermeyi, C_k doğruluk değeri hesaplanacak önermeyi ve f_{kj} kuralın belirginlik çarpanını göstermektedir. Eğer t_j ile C_j önermesinin doğruluk değerini gösterirsek C_k önermesinin doğruluk değeri t_k (1.2) denklemiyle hesaplanır:

$$t_k = t_j \times f_{kj} \quad (1.2)$$

Kuralların koşul kısmında AND ve/veya OR bağlaçlarıyla birleştirilmiş önermeler yeralabilir. Bu bileşik önermelerin doğruluk değerleri (1.3), (1.4) denklemleriyle hesaplanır.

a) IF C_{j1} AND C_{j2} AND... AND C_{jn} THEN C_k ($CF=f_{kj}$):

$C_{j1}, C_{j2}, \dots, C_{jn}$ önermelerinin doğruluk değerleri ve ağırlıkları sırasıyla $t_{j1}, t_{j2}, \dots, t_{jn}$ ve $w_{j1}, w_{j2}, \dots, w_{jn}$ olsun. $C_j = C_{j1}$ AND C_{j2} AND... AND C_{jn} şeklinde tanımlanan C_j bileşik önermesinin doğruluk değeri t_j :

$$t_j = (t_{j1} \times w_{j1} + t_{j2} \times w_{j2} + \dots + t_{jn} \times w_{jn}) / (w_{j1} + w_{j2} + \dots + w_{jn}) \quad (1.3)$$

b) IF C_{j1} OR C_{j2} OR... OR C_{jn} THEN C_k ($CF=f_{kj}$):

$C_{j1}, C_{j2}, \dots, C_{jn}$ önermelerinin doğruluk değerleri ve ağırlıkları sırasıyla $t_{j1}, t_{j2}, \dots, t_{jn}$ ve $w_{j1}, w_{j2}, \dots, w_{jn}$ olsun. $C_j = C_{j1}$ OR C_{j2} OR... OR C_{jn} şeklinde tanımlanan C_j bileşik önermesinin doğruluk değeri t_j :

$$\begin{aligned} t_j = & [(t_{j1} \times w_{j1}) / (w_{j1} + w_{j2} + \dots + w_{jn})] \vee \\ & [(t_{j2} \times w_{j2}) / (w_{j1} + w_{j2} + \dots + w_{jn})] \vee \\ & \dots \\ & [(t_{jn} \times w_{jn}) / (w_{j1} + w_{j2} + \dots + w_{jn})] \end{aligned} \quad (1.4)$$

Öte yandan kural tabanında pekçok kural yer aldığından (1.2) ile tanımlanan işlemin her bir kural için tekrarlanması gereklidir. Bu işlem için kural matrisi (F) ve doğruluk değer matrisi (T) olarak adlandırılan iki matrise gereksinim vardır.

T; önermelerin doğruluk değer matrisidir. Bilgi tabanında m adet basit önermenin ve k adet bileşik önermenin olduğunu varsayarsak $j = 0, 1, \dots, m+k$ için $T(j)$, (1.5a) denklemiyle tanımlanır. T'in matris gösterimi (1.5b) de verilmiştir.

$$T(j) = t_j \quad (1.5a)$$

$$T = \begin{bmatrix} t_1 \\ t_2 \\ \cdot \\ \cdot \\ \cdot \\ t_{m+k} \end{bmatrix} \quad (1.5b)$$

Çıkarım sürecinin başlangıcında C_j önermesinin doğruluk değeri bilinmiyor ise $T(j) = \text{"unknown"} = (0,0,0,0)$ değerini alır. C_j önermesi bir bileşik önerme ise $T(j)$ (1.3) veya (1.4) denklemi ile hesaplanır.

F , kuralların belirginlik çarpan matrisidir. Tüm kuralların koşul veya sonuç kısımlarında toplam $m+k$ adet önermenin olduğunu varsayalım. Herhangi iki C_j ve C_k önermeleri arasında bir ilişki yoksa, $F(k,j) = \text{"unknown"}$, C_j ve C_k önermeleri arasında birebir aynı olmak durumu sözkonusu olduğundan, refleksifliği sağlamak için $k=0, 1, \dots, m$, $F(k,k) = \text{"absolutely-high"}$ olarak atanır. F matrisinin boyutu $m+k \times m+k$ olur ve F 'in elemanları ($F(k,j)$) $k,j = 0, 1, \dots, m+k$ için (1.1) kullanılarak (1.6a) denklemiyle tanımlanır. F 'in matris gösterimi (1.6b) de verilmiştir.

$$F(k,j) = f_{kj} \quad (1.6a)$$

$$F = \begin{bmatrix} f_{11} & f_{12} & \dots & f_{1(m+k)} \\ f_{21} & f_{22} & \dots & f_{2(m+k)} \\ \vdots & \vdots & \dots & \vdots \\ f_{(m+k)1} & f_{(m+k)2} & \dots & f_{(m+k)(m+k)} \end{bmatrix} \quad (1.6b)$$

Ara doğruluk değer matrisi T^* ; önermelerin çıkarım sürecinde hesaplanan ara doğruluk değerlerini barındırır ve (1.7) denklemiyle tanımlanan $T^* = F \otimes T$ işlemiyle oluşturulur.

$$T^* = F \otimes T = \begin{bmatrix} (f_{11} \times t_1) \vee (f_{12} \times t_2) \vee \dots \vee (f_{1(m+k)} \times t_{m+k}) \\ (f_{21} \times t_1) \vee (f_{22} \times t_2) \vee \dots \vee (f_{2(m+k)} \times t_{m+k}) \\ \vdots \\ \vdots \\ \vdots \\ (f_{(m+k)1} \times t_1) \vee (f_{(m+k)2} \times t_2) \vee \dots \vee (f_{(m+k)(m+k)} \times t_{m+k}) \end{bmatrix} \quad (1.7)$$

Eğer $T^* = T$ eşitliği sağlanırsa çıkarım işlemi tamalanmış demektir ve T matrisi önermelerin sonuç doğruluk değerlerini içerir. $T^* = T$ eşitliği sağlanmadığı sürece, $T = T^*$ atamasının ardından (1.7) ile tanımlanan işlem $T^* = T$ eşitliği sağlanıncaya kadar çevrimler halinde yinelenir.

Birinci bölümde, yüksek lisans tez çalışmamızın ana başlıklarından olan bilgi ve belirsizlik, ikinci bölümde, bilginin, çıkarım sürecinde kullanılabilmesi için temsil edilmesinde yararlanılan gerçek, kural, hiyerarşi, çerçeve yapıları ve kalıtım, belirginlik çarpanı kavramları incelenmiştir.

Üçüncü bölümün konusu uzman sistemlerin yapısı ve çalışma ilkeleridir.

Dördüncü bölümde, çıkarım süreci için şekillendirilen bilginin uzman sistem tarafından kullanılmasında kullanılan yöntemler (ileri yönlü çıkarım, geri yönlü çıkarım) ele alınmıştır.

Bilgideki belirsizliğin temsil edilmesinde kullanılan belirgin küme, bulanık küme, üyelik fonksiyonu kavramları beşinci, bilgi parçaları arasında ilişkiler kurmakta kullanılan belirgin bağıntılar ve bulanık bağıntılar altıncı bölümde tanıtılmıştır.

Yedinci bölüm, ana konusu, belirsizlik içeren bilgi ile çıkarım yapmak olan bulanık mantık kavramını tanıtmaktadır.

Ağırlıklı Bulanık Çıkarım Algoritması'nın tanıtımı ve bu algoritmada anlatılanların, uzman sistemin geliştirme sürecinde nasıl yorumlanarak hayata geçirildiği sekizinci bölümde detaylı olarak açıklanmıştır.

Yüksek lisans tez çalışmasının sonundaki eklerde, uzman sistemin, bazı hastalıklarının teşhisini nasıl yaptığının örnekleri yer almaktadır. Bu örneklerde kullanılan kural tabanı, İstanbul Üniversitesi Dış Hekimliği Fakültesi 5.sınıf öğrencisi Sn.Onur Çağlar'ın yardımlarıyla hazırlanmıştır.



A GENERAL PURPOSE FUZZY EXPERT SYSTEM

SUMMARY

Expert systems are computer programs that consists of a knowledgebase, an inference engine and a user interface and performs operations that are carried out by human experts in our daily life. Knowledge, usually expressed in natural language, contains uncertainty. This uncertainty may result from many reasons (ambiguity, complexity, randomness, ignorance etc) and human experts must deal with this uncertainty in decision making process. So, it is essential that, experts system that has the same task, must be capable of managing uncertainty.

Main purpose of this thesis is to develop a general purpose expert system that can use uncertain knowledge. In development process, a weighted fuzzy reasoning algorithm for rulebased systems based on fuzzy logics (A Fuzzy Reasoning Approach for Rule-Based Systems Based On Fuzzy Logics proposed by Shyi Ming Chen) was used.

In this algorithm, truth values of conditions appearing in the antecedent portions of rules, certainty factors of rules and weights of conditions appearing in the antecedent portions of rules are represented by trapezoidal fuzzy numbers. A fuzzy number is a fuzzy set that is normal and convex. Trapezoidal fuzzy numbers are fuzzy numbers and are in the form of a trapezoid. In the algorithm, each trapezoidal fuzzy number is represented by a 4-tuple. Linguistic variables that are used to express uncertainty in truth values, weights and certainty factors and their corresponding 4-tuples are as follows:

unknown	(0,0,0,0)
absolutely-false	(0,0,0,0)
very-low	(0,0,0.02,0.07)
low	(0.04,0.1,0.18,0.23)
medium-low	(0.17,0.22,0.36,0.42)
medium	(0.32,0.41,0.58,0.65)
medium-high	(0.58,0.63,0.80,0.86)
high	(0.72,0.78,0.92,0.97)
very-high	(0.975,0.98,1,1)
absolutely-high	(1,1,1,1)

General structure of a rule in the rulebase is as follows:

$$R_i: \text{IF } C_j \text{ THEN } C_k \text{ (CF=f}_{kj}\text{)} \quad (1.1)$$

In (1.1), C_j is the condition whose truth value is in hand, C_k is the statement whose truth value to be calculated and f_{kj} is the certainty factor of the rule. If t_j is the truth value of C_j then truth value C_k : t_k is calculated by (1.2);

$$t_k = t_j \times f_{kj} \quad (1.2)$$

Rulebase of the expert system may contain rules that have complex conditions in their antecedent portions. To calculate truth values of these complex conditions (1.3), (1.4) are used.

a) IF C_{j1} AND C_{j2} AND... AND C_{jn} THEN C_k ($CF = f_{kj}$) :

Truth values of conditions $C_{j1}, C_{j2}, \dots, C_{jn}$ are $t_{j1}, t_{j2}, \dots, t_{jn}$ and weights $C_{j1}, C_{j2}, \dots, C_{jn}$ are $w_{j1}, w_{j2}, \dots, w_{jn}$ respectively. Truth value of C_j where $C_j = C_{j1}$ AND C_{j2} AND... AND C_{jn} is

$$t_j = (t_{j1} \times w_{j1} + t_{j2} \times w_{j2} + \dots + t_{jn} \times w_{jn}) / (w_{j1} + w_{j2} + \dots + w_{jn}) \quad (1.3)$$

b) IF C_{j1} OR C_{j2} OR...OR C_{jn} THEN C_k ($CF = f_{kj}$) :

Truth values of conditions $C_{j1}, C_{j2}, \dots, C_{jn}$ are $t_{j1}, t_{j2}, \dots, t_{jn}$ and weights $C_{j1}, C_{j2}, \dots, C_{jn}$ are $w_{j1}, w_{j2}, \dots, w_{jn}$ respectively. Truth value of C_j where $C_j = C_{j1}$ OR C_{j2} OR... OR C_{jn} is

$$\begin{aligned} t_j = & [(t_{j1} \times w_{j1}) / (w_{j1} + w_{j2} + \dots + w_{jn})] \vee \\ & [(t_{j2} \times w_{j2}) / (w_{j1} + w_{j2} + \dots + w_{jn})] \vee \\ & \dots \\ & [(t_{jn} \times w_{jn}) / (w_{j1} + w_{j2} + \dots + w_{jn})] \end{aligned} \quad (1.4)$$

On the other hand, there are more than one rule in the rulebase. So (1.2) must be repeated for each of these rules. For this operation two matrices, called rule matrix (F) and truth value matrix (T), are used.

T is the matrix that holds the truth values of statements. If there are m simple statements and k complex statements in the rulebase then; for $j = 0, 1, \dots, m+k$ $T(j)$ is defined by (1.5a) and represented by (1.5b).

$$T(j) = t_j \quad (1.5a)$$

$$T = \begin{bmatrix} t_1 \\ t_2 \\ \cdot \\ \cdot \\ \cdot \\ t_{m+k} \end{bmatrix} \quad (1.5b)$$

If truth value of statement C_j is not known in the beginning of inference then $T(j)$ ="unknown". If C_j is a complex statement then $T(j)$ is calculated by (1.3) or (1.4).

F is the matrix that holds certainty factors of the rules. If there are $m + k$ statemens in the rulebase then; $k=0, 1, \dots, m+k$ $F(k,k)$ ="absolutely-true"= $(1,1,1,1)$ for reflexivity. If there is no relation between statements C_j and C_k via a rule then $k,j=0, 1, \dots, m+k$ $F(k,k)$ ="unknown"= $(0,0,0,0)$. If there is a relation between statements C_j and C_k via a rule then $k=0, 1, \dots, m+k$ $F(k,j)=f_{kj}$ by (1.6a). Formally:

$$F(k,j)=f_{kj} \quad (1.6a)$$

$$F = \begin{bmatrix} f_{11} & f_{12} & \dots & f_{1(m+k)} \\ f_{21} & f_{22} & \dots & f_{2(m+k)} \\ \cdot & \cdot & \dots & \cdot \\ \cdot & \cdot & \dots & \cdot \\ \cdot & \cdot & \dots & \cdot \\ f_{(m+k)1} & f_{(m+k)2} & \dots & f_{(m+k)(m+k)} \end{bmatrix} \quad (1.6b)$$

Temporary truth values of statemets (T^*) is calculated by $T^* = F \otimes T$ given in (1.7)

$$T^* = F \otimes T = \begin{bmatrix} (f_{11} \times t_1) \vee (f_{12} \times t_2) \vee \dots \vee (f_{1(m+k)} \times t_{m+k}) \\ (f_{21} \times t_1) \vee (f_{22} \times t_2) \vee \dots \vee (f_{2(m+k)} \times t_{m+k}) \\ \dots \\ \dots \\ \dots \\ (f_{(m+k)1} \times t_1) \vee (f_{(m+k)2} \times t_2) \vee \dots \vee (f_{(m+k)(m+k)} \times t_{m+k}) \end{bmatrix} \quad (1.7)$$

If $T^* = T$ then it's concluded that inference is over and truth values of statements are in T . If it's not, these temporary truth values are taken in hand by $T = T^*$ and truth vales of complex statements are calculated again. (1.7) is repeated in cycles until $T^* = T$.

In the first part, concepts that are among main ideas of this thesis, information and uncertainty are defined.

In the second part, methods of knowledge representation are explained.

In the third part, experts systems are introduced.

In the forth part, inference methods are presented.

In the fifth part, fuzzy sets are defined.

In the sixth part, crisp and fuzzy relations are explanied.

In the seventh part, fuzzy logic concepts are presented.

In the eighth part, Weighted Fuzzy Reasoning Algorithm and expert system development process are explained.

In the ninth part, usage of expert system is shown.

In the tenth part, results and comments are given.

Appendix A and B contains two examples of expert system's inference process.



1. BİLGİ VE BELİRSİZLİK

Uzman sistemler (US), belirli bir konuda uzmanlaşmış kişiler tarafından yürütülen işlemleri, bünyelerinde barındırdıkları bilgiyi kullanarak gerçekleştiren, kullanıcı arabirimi, bilgi tabanı ve çıkarım mekanizmasından oluşan bilgisayar programlarına denir. US'ler, güçlerini bilgiden alır. US'lerin en önemli bileşeni sahip oldukları bilgidir ve çıkarım işlemini başarılı kılan; bu bilginin etkin kullanımıdır. Bu aşamada bilginin, US'lerde kullanımı açısından tanımını yapalım.

Bilgi; US'lerin çıkarım sürecinde kullandığı, günlük yaşamda karşılaşılan gerçeklerin ve hangi durumlarda, bu gerçeklerden hareketle hangi yeni gerçeklere ulaşılabileceğini ortaya koyan kuralların bileşimidir [1].

US'lerce kullanılan bilginin temsil edilmesi ve çıkarım sürecinde kullanılabilmesi için biçimlendirilmesi gereklidir. Nasıl ki; verinin saklanması için veri yapıları (data structures) kullanılıyorsa, bilginin şekillendirilmesi için de, bilgi yapıları (knowledge structures) kullanılır. Bu yapılar üzerinde daha ilerideki bölümlerde detaylı olarak durulacaktır.

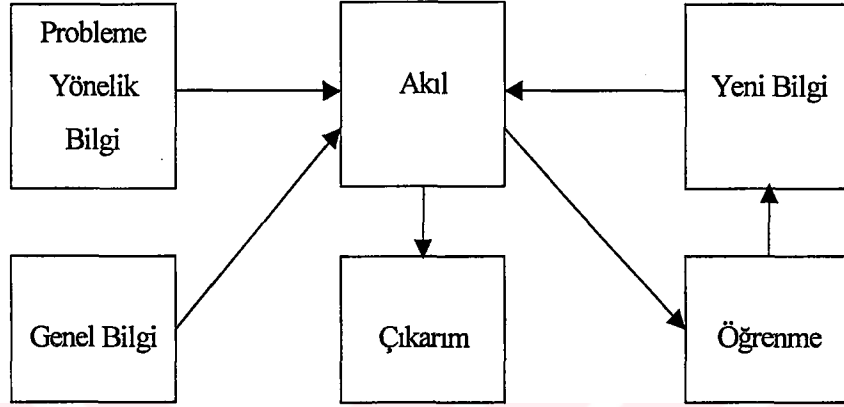
Bilgi, US geliştirme sürecinde, akıl gibi doğal yollardan edinilen bir özellik olmaktan ziyade insanlar ve sistemler arasında değiş tokuşu yapılan bir öge olarak nitelendirilebilir. Akıl dolu davranışlar bilgiye dayansa da akıl ve bilgiyi eşdeğer kabul etmek yanlışlara neden olabilir. Akıl ve bilgi arasındaki etkileşim Şekil 1.1'de gösterilmiştir [1].

US'in çıkarım yapmak için ihtiyaç duyduğu bilgi, US'den beklenen davranışların bir yansımasıdır. Bazı US'ler son derece hızlı değişen, dinamik bilgiyle çıkarım yaparken, bazı US'lerin sahip oldukları bilgi statiktir. Hızla değişen bir ortama ayak uydurabilmek için derin bir bilgi birikimi gereklidir.

1.1. Bilginin Düzeyleri (levels of knowledge)

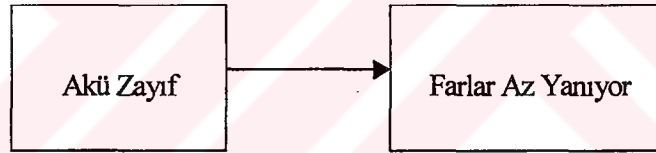
Bilgi, hangi temel ilkelerin ve ilişkilerin çıkarım sürecine dahil edildiğine bağlı olarak düzeylendirilebilir.

Sığ bilgi (shallow knowledge), sadece karşılaşılan problemle ilgili olarak edinilmiş bilgidir. Örneğin; bir cismin hızı, bir miktar suyun sıcaklığı vb. Derin bilgi (deep knowledge) ise, daha derinlerde, bilgi parçacıklarının etkileşimi ile ortaya çıkan bilgidir. Derin bilgi, karşılaşılan farklı problemlerin çözümünde kullanılabilir. Bir doktorun sahip olduğu mesleki bilgi buna örnek olarak verilebilir. Sığ ve derin bilginin şematik gösterimi Şekil 1.2’de gösterilmiştir [1]:

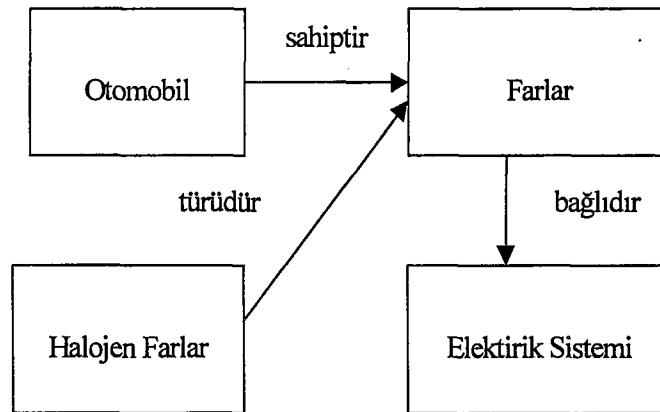


Şekil 1.1 Akıl ve bilgi arasındaki ilişki

Sığ Bilgi



Derin Bilgi



Şekil 1.2 Sığ ve derin bilgi

1.2. Bilginin Bileşenleri (components of knowledge)

Bilgi temsilinde kullanılan herhangi bir yöntemin; bilgiyi çeşitli düzeylerde temsil edebilmesi, tanımlayabilmesi, organize edebilmesi, bilgi parçacıkları arasında ilişki kurabilmesi gereklidir. Bunun için nesneler (objects) kullanılır. Hastalıklar, araçlar vb. birer nesne olarak tanımlanabilir. Probleme özgü nesne yelpazesinin belirlenmesinin, bilginin temsil edilmesinde çok önemli bir işlevi vardır. Bir kentin taşımacılık modelinin kurulmasında otobüs, otomobil, kamyon, kavşak vb. nesneler kullanılacak iken bu kentteki suç oranlarının düşürülmesine yönelik bir modeldeki nesneler farklı olacaktır. Bu nesnelerin belirlenmesi sonrasında yapılması gereken nesnelerin tanımlanması, organize edilmesi ve aralarındaki ilişkilerin kurulmasıdır. Bu işlemler, bilginin temel bileşenleri üzerine oturur [1]. Bu bileşenler:

- a) Adlandırma (naming) : Bilgi parçalarının isimlendirilmesidir. Bu işlem sırasında, doğal olarak isimler (names) kullanılır. Örneğin; bir kimyasal maddeye “NaCl”, bir elektronik araca “potansiyometre” adını vermek.
- b) Nitelme (describing) : Nesnelerin, genellikle sıfatlar kullanılarak niteliklerinin belirtilmesidir. Örneğin; hastanın yaşı 38.
- c) İlişkilendirme (relating) : Nesnelerin ilişkilendirilmesi ile eldeki bilgiye ek, yeni bilgi üretilmesidir. Örneğin; boru ve vana bağlantılıdır.
- d) Organizasyon (organizing) : Nesnelerin, belirli nesne sınıfları altında gruplandırılmasıdır. Örneğin; x marka otomobil motoru benzinli motordur.
- e) Kısıtlamalar (constraints) : Nesnelerin özelliklerine veya aralarındaki ilişkilere getirilen kısıtlardır. Örneğin; 18 yaşından küçük çocuklara alkollü içki satılamaz.

1.3. Belirsizlik (uncertainty)

Yaşam karmaşıktır. Bu karmaşıklığın önde gelen nedenlerinden biri de bilgidaki belirsizliktir (uncertainty). Bilgi, her zaman 1.75 m., 75 kcal. gibi kesin veri şeklinde ifade edilemez. “Çok yüksek”, ”oldukça fazla” gibi belirsizlik içeren önermeler, bilginin ifade edilebilmesi ve paylaşılabilmesi için vazgeçilmezdir. Belirsizlik pek çok nedenden kaynaklanabilir [1]:

- a) Karmaşıklık (complexity)
- b) İhmal (ignorance)
- c) Rastgele seçim (randomness)
- d) Ölçümlerdeki duyarsızlık ve/veya hata (inadequate measurement)
- e) Bilgideki yetersizlik (lackness of knowledge)
- f) Doğal dildeki bulanıklık (fuzziness in natural language)
- g) Çok yönlülük (ambiguity) vb.

Ancak, insanlar belirsizlik altında da karar vermek zorundadır. Yatırım kararları almak, hastalık teşhisleri yapmak gibi pekçok sürecin net olmayan bilgiyle tamamlanması gerekir. Bu nedenle, insanoğlunun düşünmeye başlamasından bu yana temel uğraşlarından biri; karmaşıklığa dayalı problemler olmuş, karmaşıklık ve belirsizlik, insanoğlunun karşılaştığı ekonomik, sosyal, teknolojik pek çok problemin temel nedenlerini oluşturmuştur.

Peki o halde, insanoğlu tarafından tasarlanan, karmaşıklık ve çok yönlülük içeren problemleri çözme yetisi olmayan bilgisayarlara neden gereksinim duyulmaktadır? Gerçek bir sistemin tam ve doğru tanımı, bir insanın tüm dağarcığındaki bilgiden daha fazlasını gerektirirken, bu sistemler hakkında nasıl fikir yürütülebilir ?

Bu sorunun yanıtı şu biçimde verilebilir. İnsanlar, bilgisayarların şu aşamada sahip olmadığı yaklaşımla çıkarım yapma (approximate reasoning) yetisine sahiptir. İnsanlar, karmaşık bir sistem hakkında çıkarım yaparken sistemin davranışı hakkındaki genel bilgilerini kullanır [2].

Durum böyle olunca, insanoğlunun karar verme yetisini taklit eden US'lerin de belirsizlik altında çıkarım yapabilmesi bir zorunluluk haline gelmiştir.

Yaşamdaki farklı etkinlik alanları farklı düzeyde belirsizlik içerir. Bir bilgisayarı ele alacak olursak; bilgisayarın işlem hızı, bilgi depolama kapasitesi gibi özellikleri bazı sayısal veriyle (255 mhz, 5 gbyte vb.) ifade edilebilir. Ancak borsa analizi yapan bir yatırım uzmanı için durum çok daha farklıdır. Bu uzman, "talep çok düşük", "enflasyon oldukça yüksek" gibi belirsizlik içeren bilgiyi kullanarak karar vermek zorundadır. Az karmaşıklığı, dolayısıyla az belirsizliği olan sistemlerin

tanımında kapalı matematiksel ifadeler yeterlidir. Daha karmaşık ancak hakkında yeterli verinin edinebildiği sistemlerin belirsizliği, yapay sinir ağları (artificial neural network) gibi modeller ile azaltılabilir. Sayısal verinin çok az, belirsizliğin fazla olduğu sistemlerde ise, bulanık çıkarım yöntemleri (fuzzy reasoning methods) kullanılarak sistemin davranışı anlaşılabilir. Bulanık çıkarım yöntemleri, yeterli verinin edinebildiği problemlerden daha çok, belirsizliğin söz konusu olduğu durumlarda kullanılır [2].



2. BİLGİNİN TEMSİL YÖNTEMLERİ

US, sonuçta bir bilgisayar programı olduğundan, sahip olduğu bilgisinin, çıkarım sürecinde kullanılabilecek biçimde şekillendirilmesi gereklidir. Bu gereklilik dizi, yığın, bağlı liste gibi yapılara benzer yapıların kullanımını gündeme getirir.

Bilginin işlenebilir hale getirilmesi için, bilgi yapıları (knowledge structures) kullanılır. Bunlar;

- a) gerçekler (facts)
- b) kurallar (rules)
- c) hiyerarşi ve çerçeveler (hierarchy and frames)

olarak sıralanabilir. Şimdi bu yapıları inceleyelim:

2.1. Gerçekler (facts)

Gerçekler, çevremizdeki olaylar, olgular vb. hakkında bize doğrudan bilgi veren, bilginin temel parçacıkları olarak nitelenebilir. "Meydana gelen bir depremin aletsel büyüklüğü 4.5", "hareketli bir cismin hızının çok yüksek oluşu" vb. Bir insanın yaşı gibi gerçekler zaman içinde değişebilir.

US'lerde gerçekler, bilgi tabanı içinde yer alan ve gerçek tabanı (fact base) adı verilen bir ortamda saklanır.

Uzman kişilerin düşünüş biçimleri tam olarak bilinmese de, sahip oldukları genel bilgiyi problem ile ilgili gerçekleri dikkate alarak kullandıkları söylenebilir. İşte US'in bilgisi, uzmanın bu genel bilgisine eşdeğer durumdadır.

Uzmanlar, çalışmalarını net olmayan, belirsiz bilgiye dayandırabilir. Buna benzer olarak, US'in çıkarımda kullandığı gerçeklerin doğruluk değerleri de belirsiz (uncertain) olabilir. Örneğin; elektrikli sistemindeki bir arızanın, elektrik anahtarından kaynaklandığı konusunda "oldukça emin" olabiliriz. Bu belirsizlik; bilginin doğası, bilgi eksikliği, ölçüm hataları vb. nedenlerden kaynaklanabilir [1].

2.2. Kurallar (rules)

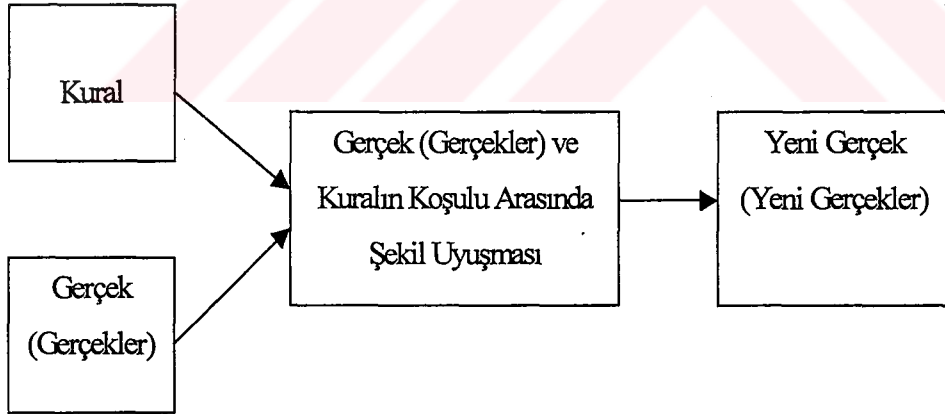
Gerçekler, çıkarım işleminde tek başlarına kullanılamaz. Gerçekler arasında ilişkiler kurmak ve yeni gerçeklere varmak için, kural (rule) dediğimiz bilgi yapısına ihtiyaç vardır. Genel kural yapısı,

IF koşul1 [VE / VEYA koşul2 ...] THEN sonuç1 [VE / VEYA sonuç2 ...] (2.1)

şeklindedir.

Bir kural, yapısındaki VE ve/veya VEYA operatörlerine bağlı olarak, koşul kısmındaki şartlardan tümünün veya bazılarının, bilgi tabanındaki gerçeklerle uyuşması halinde ateşlenir. Bu işlem, kuralın sonuç kısmındaki önermelerin doğruluk değerlerinin belirlenmesi demektir.

Kurallar, gerçekler halindeki bilgi parçacıklarından yeni gerçekler elde etmemizi sağlayan bilgi yapıları şeklinde nitelenebilir. Yeni bir gerçeğe ulaşmak için en az bir kural ve bir gerçek gereklidir. Kurallar, işlem sürecinin başlangıcında varolmayan bilginin yeni gerçekler halinde elde edilmesini sağlar (bakınız Şekil 2.1 [1]). Kuralların, çıkarım işleminde nasıl kullanıldığı Yaklaşımla Çıkarım başlığı altında detaylı olarak incelenmiştir.



Şekil 2.1 Eldeki gerçeklerden yeni gerçeklerin üretilmesi

2.3. Hiyerarşi, Çerçeveler ve Kalıtım (hierarchy, frames and inheritance)

2.3.1. Hiyerarşi (hierarchy)

Hiyerarşiler, bilginin nesneler arasındaki ilişkiler halinde temsil edilmesi için kullanılır. Hiyerarşi içinde yeralan her nesne, hiyerarşi içerisinde daha üst düzeylerde

yeralan ve ana (parent) olarak nitelenen başka nesneler ile ilişkilendirilir. Daha alt düzeylerdeki nesneler ana nesnelerin çocuklarıdır (child).

Hiyerarşiler, nesneleri organize etmek ve bu yolla bilginin düzenliliğini sağlamak için kullanılır.

Hiyerarşilerde en fazla dikkate alınması gereken nokta ana nesne için geçerli olan her bilgi parçasının çocukları olan nesneler için de geçerli olmasıdır. Bu kabul, birbirleri ile ilişkili olan nesnelerin nitelenmesinde, ortak olan özelliklerin her nesne için yinelenmesi gereğini ortadan kaldırır [1].

2.3.2. Çerçeveler (frames)

Gerçeklerin ve kuralların çıkarımda nasıl birarada kullanıldıklarını gördük. Ancak bilgiye kolay ulaşabilmek için başka bir yapıya ihtiyaç vardır.

“Bilgi paketi” olarak nitelendirilebilecek olan bu yeni yapı; nesnelerin özelliklerinin birarada ve düzenli bir şekilde tutulmasını sağlar [1]. Şu örneği ele alalım:

Otomobil bir çeşit taşıma aracıdır

Otomobil yolcu taşır

Otomobilin bir sürücüsü vardır

Otomobil, hareket enerjisi üretmek için yakıt kullanır

Yukarıdaki temsil yöntemi, bir otomobilin özelliklerini yansıtmakla birlikte, bu özellikleri birbirinden ayırık ve dağınık biçimde ifade ettiğinden bilgiye kolay erişimi sağlayamamaktadır. Oysa; etkin bir biçimde işlem yapmak zorunlu olduğundan, bilgiye kolay ve kısa yoldan erişebilmek çok önemlidir.

İşte; bilgiyi düzenli bir biçimde saklayarak, bilgiye kolay erişimi sağlayan ve hiyerarşiler ile birlikte nesneler arasındaki ilişkileri ortaya koyan bilgi yapısına çerçeve (frame) denir. Genel çerçeve yapısı:

Çerçeve-adı

Ana-çerçeve-adı

çizik1 değer1

çizik2 değer2

...

...

çizikN değerN

şeklindedir.

Çerçeveler, hiyerarşiler ile birlikte kullanılır. Hiyerarşi içinde, her çerçeveden daha yüksek düzeyde, ana çerçeve (parent frame) adı verilen bir başka çerçeve yeralır.

Herhangi bir nesnenin sahip olduğu özellikler, çerçeve içerisinde çizik (slot) denen alanlarda saklanır. Bu çizikler belli değerler (values) olarak o nesnenin özelliklerini yansıtır. Çerçevelere örnek, Şekil 2.2’de verilmiştir [1].

Ad	: Serçe
Ana	: Tofaş
Max. Hız	: 140 km/saat
Yakıt	: Benzin
Yolcu Sayısı	: 5

Şekil 2.2 Bir çerçeve örneği

2.3.3. Kalıtım (inheritance)

Canlıların ana ve babalarından, hatta daha önceki kuşaklardan bazı özellikler edinmeleri gibi, hiyerarşi içindeki yüksek düzeyli bir çerçevenin özellikleri, kendisi ile ilişkili olan ve daha alt düzeylerde yeralan çerçevelere aktarılır. Bu özellik aktarımına kalıtım (inheritance) denir.

Kalıtım ve hiyerarşi kavramları birarada kullanılarak, aynı sınıfa giren veya aynı nitelikleri taşıyan nesnelerin özellikleri, o sınıfın bir ana temsilcisinin özelliklerinin verilmesi ile belirlenebilir. Ancak bu, aynı sınıfa giren nesnelerin tüm özelliklerinin aynı olacağı anlamına gelmez.

Kurallar ve çerçeveler bilginin temsil edilmesinde kullanılan yapılar olarak karşılaştırılmamalıdır. Çerçeveler bilginin düzenlenmesi ve bilgiye kolay ulaşım için uygun yapılar iken, kurallar çıkarım sürecinde, gerçeklerden yeni gerçekler elde etmekte kullanılır [1].

2.4. Belirginlik Çarpanları (certainty factors)

Bilgi, her zaman net değildir ve belirsizlik içerebilir. Doğru sonuçlara varabilmek için, eldeki bilginin belirsizliğinin (bir başka açıdan bakıldığında netliğinin), çözüm sürecinde dahil edilmesi gereklidir. Bunun için kullanılan araçlardan biri; belirginlik çarpanlarıdır (certainty factors) [1].

Bir A gerçeğinin belirginlik çarpanı; $cf(A)$ ile gösterilir ve $cf(A)$, $[0,100]$ aralığında değerler alacak A gerçeğinin doğruluk derecesini belirler.

100 : gerçeğin tamamiyle doğru

0 : gerçeğin tamamiyle yanlış

$(0,100)$: gerçeğin doğruluk değerinin belirsiz olduğu anlamına gelir.

Ancak, karar verme sürecinde her zaman basit gerçekler kullanılmaz. Pekçok gerçek VE, VEYA gibi bağlayıcılar kullanılarak biraraya getirilebilir. Bu durumda, elde edilen bu yeni bileşik gerçeklerin belirginlik çarpanlarının da hesaplaması gereklidir. A ve B iki gerçek, $cf(A)$ ve $cf(B)$ sırasıyla bu gerçeklerin belirginlik çarpanları olmak üzere bu hesaplama aşağıdaki şekilde yapılır:

a) $A \text{ VE } B$: $cf(A \text{ VE } B) = \min[cf(A), cf(B)]$

b) $A \text{ VEYA } B$: $cf(A \text{ VEYA } B) = \max[cf(A), cf(B)]$

c) $DEĞİL A$: $cf(DEĞİL A) = 1 - cf(A)$

Belirginlik çarpanları, sadece gerçeklerden değil, kurallardan emin olma derecesini de gösterir. Bir kuralın belirginlik çarpanı, bu kural kullanılarak elde edilmiş bir yeni gerçeğin doğruluğundan ne kadar emin olabileceğimizi gösterir. Belirginlik çarpanı = a olan bir kuralın genel biçimi (2.2) de verilmiştir [1]:

$$\text{IF koşul THEN sonuç } cf = a \quad (2.2)$$

Sonucun belirginlik çarpanı: $cf(\text{sonuç})$, (2.3) ile elde edilir:

$$cf(\text{sonuç}) = cf(\text{koşul}) * a / 100 \quad (2.3)$$

100'e bölme işlemi, elde edilen değer $[0,100]$ sınırları arasında kalması için yapılmaktadır.

Bu yüksek lisans tez çalışmasında kullanılan ve Bölüm 8’de detaylı olarak tanıtılacak olan Ağırlıklı Bulanık Çıkarım Algoritması’nda da belirginlik çarpanları kullanılmış, belirginlik çarpanlarının bilgisayarda temsil edilmesinde bulanık kümelerden (belirginlik çarpanlarının, doğal dildeki sözcüklerle ifade edilmelerinden ve belirsizlik içermelerinden dolayı) yararlanılmıştır.



3. UZMAN SİSTEMLER

Genel problem çözme amaçlı ilk yapay zeka (YZ) programları ile yapılan uygulamalar, bu yazılımların ancak çok basit problemleri çözebildiklerini ve karmaşık problemlerin çözümünde çok fazla donanım birimi (bellek, disk vb.) ve çok fazla zaman kullandıklarını göstermiştir.

Bunun sonucunda, 1970'li yıllarda uzman sistem fikri ortaya atılmıştır.

Uzman sistem (US); belirli bir konuda uzmanlaşmış kişiler tarafından yürütülen işlemleri, bünyelerinde barındırdıkları bilgiyi kullanarak gerçekleştiren, kullanıcı arabirimi, bilgi tabanı ve çıkarım mekanizmasından oluşan bilgisayar programlarına denir. US'in işlem yapabilme gücü, problemin çözümü sırasında kullanılan algoritmaların etkinliğine ve kullanıldığı konu üzerinde sahip olduğu bilginin derinliğine bağlıdır. US'ler, çıkarım işlemi süresince, bilgi tabanında yeralan bilgiyi kullanır. Bu bilgi, US'in, kullanıcıya sorduğu soruların yanıtları ile birleştirilir ve böylelikle yeni bilgi elde edilir. US'ler, konu bağımlı olarak nitelendirilebilir. Yani; US'in bilgi tabanında, bir jeologtan alınmış bilgi varsa, sistemden hastalık teşhisi yapması beklenemez [1].

Nasıl belirli bir konuda uzmanlaşmış kişiler, o konu ile ilgili bilgilerini kullanarak problem çözerlerse, US'ler de, işlem sırasında bir uzmandan alınarak bilgisayar tarafından kullanılmaya uygun hale getirilmiş bilgi ile işlem yapar.

US'lerin kullanımı bir uzmanın;

- a) bulunmasının veya yetiştirilmesinin uzun zaman alacağı,
- b) fiziksel ve/veya zihinsel olarak yorulması,
- c) çözüm için çok gerekli bilgiyi yanlış hatırlaması veya unutması,
- d) günden güne verdiği kararlarda tutarsızlıkların olabilmesi,
- e) büyük bilgi yığınlarını çabuk anlayabilme yetisine sahip olmaması,
- f) hastalanması veya ölmesi gibi durumlarda büyük avantajlar sağlar [1].

Yakın geçmişe bir gözattığımızda başarılı US uygulamalarını görürüz:

Amerika Birleşik Devletleri, Stanford Üniversitesi'nde geliştirilen DENDRAL, jeologlardan derlenen bilgi ile donatılmıştı. Araştırmacıların DENDRAL'ı geliştirmekteki amaçları; yazılımı, bir uzay aracındaki bilgisayara yükleyerek, Mars yüzeyinin jeolojik analizini yapmak şeklindeydi. DENDRAL, kendisinden önce geliştirilen YZ yazılımlarının farklı konulardaki problemleri çözmeye yönelik olmasından ve sahip olduğu bilginin, belirli bir konuyla sınırlandırılmasından dolayı ilk US olarak nitelendirilebilir.

Bir diğer, adını duyurmuş US; MYCIN'dir. Bir fizikçiden alınan bilgiyi kullanan MYCIN, bakteriyel hastalıkları teşhis eden ve tedavi yöntemleri öneren bir US'dir.

Bir başka US olan INTERNIST ise 100.000 kuralı kullanarak hastalıklar ile belirtiler arasında ilişkiler kurabilir.

Belirsiz içeren bilgi ile çıkarım yapabilen US'lere örnek olarak tıpta kullanılan Cadiac-2 ve Rendir verilebilir.[1]

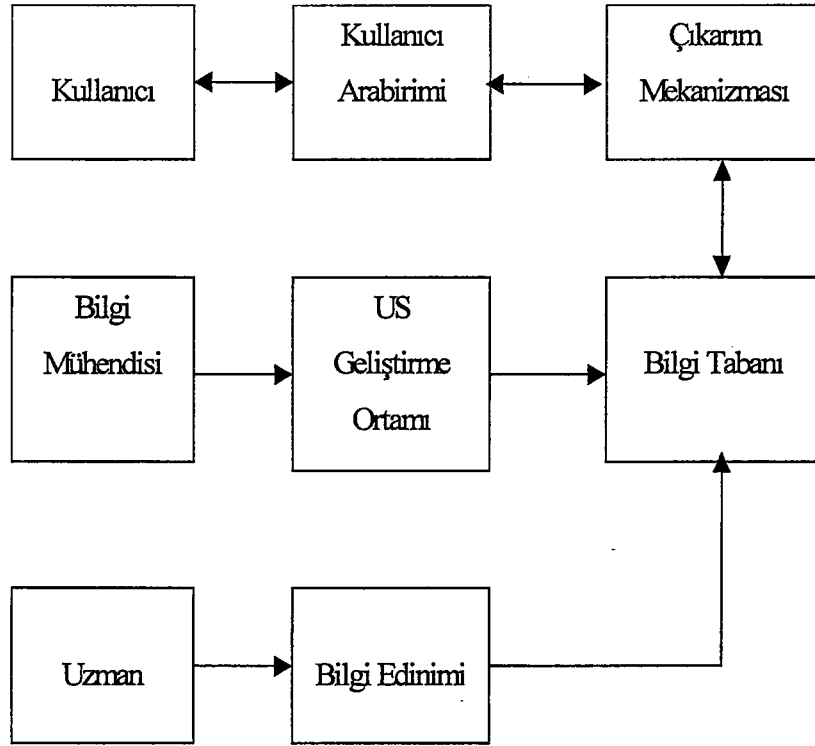
3.1. Uzman Sistemin Öğeleri

US'in çalışması, kendi alanlarında uzmanlaşmış kişilerden oluşan bir grubun yürüttüğü takım çalışmasına benzetilebilir. Bu takım;

- a) Uzman (domain expert) : US'in problem çözüm sürecinde kullandığı bilgiyi sağlayan kişi (matematikçi, teknisyen vb.)
- b) Bilgi (domain knowlegde) : US'in kullandığı bilgi
- c) Bilgi mühendisi (knowledge engineer) : Uzman ile etkileşime geçerek, derlediği bilgiyi, bilgisayarın kullanabileceği biçimde düzenleyen, kısacası bilgi tabanını oluşturan kişi
- d) Kullanıcı (user) : Geliştirilen US'i kullanan kişi

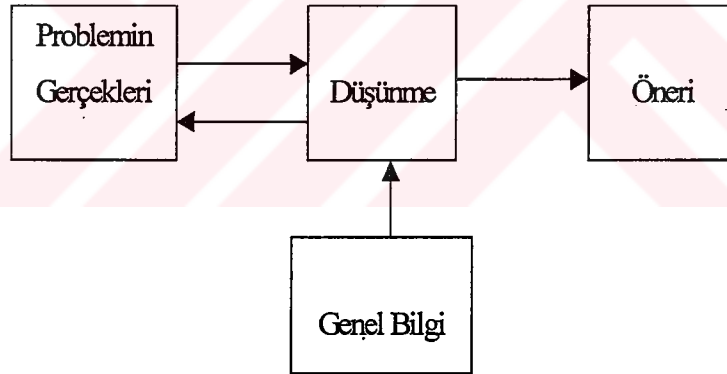
şeklinde oluşturulabilir.

Bu takımın elemanları ve etkileşim içinde oldukları US parçaları Şekil 3.1'de, günlük yaşantımızda karşımıza çıkan uzmanlar ile US'lerin birbirlerinininkine benzer çalışma biçimleri Şekil 3.2'de gösterilmiştir [1]:

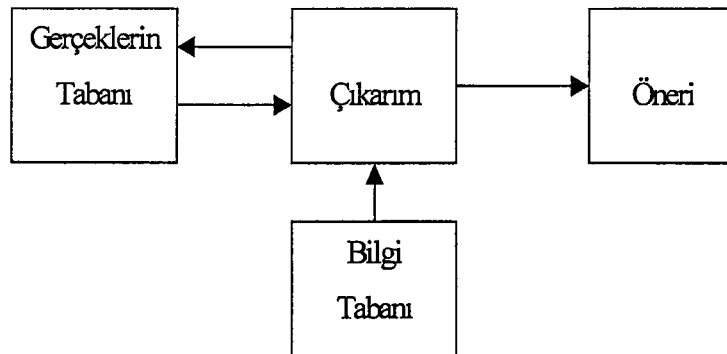


Şekil 3.1 Takım elemanları ve etkileşim içinde oldukları US parçaları

Uzman Kişi:



Uzman Sistem:



Şekil 3.2 Uzmanların ve US'lerin benzer çalışma biçimleri

3.2. Uzman Sistemlerin Yapısı ve Çalışma İlkeleri

Uzman kimdir? Uzman, "belirli bir uzmanlık alanında, sahip olduğu doğru, tam, güvenilir ve nitelikli bilgiyi kullanarak etkinlik gösteren kişi" şeklinde tanımlanabilir. Günlük hayatımıza karşılaştığımız bir doktor, mühendis veya tamirci uzmandır kişilerdir. Uzmanlar, günlük hayatımızda karşılaştığımız ve kendi uzmanlık alanlarına giren problemlerimizi, sahip oldukları bilgiyi kullanarak çözmeye çalışır.

US'leri, bir uzman kişiye ulaşmanın mümkün olmadığı, bir uzman kişinin ücretinin çok yüksek olduğu vb. durumlarda, uzman kişilerin etkinliklerini yürüten bilgisayar programları olarak nitelendirebiliriz. US'ler, bir veya birden fazla uzmandan derlenmiş bilgiyi kişisel bilgisayarlardan, büyük sistemlere kadar pekçok bilgi işlem sisteminin kullanabileceği biçimlerde barındırır. Şartlar değiştikçe bu bilginin güncellenmesi kolaydır. US'lerle ilgili olarak şu tanımları yapalım [1]:

Nesne : Olayları, insanları, yerleri vb. temsil eden yapılardır.

Şekil Uyuşması (pattern matching) : Bilgi parçacıklarını temsil eden iki nesnenin benzer olup olmadığının ortaya konması için karşılaştırılmasıdır. Bu teknik ses, görüntü, gerçek ve kavramların karşılaştırılmasında kullanılır.

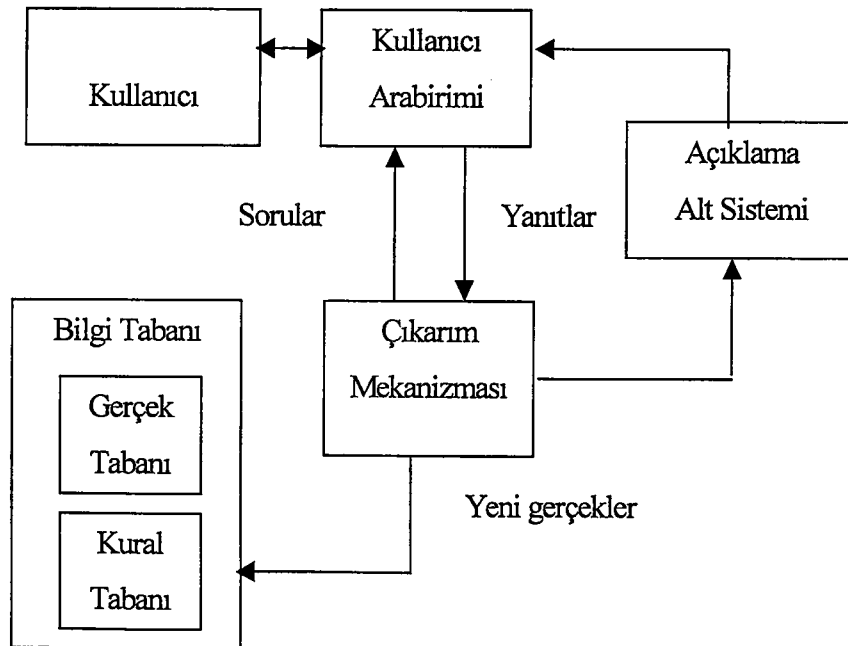
Çıkarım Mekanizması (inference engine) : Çıkarım mekanizması (ÇM), bir programlama dilinde yazılmış komutları ve şekil uyuşması tekniğini kullanarak, bilgi parçacıkları arasında ilişkiler kurmaya çalışır. Bu işlemin yürütülmesindeki amaç, sistemin o ana kadar doğruluklarını kanıtladığı gerçekleri (facts) ve kullanıcının verdiği ipuçlarını biraraya getirerek, kurallar aracılığıyla yeni gerçekler (new facts) elde etmek, ara ve ana sonuçlara ulaşmaktır. Kısacası ÇM, US'in sahip olduğu bilgi üzerinde yapılan işlemleri gözetten bir bilgisayar programıdır. US'in kullanıcısı tarafından, sorulan sorulara verilen yanıtları dikkate alarak bir yargıya varıncaya kadar bilgi tabanını tarar. İlk komuttan başlayıp son komuta doğru giden sıradan programlama mantığının aksine, bilgi parçacıkları arasındaki bir uyuma, herhangi bir kuralın ateşlenmesine, bu kural aracılığıyla elde edilen ara sonuçlar ise başka başka kuralların incelenmesine yol açabilir. Yapılan bu incelemenin belli bir sıra dahilinde yürütülmesi gibi bir zorunluluk yoktur. Bu işlem problemin kurgusuna bağlı olarak bilgi tabanının sonundan başlanarak geri yönde, bilgi tabanının başlangıcına doğru da yürütülebilir. Birinci tarama sonucunda elde edilen bir ara

sonuç bir başka taramanın, bu taramada elde edilen ikinci bir ara sonuç başka taramaların yapılmasına yolaçabilir. Unutulmamalıdır ki; ÇM, bilgi tabanında yeterli bilgi olmaması durumunda sonuç üretemeyebilir [1]. Çıkarım işleminin nasıl yürütüldüğü, Çıkarım Yöntemleri bölümünde detaylı olarak incelenecektir.

Bilgi Tabanı (knowledgebase) : Uzmardan alınan kuralların ve US'in kullanıcılarından alınan gerçeklerin barındırıldığı ortamdır. Bilgi tabanı (BT), gerçeklerin bulunduğu gerçek tabanı (factbase) ve kuralların bulunduğu kural tabanı (rulebase) olarak adlandırılan iki kısımdan oluşur. US'ler, çözüm üretmek için kuralları kullandıklarından kural tabanlı sistemler (rule based systems) olarak da adlandırılır. BT'ndaki bilginin, ÇM'ndan bağımsız olduğu unutulmamalıdır.

BT, uzmandan alınan bilginin, IF-THEN yapısındaki kurallara dönüştürülmesi ile oluşturulur. ÇM'nın, şekil uyuşması yöntemi ile ara ve ana sonuçlara ulaşması IF-THEN yapısı ile sağlanır.

BT'ları, ÇM'ndan bağımsız elemanlar olduğundan, içlerindeki bilginin değiştirilmesi veya güncellenmesi kolaydır. Eldeki bilgide bir değişiklik olduğunda BT'na yeni kurallar eklenebilir, BT'ndan birtakım kurallar çıkartılabilir veya mevcut kuralların içerikleri değiştirilebilir. ÇM, yürüttüğü işlem sırasında çakışma veya uyuşmaları aradığından kuralların nasıl sıralandığı önemli değildir. Bir US'in yapısı Şekil 3.3'de gösterilmiştir [1]:



Şekil 3.3 Uzman sistemin yapısı

4. ÇIKARIM YÖNTEMLERİ

US'in sahip olduđu bilgiyi kullanarak işlem yaptığını söylemiştik. Peki nasıl ?

US bir sonuca varmak için çıkarım (inference) işlemini yürüterek, sahip olduđu bilgi parçacıkları arasında ilişkiler kurmaya çalışır. US'in bu işlemi yürüten parçasına çıkarım mekanizması demiştik.

Çıkarım işleminin nasıl yürütüldüğünü anlayabilmek için bir garajın güvenliğinden sorumlu ve yangınlar konusunda uzmanlaşmış bir kişiyi ele alalım. Garajda bir yangının çıktığını varsayalım ve yangının başlangıç anında garaj ile ilgili gerçekler şunlar olsun:

Gümüş değerli bir metaldir.

Değerli çatal bıçak takımları gümüşten yapılır.

Radyatör soğutucusu etilen-glikol içerir.

Asbestos kanserojen bir maddedir.

Radyatör soğutucusu otomobillerde kullanılır.

Otomobil garajda korunur.

Uzmanımız, garajda yaptığı araştırmada bir çatal bıçak takımının açıkta bırakıldığını tespit etmiş olsun. Olay ile ilgili gerçeklerin birbirlerini tamamlar nitelikte olmadığı ve hatta bazılarının problemimiz ile hiçbir ilişkisinin olmadığı söylenebilir. Ancak uzmanımızın, bir yangının çıktığı gerçeğinden hareketle tüm gerçekleri değerlendirmesi ve yangının nedeni olarak yeni bir gerçeği ortaya koyması gerekir.

Bizim ele aldığımız örnekte bu gerçek "etilen-glikol, gümüş ile tepkimeye girer" şeklinde ortaya çıkabilir.

Birbirleri ile ilişkili gerçekler ortaya konunca, olayın gelişimini yansıtan senaryo oluşturulmuş demektir. Yangın olayında bu senaryo şu biçimdedir; "etilen-

glikol içeren radyatör soğutucusu, gümüşten yapılmış çatal bıçak takımının üzerine dökülmüş ve Etilen-Glikol, gümüş tepkimesi yangına neden olmuştur".

Bu olayda uzmanımız etkin bir yöntem ile sonuca varmıştır. Bu yöntem; "gerçeklerin toplanması ve bu gerçekler ile ilgili kuralların incelenmesi şeklindedir". Örneğimizde, etilen-glikol ve gümüşün tepkimeye giren maddeler olması yangının sebebini ortaya çıkaran kuraldır [1].

Bu aşamada değinmemiz gereken bir diğer nokta, uzmanımızın bilgisinin geçici bilgi (temporary knowledge) ve kalıcı bilgi (permanent knowledge) şeklinde iki kısma ayrılmasıdır. Garajın o andaki durumunu yansıtan gerçekler geçici, uzmanın yangınlar konusundaki genel bilgisi ise kalıcı bilgiyi oluşturur. Kurallar, genelde kalıcı bilgi sınıfına girerken, gerçekler, geçici veya kalıcı bilgi içinde yer alabilir[1].

4.1. Çıkarım Yöntemleri (inference methods)

ÇM'nın kuralları ve gerçekleri birleştirerek çözüm ürettiğini biliyoruz.

Birden fazla kural ve gerçek birleştirilirse ne olur? US doğru bir çıkarım için hangi kuralları ateşlemelidir? Yanlış bir seçim, US'i yanlış sonuçlara yönlendirmez mi?

US'in bu tür açmazlara girmemesi için çıkarım sürecinin hangi aşamasında hangi kuralın ele alınacağını kesin olarak belirlenebilmesi gerekir.

ÇM, çıkarımı sırasında amaç (goal) olarak adlandırılan bir önermenin doğruluğunu veya yanlışlığını belirlemeye çalışır. Amaç, doğruluk değerinin belirlenmesi gereken bir gerçektir.

Bir sonuca varmak için birden fazla kuralın ispat edilmesi gerekebilir. Bu durumda Şekil 4.1'dekine benzer bir ağ (network) ortaya çıkar [1].

Gerçek tabanı : A, B

Kural tabanı :

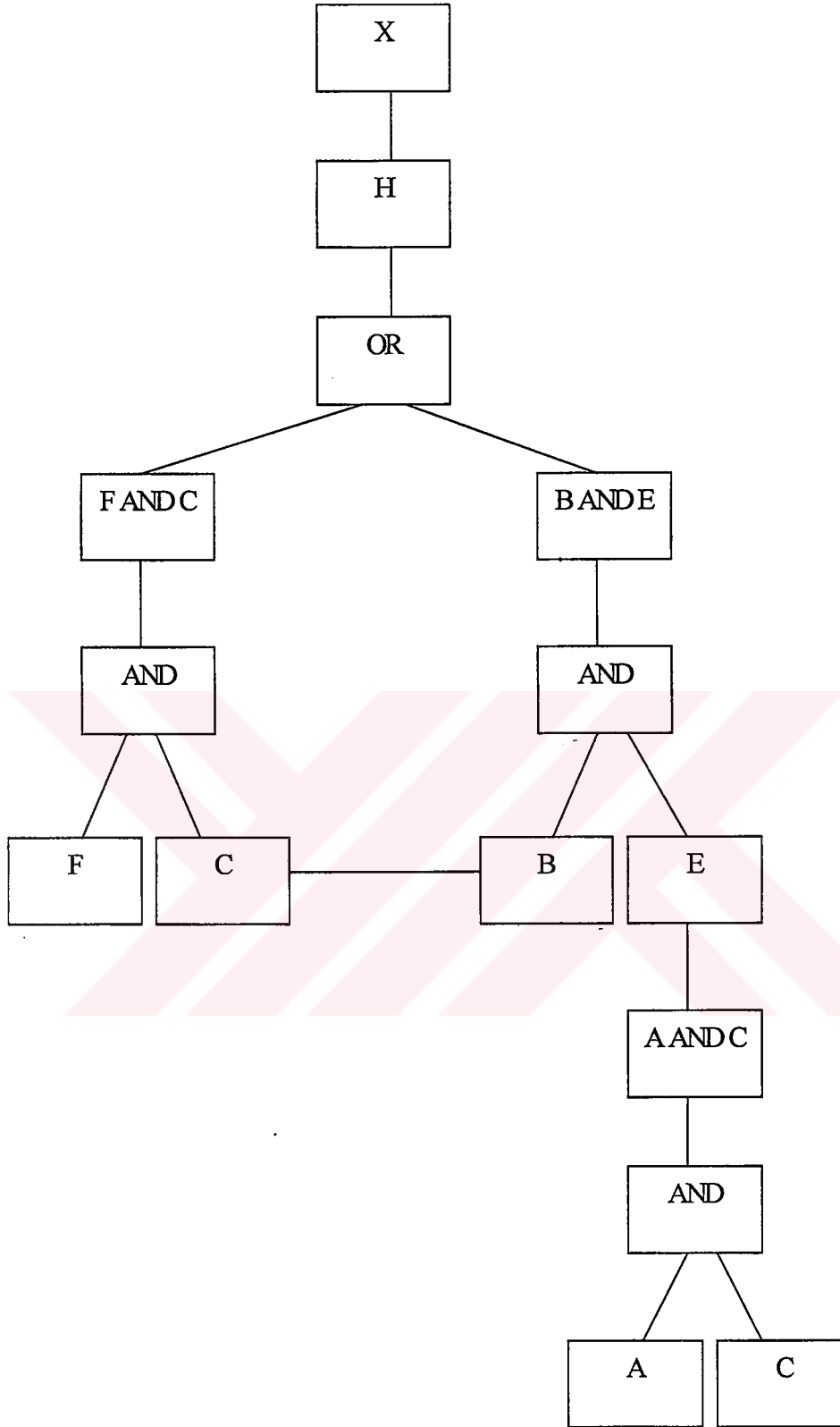
a) IF A AND C THEN E

d) IF B THEN C

b) IF F AND C THEN H

e) IF H THEN X

c) IF B AND H THEN H



Şekil 4.1 Kurallardan oluşmuş bir ağ yapısı

Ağların, çıkarımda nasıl bir rol oynadığını şu örnekte inceleyelim [1]: İstanbul'dan Tokyo'ya gitmek isteyen bir yolcu olduğumuzu varsayalım ve bu iki şehir arasındaki tüm doğrudan uçuşlar iptal edilmiş olsun. Tokyo'ya aktarmalı olarak

gitmek zorunda kalacağımızdan yapmamız gereken İstanbul'dan başlayan, Tokyo'da son bulan ve bu iki kent arasındaki kentlere uğrayan uçakların seferlerinden kurulu bir ağ yaratmaktır. Bu ağı iki şekilde oluşturulabiliriz:

İleri yönde : İstanbul'dan ayrılan uçakları ve bu uçakların gittikleri kentleri dikkate alırız. Sonra bu varış kentlerinden kalkan uçakları dikkate alarak aynı işlemi, bu kez de bu uçaklar için uygularız. Bu süreci Tokyo'ya ulaşınca kadar devam ettiririz.

Geri yönde : Tokyo'ya gelen uçakları dikkate alır ve bunların Tokyo'ya hangi kentlerden geldiklerini belirleriz. Daha sonra bu kentlerin her birini ele alıp buralara hangi kentlerden seferlerin olduğunu buluruz. Bu işleme İstanbul'a ulaşınca kadar devam ederiz.

US mantığında, ilk yöntem ileri yönlü çıkarım (forward chaining) ikinci yöntem ise geri yönlü çıkarım (backward chaining) olarak adlandırılır [1].

Peki bu iki yöntemden hangisi daha kullanışlıdır ?

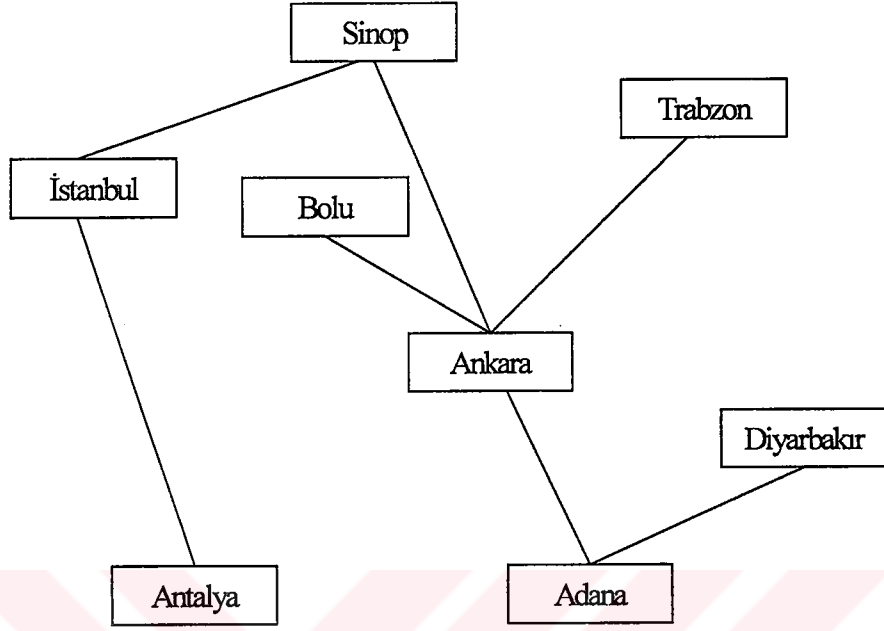
US'i hangi yöntemin daha kısa sürede doğru çözüme götüreceğini probleme özgü gerçekler belirler. Vereceğimiz örnek bunu açıkça ortaya koymaktadır. Şekil 4.2'de çeşitli kentler arasındaki uçuş koridorları verilmiştir. Bu örneğin doğal olarak gerçeğe bir ilgisi yoktur.

İlk durumda Diyarbakır'da bulunan ve aktarmalı olarak Ankara'ya gitmek isteyen bir yolcu ele alalım. İleri ve geri yönlü çıkarım yöntemleri ile elde edilen ağlar Şekil 4.3'de gösterilmiştir.

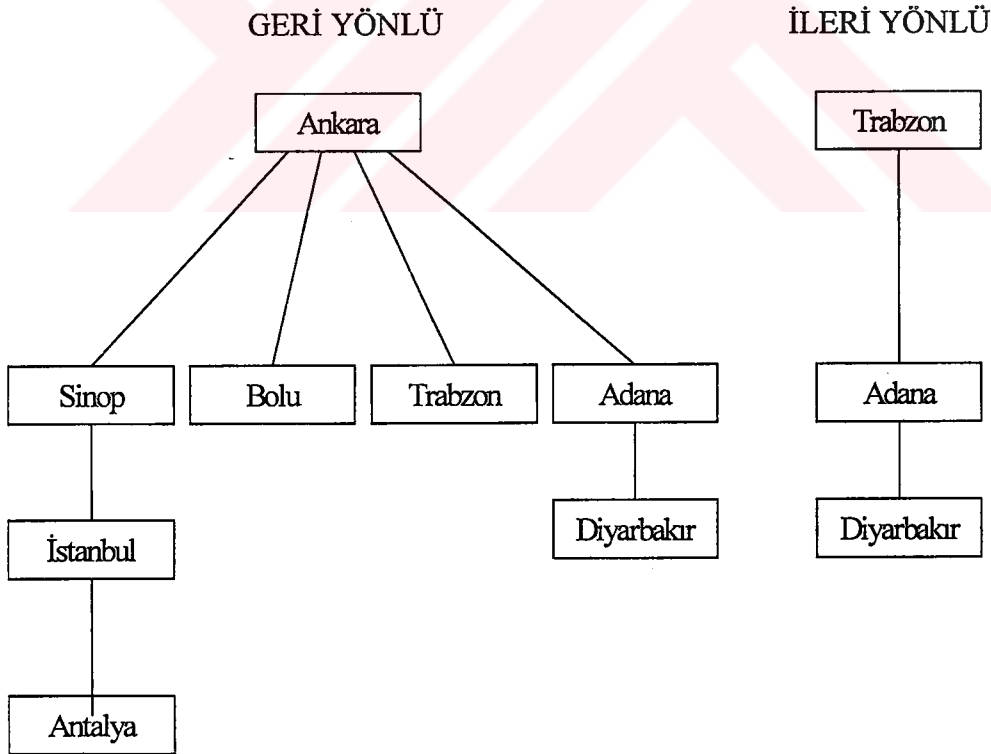
Bizim amacımız, elde edilen ağlarda uç noktaları Diyarbakır ve Ankara olan bir dal bulmaktır. Bu dalı, geri yönlü çıkarım ile oluşturulan ağda bulmak daha zordur.

Problem, Sinop'tan Bolu'ya gitmek şeklinde olsaydı elde edilen ağlar Şekil 4.4'deki gibi olacaktı. Bu durumda, elde edilen ağlar arasındaki yapısal farklılık Şekil 4.3'deki kadar açık olmamakla birlikte ileri yönlü çıkarım yöntemi ile elde edilen ağın tepesindeki ayrım çıkarımı daha başlangıcında yanlış yönlendirebilecek niteliktedir.

Diyarbakır-Ankara örneğinde ileri yönlü, Sinop-Bolu örneğinde ise geri yönlü çıkarım yönteminin etkin olmasının tek nedeni, yolcumuzun bulunduğu ve gideceği kentlerin nereleri olduğu, kısacası problemimizin gerçekleridir.

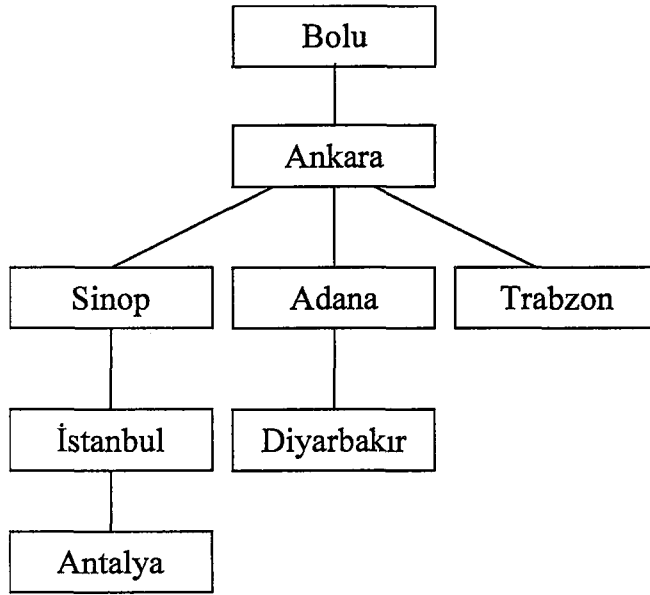


Şekil 4.2 Türkiye’deki bir uçuş ağı

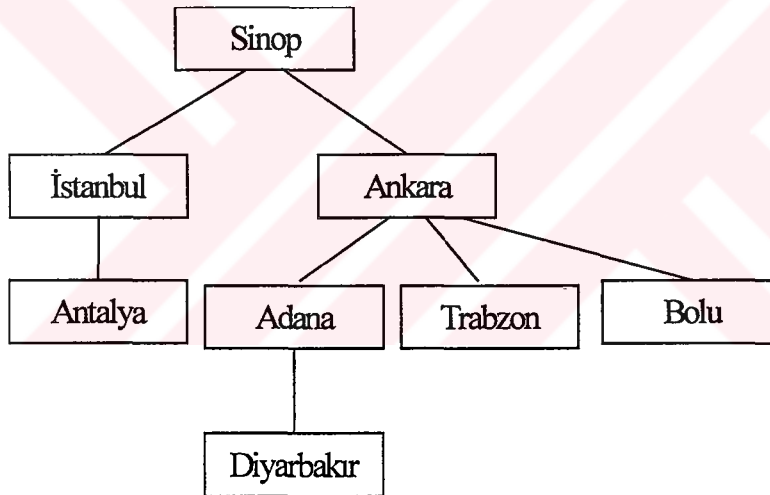


Şekil 4.3 Ankara-Diyarbakır uçuş ağı

GERİ YÖNLÜ



İLERİ YÖNLÜ



Şekil 4.4 Sinop-Bolu uçuş ağı

4.1.1. İleri Yönlü Çıkarım (forward chaining)

İleri yönlü çıkarım (İYÇ), çıkarım sürecine birtakım gerçekler ile başlar ve bazı yeni gerçekler üretir. Temel olarak İYÇ, “Modus Ponens” kalıbının uygulamasıdır. Modus Ponens kalıbı (4.1)’de verilmiştir [1]:

$$(a \wedge (a \Rightarrow b)) \Rightarrow b \quad (4.1)$$

Bu yöntem genellikle çıkarım sürecinin başlangıcında birkaç tane gerçeğin, buna karşılık doğrulukları ispat edilecek olan çok sayıda önermenin olduğu problemlerde tercih edilir.

İYÇ yöntemini kullanan sistemler, içerdikleri gerçekler ile kuralların koşul kısımlarını karşılaştırır. Herhangi bir gerçek ile bir kuralın koşul kısmı arasında şekil uyuşması sağlandığında, o kuralın sonuç kısmındaki önermeler, yeni birer gerçek olarak gerçek tabanına yerleştirilir. Bu işlem, herhangi bir kural ateşlenmeyinceye kadar çevrimler halinde devam eder.

İYÇ'in adımları şu şekildedir [1]:

1. ÇM, bilgi tabanındaki kuralları sırayla kontrol eder. Koşul kısmındaki önermeler, mevcut gerçeklerle (current fact collection) çakışan kurallar ateşlenir (fired) ve ateşlenen kuralların sonuçları bilgi tabanına yerleştirilir.
2. Ateşlenen her kural, bir veya birden fazla önermenin doğruluğunu ispatlar ve bu yeni gerçekler daha başka kuralların ateşlenmesine neden olabilir.
3. ÇM, kural listesinin sonuna ulaştığında başa dönerek bu işleme devam eder.
4. Hiçbir kural ateşlenmesinin yapılmadığı bir çevrim gerçekleştiğinde, işlem sona erer.

4.1.2. Geri Yönlü Çıkarım (backward chaining)

Geri yönlü çıkarım (GYÇ) işlemine, ana amacın (main goal) tanımlanması ile başlanır. Sonraki adımlarda ana amaç, bir veya birden fazla alt amaca (sub goal) indirgenir. Bu alt amaçlar, problemin başlangıcındaki gerçeklere daha yakın olduklarından, daha kolay ispatlanır nitelikte olabilir.

Bu aşamadan sonra her alt amaç artık bir ana amaç haline gelir ve çıkarım süreci, bu alt amaçların her birinin doğruluğunun yine GYÇ yöntemi ile elde edilmesine veya alt amaçlardan en az birinin doğruluğunun bulunamamasına kadar sürdürülür. GYÇ, “Modus Tolens” kalıbının uygulamasıdır. Modus Tolens kalıbı (4.2) de verilmiştir [1]:

$$(\neg b \wedge (a \Rightarrow b)) \Rightarrow \neg a \quad (4.2)$$

GYÇ, İYÇ'in aksine, amaçların sayısının az, çıkarımın başlangıcındaki gerçeklerin sayısının daha fazla olduğu durumlarda kullanılır.

GYÇ'in adımları şu şekildedir [1]:

1. Ana amaç veya ana amaçlar belirlenir.
2. ÇM, ana amaç veya ana amaçları gerçek tabanında arar. Ana amaç veya ana amaçlar gerçek tabanında bulunurlarsa, problem zaten çözülmüş demektir.
3. Ana amaç veya ana amaçlar gerçek tabanında bulunamazlarsa, sonuç kısmında ana amaç veya ana amaçları içeren kurallar belirlenir.
4. Bu kural veya kuralların koşul kısmındaki önermelerin doğruluk değerleri araştırılır. Bu önermeler, bilgi tabanında birer gerçek olarak yer almıyor ise, birer alt amaç haline gelir ve doğruluk değerleri 2. ve 3. adımlarda belirtilen şekilde araştırılır.
5. Bu işleme, koşul kısmındaki şartlar sağlanan bir kural bulununcaya kadar devam edilir. Zaman zaman kullanıcıya bazı soruların sorulması da gerekebilir.
6. Bu süreç, en son alt amacın doğruluk değerinin bulunduğu veya incelenecek başka kural veya gerçeğin bulunamadığı aşamada sona erer.

5. BULANIK KÜMELER

Bulanık küme teorisi (fuzzy set theory), belirsizliğin, matematiksel olarak temsil edilmesinde olarak kullanılır. Çoğu insan, özellikle fiziksel süreçlerin modellemesini yapanlar, problemin çözümünde kullandıkları bilginin eksikliğini sıklıkla farkeder. Çözüm sürecinde kullanılan bilginin çoğu, sayısal veriden daha çok çözümü üreten kişinin verdiği hükümler, yaptığı varsayımlar şeklindedir. O halde insanın sezgi gücü (intuition), problemin çözüm sürecine bir şekilde dahil edilmelidir [2]. Bu nasıl mümkün olabilir ?

Günlük yaşantımızda kullandığımız doğal dil (natural language), bu işlem için kullanılabilecek en etkili araçlardan biridir. Konuşma dilleri belirsizlik içerir ve insanlar arasında bilgi değişiminde kullanılan bir numaralı öğedir. İnsanlar, doğal dildeki belirsizliğe rağmen, birbirlerinin düşüncelerini anlamakta fazla zorluk çekmez. Bu, kesin ve tam veri ile işlem yapan bilgisayarlar ile iletişim kurarken geçerli değildir.

Bulanık kümeler, bu yüksek lisans tez çalışmasına, Ağırlıklı Bulanık Çıkarım Algoritması ve Geliştirilen Uzman Sistem bölümünde detaylı biçimde incelenecek olan Ağırlıklı Bulanık Çıkarım Algoritması'yla dahil edilir. Bu algoritma, yüksek lisans tez çalışması olarak geliştirilmiş olan US tarafından kullanılmakta olup doğal dildeki belirsizliğin (önergelerin doğruluk değerlerinin, ağırlıklarının ve kuralların belirginlik çarpanlarının belirsizliği) matematiksel olarak temsil edilmesini gerektirmektedir [4].

Örneğin “uzun insan” önermesini ele alalım. Bu önerme, A kişisi için “1.70 m boyunda veya daha uzun boylu insanları” temsil ederken, B kişisi için “2.00 m boyunda veya daha uzun boylu insanlar” anlamına gelebilir. O halde “uzun” sözcüğü bu iki insan için farklı anlam taşır.

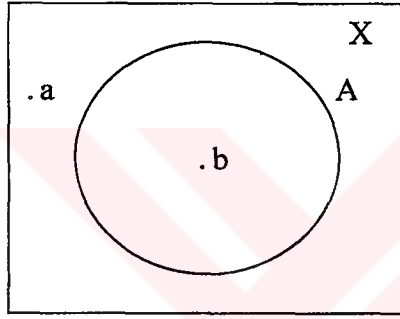
Bulanık küme teorisinin altında yatan güç, belirsiz kavramların temsil edilmesinde nicel değişkenlerden (quantitative variables) çok, dilsel değişkenlerin (linguistic variables) kullanılmasıdır [2].

5.1. Belirgin Kümeler Ve Bulanık Kümeler (crisp sets & fuzzy sets)

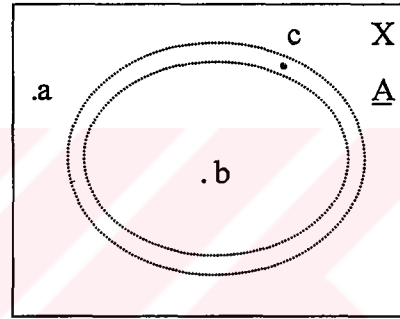
Evrensel küme (universe of discourse), sözkonusu problem hakkında edinilmiş tüm bilginin uzayıdır. Bu uzay tanımlandıktan sonra, uzaydaki olayları (events), kümeler aracılığıyla tanımlamak mümkün olur.

Belirgin kümeler (crisp sets), belirgin sınırlar ile tanımlanır ve bu sınırların evrensel kümedeki yerleşiminde herhangi bir belirsizlik yoktur. Öte yandan, bulanık kümeler (fuzzy sets), belirsiz sınırlar ile tanımlanır ve bu sınırların evrensel kümedeki yerleşimi de belirsizdir [2].

X evrensel kümesinde tanımlanmış, A belirgin kümesi ve $\underline{A}$ bulanık kümesi sırasıyla Şekil 5.1 ve Şekil 5.2’de gösterilmiştir.



Şekil 5.1 A belirgin kümesi



Şekil 5.2 $\underline{A}$ bulanık kümesi

Şekil 5.1’de, b noktası A kümesinin kesinlikle bir elemanı iken, a noktası A kümesinin bir elemanı kesinlikle değildir.

Şekil 5.2’de, $\underline{A}$ kümesi belirgin olmayan sınırlar ile gösterilmiştir. Yine b noktası $\underline{A}$ kümesinin kesinlikle bir elemanı iken, a noktası $\underline{A}$ kümesinin bir elemanı kesinlikle değildir. Ancak c noktasının $\underline{A}$ kümesine üyelik derecesi konusunda kesin birşey söylenemez. Eğer herhangi bir kümenin kesinlikle üyesi olmak 1, üyesi kesinlikle olmamak 0 ile gösterilecek olursa, c noktasının $\underline{A}$ kümesine üyelik derecesi (0,1) aralığında bir değere karşılık gelmelidir. Bu değer; c noktası, $\underline{A}$ kümesinin merkezine yaklaştıkça 1’e giderken, $\underline{A}$ kümesinin sınırlarına yaklaştıkça 0’a gider [2] .

5.1.1. Belirgin Kümeler

Belirgin kümeler teorisinde, evrensel kümedeki bir elemanın belirgin kümenin elemanı olması veya olmaması durumları ($\in$ veya $\notin$) arasındaki geçiş de belirgindir [2]. Bunu bir örnek ile açıklayalım:

$X: \{\text{insanların boyları}\}, x \in X$ ve $A \subset X$ olmak üzere;

$$A = \{x \in X \mid 1.80 \leq x \leq 1.90\} \quad (5.1)$$

olarak tanımlansın.

$\forall x \in X$ 'in A kümesinin bir elemanı oluşu veya olmayışı $\chi_A(x)$ fonksiyonu ile gösterilir. $\chi_A(x)$ karakteristik fonksiyon (characteristic function) olarak adlandırılır ve (5.2) ile tanımlanır [2].

$$\chi_A(x) = \begin{cases} 1, & x \in A \\ 0, & x \notin A \end{cases} \quad (5.2)$$

$x_1 = 1.85$ m ve $x_2 = 1.9001$ m'yi ele alalım: $1.80 \leq x_1 \leq 1.90$ ve $x_2 > 1.90$ olduğundan $\chi_A(x_1) = 1$ ve $\chi_A(x_2) = 0$ 'dır.

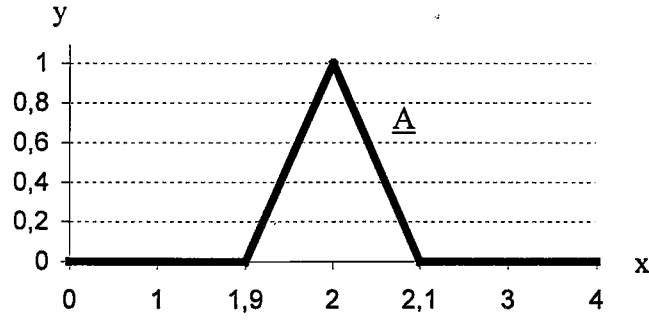
5.1.2. Bulanık Kümeler

Bulanık kümeler söz konusu olduğunda ise, x elemanının $\underline{A}$ bulanık kümesine üyeliğinin derecesi (degree of membership), $[0,1]$ aralığındaki bir sayı ile gösterilir. Aralığın uç noktaları olan 0 ve 1 değerleri (belirgin kümeler teorisinde olduğu gibi) sırasıyla, "elemanı olmamak" ve "tam elemanı olmak" durumlarına karşılık gelir.

Bir bulanık küme olarak, $\underline{A} = \{2.00 \text{ m civarındaki insan boyları}\}$ kümesini tanımlayalım. $x_1 = 2.00$ m değerinin $\underline{A}$ kümesine üyelik derecesi 1 iken, $x \neq x_1$ olduğu durumlarda bu üyelik derecesi azalacaktır. $\underline{A}$ bulanık kümesinin üyelik fonksiyonu (membership function) $y = \mu_{\underline{A}}(x)$, Şekil 5.3'de görülmektedir.

Bulanık kümeler içeren bir evrensel kümedeki bir elemanın, bulanık kümenin elemanı olması veya olmaması durumları arasındaki geçiş, aşamalı olarak gerçekleşir. Buna neden olarak; bulanık kümelerin belirgin olmayan sınırlarını verebiliriz. Evrensel kümedeki bir elemanın, bulanık kümeye üyelik derecesi, eleman oluşundaki belirsizliği gösteren üyelik fonksiyonu ile belirlenir [2].

Bulanık küme, değişen üyelik derecelerine sahip elemanlardan oluşan küme olarak tanımlanabilir. Bu kümenin elemanlarının üyelik dereceleri $[0,1]$ aralığında olacağından, söz konusu elemanlar bir başka bulanık kümenin elemanları da olabilir.



Şekil 5.3 $\underline{A}$ kümesinin üyelik fonksiyonu

Üyelik fonksiyonu, kümenin her bir elemanına $[0,1]$ aralığında bir üyelik derecesi karşılık getirir. $\forall x \in \underline{A}, \mu_{\underline{A}}(x) \in [0,1]$

Bulanık kümelerin gösterimi, evrensel kümenin sonlu veya sonsuz elemanlı oluşuna göre iki biçimde yapılabilir.

X sonlu elemanlı ise;

$$\underline{A} = \{\mu_{\underline{A}}(x_1)/x_1 + \mu_{\underline{A}}(x_2)/x_2 + \dots + \mu_{\underline{A}}(x_n)/x_n\} = \sum_{i=1}^n \mu_{\underline{A}}(x_i)/x_i \quad (5.3)$$

X sonsuz elemanlı ise;

$$\underline{A} = \int \mu_{\underline{A}}(x)/x \quad (5.4)$$

şeklinde gösterilir. Bu gösterimlerde kullanılan toplam ve integral sembolleri cebirsel anlam taşımamaktadır. Paydadaki x_i değişkenleri X'deki elemanları, paydaki $\mu_{\underline{A}}(x_i)$ değeri ilgili değişkenin $\underline{A}$ kümesine üyelik derecesini gösterir [2].

İkinci bir örnek olarak; $X = \{\text{sıcaklık değerleri}\} = \{0, 20, 40, 60, 80, 100\}$ evrensel kümesini ele alalım. X' de Soğuk, Ilık, Sıcak, Çok Sıcak bulanık kümeleri tanımlayacak olursak Tablo 5.1'i elde ederiz.

Buna göre;

$$\text{Soğuk} = \{1/0 + 0.7/20 + 0.6/40 + 0.3/60 + 0.2/80 + 0/100\}$$

$$\text{Ilık} = \{0.7/0 + 0.8/20 + 1/40 + 0.6/60 + 0.4/80 + 0.2/100\}$$

$$\text{Sıcak} = \{0.1/0 + 0.3/20 + 0.4/40 + 1/60 + 0.6/80 + 0.9/100\}$$

$$\text{Çok Sıcak} = \{0/0 + 0.1/20 + 0.2/40 + 0.6/60 + 0.9/80 + 1/100\}$$

şeklinde temsil edilir.

Tablo 5.1 X’de tanımlı bazı bulanık kümeler

Değerler	Soğuk	Ilık	Sıcak	Çok Sıcak
0	1	0.7	0.1	0
20	0.7	0.8	0.3	0.1
40	0.6	1	0.4	0.2
60	0.3	0.6	1	0.6
80	0.2	0.4	0.6	0.9
100	0	0.2	0.9	1

5.1.2.1. Üyelik Fonksiyonunun Özellikleri

Bulanık kümenin içerdiği bilginin üyelik fonksiyonuyla tanımlanmasından dolayı, bu fonksiyon ile ilgili tanımların incelenmesinde fayda vardır.

Çekirdek (core) : $\underline{A}$ bulanık kümesinin üyelik fonksiyonunun çekirdeği, “evrensel kümesinin $\underline{A}$ bulanık kümesine tam olarak üye olan elemanlarının bölgesi” olarak tanımlanır [2].

$$\text{core}(\underline{A}) = \{ x \in X \mid \mu_{\underline{A}}(x) = 1 \} \quad (5.5)$$

Destek (support) : $\underline{A}$ bulanık kümesinin üyelik fonksiyonunun desteği, “evrensel kümesinin $\underline{A}$ bulanık kümesine üyelik derecesi 0 (sıfır)’dan büyük olan elemanlarının bölgesi” olarak tanımlanır [2].

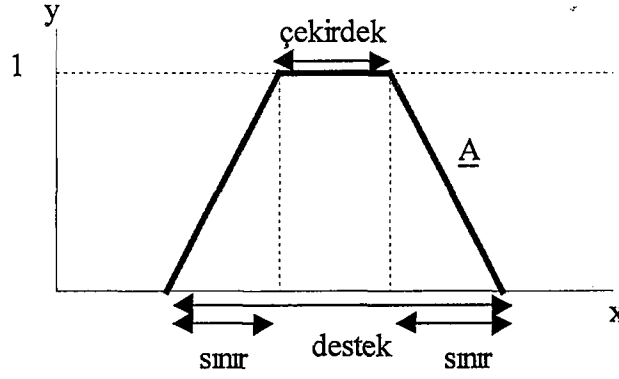
$$\text{support}(\underline{A}) = \{ x \in X \mid \mu_{\underline{A}}(x) > 0 \} \quad (5.6)$$

Sınır (boundary) : $\underline{A}$ bulanık kümesinin üyelik fonksiyonunun sınırı, “evrensel kümesinin $\underline{A}$ bulanık kümesine üyelik derecesi 0 (sıfır)’dan büyük olan ancak $\underline{A}$ bulanık kümesine tam olarak üye olmayan elemanlarının bölgesidir [2].

$$\text{boundary}(\underline{A}) = \{ x \in X \mid 0 < \mu_{\underline{A}}(x) < 1 \} \quad (5.7)$$

$\underline{A}$ kümesinin çekirdek, destek ve sınır bölgesi Şekil 5.4’de gösterilmiştir ($y = \mu_{\underline{A}}(x)$).

Yükseklik (height) : $\underline{A}$ bulanık kümesinin yüksekliği, evrensel kümedeki elemanların $\underline{A}$ kümesine en büyük üyelik derecesi ($\max(\mu_{\underline{A}}(x))$) olarak tanımlanır [2].



Şekil 5.4 A bulanık kümesinin çekirdek, sınır ve destek bölgesi

Kardinalite (cardinality) : A bulanık kümesinin elemanlarının üyelik derecelerinin toplamı A kümesinin kardinalitesi olarak tanımlanır. Bu değer $|\underline{A}|$ ile gösterilir ve (5.8) ile hesaplanır.

$$|\underline{A}| = \sum \mu_{\underline{A}}(x) \quad (5.8)$$

Bulanık alt küme (sub fuzzy set) : A, B bulanık kümeleri için,

$$\forall x \in X, \mu_{\underline{A}}(x) \leq \mu_{\underline{B}}(x) \quad (5.9)$$

şartı sağlanıyorsa A, B'nin bulanık alt kümesi olarak tanımlanır [2].

Normal Bulanık Küme (normal fuzzy set) : A bulanık kümesinin elemanlarından en az biri, bu kümeye tam olarak üye ise ($\exists x \in X, \mu_{\underline{A}}(x) = 1$), A normal bulanık küme olarak tanımlanır [2].

Konveks Bulanık Küme (convex fuzzy set) : A bulanık kümesinin elemanlarının üyelik dereceleri, evrensel kümenin değerleri artan elemanları dikkate alındığında, monoton artıyor veya monoton azalıyor veya önce monoton artıyor, sonra da monoton azalıyor ise, A konveks bulanık küme olarak tanımlanır. Bir başka deyişle $x < y < z$ için

$$\mu_{\underline{A}}(y) \geq \min[\mu_{\underline{A}}(x), \mu_{\underline{A}}(z)] \quad (5.10)$$

ise, A konveks bulanık kümedir [2].

Bulanık Sayı (fuzzy number) : A bulanık kümesi normal ve konveks bulanık küme ise, bulanık sayı olarak tanımlanır.

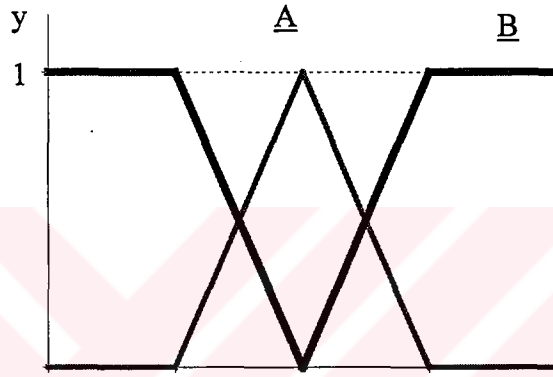
5.1.2.2. Bulanık Kümelerle İlgili İşlemler

X evrensel kümesinin, $\underline{A}$ ve $\underline{B}$ bulanık alt kümelerini inceleyelim. $\underline{A} \cup \underline{B}$, $\underline{A} \cap \underline{B}$ gibi diğer bulanık kümelerin tanımlaması şu biçimde yapılır:

Bulanık Tümlen : $\underline{A}$ bulanık kümesinin tümleniyi $\neg \underline{A}$ ile gösterilir ve $\forall x \in X$ için $\mu_{\underline{A}}(x)$; x elemanının $\underline{A}$ kümesine üyelik derecesi olmak üzere, x elemanının $\neg \underline{A}$ kümesine üyelik derecesi $\mu_{\neg \underline{A}}(x)$, (5.11) ile hesaplanır [2]:

$$\mu_{\neg \underline{A}}(x) = 1 - \mu_{\underline{A}}(x) \quad (5.11)$$

Şekil 5.5'de $\underline{A}$ bulanık kümesinin tümleniyi görülmektedir ($y = \mu_{\neg \underline{A}}(x)$).



Şekil 5.5 $\underline{A}$ bulanık kümesinin tümleniyi

Yukarıdaki örneği dikkate alacak olursak:

$$\text{Değil Ilık} = \{0.3/0 + 0.2/20 + 0.4/40 + 0.4/60 + 0.6/80 + 0.8/100\}$$

bulanık kümesi elde edilir.

Bulanık Birleşme : $\underline{A}$ ve $\underline{B}$, X de tanımlı bulanık kümeler, $\forall x \in X$ için $\mu_{\underline{A}}(x)$ ve $\mu_{\underline{B}}(x)$, x elemanının sırasıyla, $\underline{A}$ ve $\underline{B}$ bulanık kümelerine üyelik dereceleri olsun. $\underline{A} \cup \underline{B}$ de bir bulanık kümedir ve x elemanının $\underline{A} \cup \underline{B}$ kümesine üyelik derecesi $\mu_{\underline{A} \cup \underline{B}}(x)$ ile gösterilir [2].

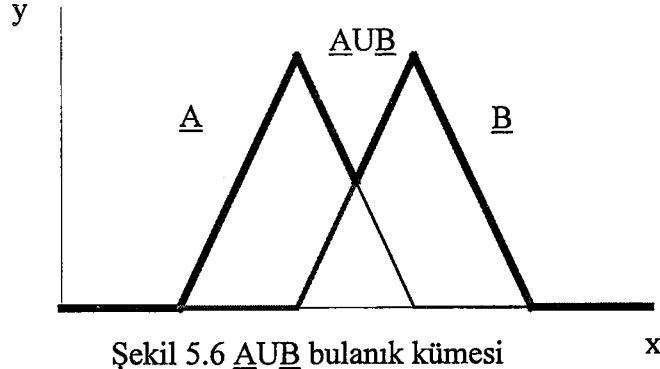
$$\mu_{\underline{A} \cup \underline{B}}(x) = \max(\mu_{\underline{A}}(x), \mu_{\underline{B}}(x)) \quad (5.12)$$

şeklinde hesaplanır. Şekil 5.6'da $\underline{A} \cup \underline{B}$ bulanık kümesi görülmektedir.

Yukarıdaki örneği dikkate alacak olursak

$$\text{Soğuk} \cup \text{Sıcak} = \{1/0 + 0.7/20 + 0.6/40 + 1/60 + 0.6/80 + 0.9/100\}$$

bulanık kümesi elde edilir.



Bulanık Kesişme : $\underline{A}$ ve $\underline{B}$, X de tanımlı bulanık kümeler, $\forall x \in X$ için $\mu_{\underline{A}}(x)$ ve $\mu_{\underline{B}}(x)$, x elemanının sırasıyla, $\underline{A}$ ve $\underline{B}$ bulanık kümelerine üyelik dereceleri olsun. $\underline{A} \cap \underline{B}$ de bir bulanık kümedir ve x elemanının, $\underline{A} \cap \underline{B}$ kümesine üyelik derecesi, $\mu_{\underline{A} \cap \underline{B}}(x)$ ile gösterilir [2].

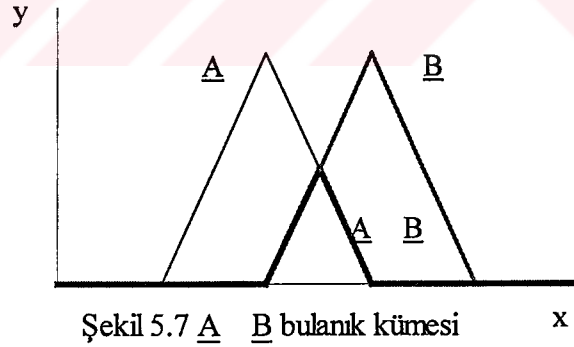
$$\mu_{\underline{A} \cap \underline{B}}(x) = \min(\mu_{\underline{A}}(x), \mu_{\underline{B}}(x)) \quad (5.13)$$

şeklinde hesaplanır. Şekil 5.7'de $\underline{A} \cap \underline{B}$ bulanık kümesi görülmektedir.

Yukarıdaki örneği dikkate alacak olursak;

$$\text{Soğuk} \cap \text{Sıcak} = \{0.1/0 + 0.3/20 + 0.4/40 + 0.3/60 + 0.2/80 + 0/100\}$$

bulanık kümesi elde edilir.



5.1.2.3. Bulanık Kümelerin Özellikleri

Bu bölümde, bulanık kümelerin özellikleri ele alınacaktır [2].

a) Değişme :

$$\underline{A} \cup \underline{B} = \underline{B} \cup \underline{A} \quad (5.14)$$

$$\underline{A} \cap \underline{B} = \underline{B} \cap \underline{A} \quad (5.15)$$

b) Birleşme :

$$\underline{A} \cup (\underline{B} \cup \underline{C}) = (\underline{A} \cup \underline{B}) \cup \underline{C} \quad (5.16)$$

$$\underline{A} \cap (\underline{B} \cap \underline{C}) = (\underline{A} \cap \underline{B}) \cap \underline{C} \quad (5.17)$$

c) Dağılma :

$$\underline{A} \cup (\underline{B} \cap \underline{C}) = (\underline{A} \cup \underline{B}) \cap (\underline{A} \cup \underline{C}) \quad (5.18)$$

$$\underline{A} \cap (\underline{B} \cup \underline{C}) = (\underline{A} \cap \underline{B}) \cup (\underline{A} \cap \underline{C}) \quad (5.19)$$

d) Geçişme :

$$\underline{A} \subseteq \underline{B} \text{ ve } \underline{B} \subseteq \underline{C} \rightarrow \underline{A} \subseteq \underline{C} \quad (5.20)$$

e) De Morgan Kuralı :

$$\neg(\underline{A} \cup \underline{B}) = \neg \underline{A} \cap \neg \underline{B} \quad (5.21)$$

$$\neg(\underline{A} \cap \underline{B}) = \neg \underline{A} \cup \neg \underline{B} \quad (5.22)$$

f)

$$\forall x \in X, \mu_{\emptyset}(x)=0 \quad (5.23)$$

g)

$$\forall x \in X, \mu_X(x)=1 \quad (5.24)$$

h)

$$\underline{A} \cup \underline{A} = \underline{A} \quad (5.25)$$

$$\underline{A} \cup \emptyset = \underline{A} \quad (5.26)$$

$$\underline{A} \cap \underline{A} = \underline{A} \quad (5.27)$$

$$\underline{A} \cap \emptyset = \emptyset \quad (5.28)$$

i)

$$\underline{A} \cup X = X \quad (5.29)$$

$$\underline{A} \cap X = \underline{A} \quad (5.30)$$

$$\underline{A} \cup \neg \underline{A} = X \quad (5.31)$$

$$\underline{A} \cap \neg \underline{A} = \phi \quad (5.32)$$

$$\neg(\neg \underline{A}) = \underline{A} \quad (5.33)$$



6. BELİRGİN BAĞINTILAR VE BULANIK BAĞINTILAR

Matematiksel tabanlı pek çok alanda, kümeler kadar bağıntılar da, büyük önem taşır. Bağıntılar mantık, yaklaşımla çıkarım, kural tabanlı sistemler, doğrusal olmayan simülasyon, şekil tanıma gibi pekçok araştırma alanında kullanılır.

6.1. Belirgin Bağıntılar

r adet elemanın bir sıralı dizisini dikkate alalım. Bu elemanlar $(a_1, a_2, \dots, a_r)$ şeklinde yazılır ve sıralı diziliş (an ordered r -tuple) olarak adlandırılır. Özel bir durum olarak, $r = 2$ kabul edildiğinde, bu sıralı dizilişe sıralı ikili (ordered pair) adı verilir. $A_1, A_2, \dots, A_r$ belirgin kümelerini dikkate aldığımızda, $a_1 \in A_1, a_2 \in A_2, \dots, a_r \in A_r$ olmak üzere, $(a_1, a_2, \dots, a_r)$ sıralı dizilişlerinin tüm kümesine; $A_1, A_2, \dots, A_r$ kümelerinin kartezyen çarpımı denir ve $A_1 \times A_2 \times \dots \times A_r$ ile gösterilir. Tüm kümeler belli bir A kümesine eşit olduğunda, bu çarpım A^r olarak gösterilebilir.

$A_1 \times A_2 \times \dots \times A_r$ kartezyen çarpımının herhangi bir alt kümesi $A_1, A_2, \dots, A_r$ üzerinde bir bağıntı olarak tanımlanır. Özel bir durum olarak yine; $r = 2$ alındığında $A_1 \times A_2$ kartezyen çarpımının her alt kümesi, ikili bağıntı (binary relation) olarak tanımlanır. X, Y iki evrensel küme olmak üzere, bu uzayların kartezyen çarpımı

$$X \times Y = \{(x, y) \mid x \in X, y \in Y\} \quad (6.1)$$

X ve Y uzaylarının elemanları arasındaki en genel, kısıtlanmamış (unconstrained) ilişkiyi gösterir. Yani, X evrensel kümesinin her elemanı, Y evrensel kümesinin her elemanı ile tam olarak (completely) ilişkilidir. X ve Y evrensel kümelerinin elemanları arasındaki daha belirgin, kısıtlanmış bir $R \subseteq X \times Y$ bağıntısı ise, 0 veya 1 değerlerini alabilen χ_R karakteristik fonksiyonu ile temsil edilir. 1 değeri x ve y elemanlarının tam olarak ilişkili, 0 değeri x ve y elemanlarının ilişkili olmadıkları durumlara karşılık gelir [2].

$$\chi_R(x, y) = \begin{cases} 1, & (x, y) \in R \\ 0, & (x, y) \notin R \end{cases} \quad (6.2)$$

Evrensel kümelerin sonlu elemanlı olduğu durumlarda bu bağıntılar bir bağıntı matrisi (relation matrix) şeklinde temsil edilebilir. Örneğin; $X=\{\text{Ankara, Paris}\}$, $Y=\{\text{Fransa, Türkiye}\}$ şeklinde ve R ; “başkenti olma” ilişkisi olarak tanımlanırsa Tablo 6.1 elde edilir

Tablo 6.1 R ilişkisi matrisi

R	Fransa	Türkiye
Ankara	0	1
Paris	1	0

6.1.1. Belirgin Bağıntı İşlemleri

R ve S , $X \times Y$ üzerinde tanımlı iki ayrı bağıntı olsun. Bu durumda aşağıdaki yeni bağıntıların tanımlanabilmesi şu biçimde yapılır [2]. (boyut=2 olarak kabul edilirse):

a) Boş bağıntı (null relation) :

$$o = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \quad (6.3)$$

b) Tam bağıntı (complete relation) :

$$E = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \quad (6.4)$$

c) Birleşim :

$$\chi_{R \cup S}(x,y) = \max[\chi_R(x,y), \chi_S(x,y)] \quad (6.5)$$

d) Kesişim :

$$\chi_{R \cap S}(x,y) = \min[\chi_R(x,y), \chi_S(x,y)] \quad (6.6)$$

e) Tümleneyen :

$$\chi_{-R}(x,y) = 1 - \chi_R(x,y) \quad (6.7)$$

f) İçerme :

$$R \subseteq S \Rightarrow \chi_R(x,y) \leq \chi_S(x,y) \quad (6.8)$$

g) Özdeşlik :

$$o, E \quad (6.9)$$

6.1.2. Belirgin Bağıntıların Kompozisyonu

X, Y, Z evrensel kümeleri R, X ve Y, S, Y ve Z evrensel kümeleri arasında tanımlı bağıntılar olsun. X ve Z evrensel kümeleri arasında tanımlı bir T bağıntısı, R ve S bağıntıları kullanılarak (6.10a), (6.10b) ile elde edilir [2]:

$$T=RoS \quad (6.10a)$$

$$\chi_T(x,z) = \bigvee_{y \in Y} (\chi_R(x,y) \wedge \chi_S(y,z)) \quad (6.10b)$$

6.2. Bulanık Bağıntılar

Tıpkı belirgin bağıntılar gibi, bulanık bağıntılar da, bir evrensel kümenin elemanlarını, bir başka evrensel kümenin elemanları ile ilişkilendirir. Ancak; bu elemanlar arasındaki ilişkinin kuvveti, karakteristik fonksiyon ile değil, ilişkinin kuvvet derecesini $[0,1]$ aralığındaki bir değer ile gösteren üyelik fonksiyonu ile ifade edilir.

$\underline{A}$, X evrensel kümesinde, $\underline{B}$, Y evrensel kümesinde tanımlı bulanık kümeler olsun. $\underline{A}$ ve $\underline{B}$ kümeleri arasında kartezyen çarpımın tanımlanması ile $\underline{R} = \underline{A} \times \underline{B} \subseteq X \times Y$ bulanık bağıntısı elde edilir. $\underline{R}$ bağıntısı, $X \times Y$ kartezyen çarpımından, $[0,1]$ aralığına bir atamadır. Evrensel kümelerin elemanları arasındaki ilişkinin kuvvet derecesini gösteren $\mu_{\underline{R}}(x,y)$ değeri (6.11) ile hesaplanır [2]:

$$\mu_{\underline{R}}(x,y) = \mu_{\underline{A} \times \underline{B}}(x,y) = \min[\mu_{\underline{A}}(x), \mu_{\underline{B}}(y)] \quad (6.11)$$

Örneğin;

$$X = \{0, 20, 40\}, Y = \{\text{nezle, hepatit, AIDS}\}$$

$$\underline{A} = \{\text{çok yüksek ateş değeri}\} = \{0/0 + 0.5/20 + 1/40\}$$

$$\underline{B} = \{\text{çok tehlikeli hastalıklar}\} = \{0/\text{nezle} + 1/\text{hepatit} + 1/\text{AIDS}\} \text{ olsun.}$$

$\underline{R} = \underline{A} \times \underline{B}$ bağıntı matrisi aşağıdaki biçimde elde edilir:

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0.5 & 0.5 \\ 0 & 1 & 1 \end{bmatrix} \quad (6.12)$$

6.2.1. Bulanık Bağıntı İşlemleri

$\underline{R}$ ve $\underline{S}$, $X \times Y$ kartezyen çarpımında tanımlı iki bulanık bağıntı olsun [2].

a) Birleşim:

$$\mu_{\underline{R} \cup \underline{S}}(x,y) = \max[\mu_{\underline{R}}(x,y), \mu_{\underline{S}}(x,y)] \quad (6.13)$$

b) Kesişim:

$$\mu_{\underline{R} \cap \underline{S}}(x,y) = \min[\mu_{\underline{R}}(x,y), \mu_{\underline{S}}(x,y)] \quad (6.14)$$

c) Tümleneyen :

$$\mu_{\neg \underline{R}}(x,y) = 1 - \mu_{\underline{R}}(x,y) \quad (6.15)$$

d) İçerme :

$$\underline{R} \subset \underline{S} \Rightarrow \mu_{\underline{R}}(x,y) \leq \mu_{\underline{S}}(x,y) \quad (6.16)$$

6.2.2. Bulanık Bağıntıların Kompozisyonu

X , Y , Z evrensel kümeleri $\underline{R}$, X ve Y , $\underline{S}$, Y ve Z arasında tanımlanmış bulanık bağıntılar olsun. X ve Z evrensel kümeleri arasında tanımlanmış bir $\underline{T}$ bulanık bağıntısı $\underline{R}$ ve $\underline{S}$ bulanık bağıntıları kullanılarak (6.17a), (6.17b)'ile elde edilebilir [2]:

$$\underline{T} = \underline{R} \circ \underline{S} \quad (6.17a)$$

$$\mu_{\underline{T}}(x,z) = \bigvee_{y \in Y} (\mu_{\underline{R}}(x,y) \wedge \mu_{\underline{S}}(y,z)) \quad (6.17b)$$

Unutulmamalıdır ki;

$$\underline{R} \circ \underline{S} \neq \underline{S} \circ \underline{R} \quad (6.18)$$

7. BULANIK MANTIK

Klasik, diğ er adıyla; iki değ erli mantığın (two valued logic)  zerine oturduėu temel prensip, “her  nermenin doėru veya yanlış olmasıdır”. Aristo, anlık  nermelerin doėruluklarının yanısıra gelecekte yařanacak olayların doėruluėuyla da ilgilenmiř ve bu olaylarla iliřkili  nermelerin hem doėru hem de yanlış olabileceėi sonucuna varmıřtır. Dolayısıyla bu  nermelerin doėruluėu belirsizdir.

Daha ilerideki d nemlerde ger ekleřtirilen bilimsel geliřmeler,  nermelerin doėruluklarının 0 veya 1 değ erlerine kısıtlı kalmadan g sterilmesini zorunlu kılmıř ve sonu ta    değ erli mantık (three valued logic) modelleri ortaya atılmıřtır. Bu     nc  değ ere belirsiz (indeterminate) adı verilmiřtir.

 ki değ erli mantığın,    değ erli mantıėa geniřletilmesi farklı y ntemlerle yapılmıř ve sonu ta kendi i inde kuralları olan mantık modelleri oluřturulmuřtur. Bu modellerde ortak olan unsur mutlak doėruluėun 1, mutlak yanlışlıėın 0 ve belirsiz doėruluk değ erinin $\frac{1}{2}$ ile g sterilmesidir. Bu modellere  rnek olarak, Lukasiewicz, Bochvar, Kleene, Heyting, Reichenbach tarafından kurulan modelleri verebiliriz. Ancak bu modellerin hi biri iki değ erli mantıėın temelini oluřturan $\neg a \vee a = 1$, $\neg a \wedge a = 0$ kurallarını ve iki değ erli mantıėın bazı totolojilerini doėrulamamıřtır.

Bulanık mantık, 60’lı yılların ortalarında L tfi Zadeh tarafından    değ erli mantıėa bir alternatif olarak ortaya atılmıřtır. Bulanık mantık, klasik mantıėı ve klasik k me teorisini, bir,  ok değ erli mantık (multi-valued logic) modelinin  zel durumları olarak ele alır.

Bulanık mantıėın temel amacı, bulanık k me teorisini bir ara  olarak kullanarak, yaklařımla  ıkarım yapma (approximate reasoning) iřlemine olanak saėlamaktadır. Yaklařımla  ıkarım yapma s recinde, belirsiz  nermeler ile  ıkarım yapmak s z konusu olduėundan, bulanık mantıėın odak noktası doėal dildir.

Bulanık mantık belirsizlik altında  ıkarım yapabilmek i in seyrek, eski, gibi bulanık y klemlerin (fuzzy predicates), genellikle, az daha gibi bulanık

niceliyicilerin (fuzzy quantifiers), çok doğru, orta derecede doğru gibi bulanık doğruluk değerlerinin (fuzzy truth values) kullanılmasına olanak sağlar [3].

7.1. Yaklaşım ile Çıkarım (approximate reasoning)

Günlük yaşantımızda karşılaştığımız, doğal konuşma diliyle ifade edilen önermeler, kişiden kişiye değişen anlamlar taşır. Yani belirsizlik içerir. “Uzun boylu insan” dendiğinde, farklı kişilerin zihinlerinde farklı kavramlar şekillenir.

Yukarıdaki örneğe benzer biçimde, konuşma diliyle ifade edilen pek çok söylem, birer bulanık önermedir (fuzzy proposition).

Bulanık önermelerin doğruluk değerlerini, bu önermelere bulanık kümeler karşılık getirerek elde edebiliriz [3]. Örneğin;

$$X = \{\text{insanların boyları}\} = \{1.50, 1.60, 1.70, 1.80, 1.90\}$$

$\underline{A} = \{\text{orta insan boyları}\} = \{0.5/1.50, 0.7/1.60, 1/1.70, 0.7/1.80, 0.5/1.90\}$ olarak tanımlanmış olsun. Bu durumda $x=1.60$ m uzunluğundaki Yavuz Bey için $\underline{P} =$ “Yavuz Bey orta boyludur”, yani; $x \in \underline{A}$ bulanık önermesinin doğruluk değeri $T(\underline{P})=0.7$ olacaktır.

Özetle bulanık önermelerin, evrensel küme içerisindeki elemanlar için doğruluk değerleri, ilgili elemanın, bulanık önermeye karşılık getirilen bulanık kümeye üyelik derecesidir. Yani belirli bir $x \in X$ elemanı için $\underline{P}$ bulanık önermesinin doğruluk değeri $T(\underline{P})$, (7.1) ile elde edilir:

$$T(\underline{P}) = \mu_{\underline{A}}(x), 0 \leq \mu_{\underline{A}}(x) \leq 1 \quad (7.1)$$

$\underline{P}$ ve $\underline{Q}$ bulanık önermeler olmak üzere $\underline{P} \vee \underline{Q}$, $\underline{P} \wedge \underline{Q}$, $\neg \underline{P}$, $\underline{P} \rightarrow \underline{Q}$ ifadelerinin doğruluk değeri şu biçimde hesaplanır [2]:

a) $\underline{P} \vee \underline{Q}$ ($\underline{P}$ veya $\underline{Q}$) :

$$T(\underline{P} \vee \underline{Q}) = \max(T(\underline{P}), T(\underline{Q})) \quad (7.2)$$

b) $\underline{P} \wedge \underline{Q}$ ($\underline{P}$ ve $\underline{Q}$) :

$$T(\underline{P} \wedge \underline{Q}) = \min(T(\underline{P}), T(\underline{Q})) \quad (7.3)$$

c) $\neg \underline{P}$ (değil $\underline{P}$) :

$$T(\neg \underline{P}) = 1 - T(\underline{P}) \quad (7.4)$$

d) $\underline{P} \rightarrow \underline{Q}$ ($\underline{P}$ gerektirir $\underline{Q}$) :

$$T(\underline{P} \rightarrow \underline{Q}) = T(\neg \underline{P} \vee \underline{Q}) = \max(T(\neg \underline{P}), T(\underline{Q})) \quad (7.5)$$

X ve Y farklı evrensel kümeler ve $\underline{A} \subseteq X$, $\underline{B} \subseteq Y$ olmak üzere $\underline{P}$: $x \in \underline{A}$ ve $\underline{Q}$: $y \in \underline{B}$ şeklinde tanımlanmış bulanık önermeler olsun.

Bu durumda, $\underline{P} \rightarrow \underline{Q}$ ifadesi, IF $x \in \underline{A}$ THEN $y \in \underline{B}$ kuralına karşılık gelir ve bu kuralın matrisi aşağıdaki biçimde elde edilir:

$$\underline{R} = (\underline{A} \times \underline{B}) \vee (\neg \underline{A} \times \underline{Y}) \quad (7.6a)$$

$$\mu_{\underline{R}}(x,y) = \max[(\mu_{\underline{A}}(x) \wedge \mu_{\underline{B}}(y)), (1 - \mu_{\underline{A}}(x))] \quad (7.6b)$$

IF $x \in \underline{A}$ THEN $y \in \underline{B}$ ELSE $y \in \underline{C}$ şeklindeki bir kural sözkonusu ise kural matrisi şu biçimde olacaktır:

$$\underline{R} = (\underline{A} \times \underline{B}) \vee (\neg \underline{A} \times \underline{C}) \quad (7.7a)$$

$$\mu_{\underline{R}}(x,y) = \max[(\mu_{\underline{A}}(x) \wedge \mu_{\underline{B}}(y)), (1 - \mu_{\underline{A}}(x)) \wedge \mu_{\underline{C}}(y)] \quad (7.7b)$$

Yaklaşım ile çıkarım yapabilmek için, kural matrisini elde ettiğimiz şu aşamada, IF $x \in \underline{A}$ THEN $y \in \underline{B}$ kuralının ve $\underline{A}'$ bulanık önermesinin elimizde olduğunu düşünelim. Bu durumda $\underline{A}'$ kullanılarak edilecek bir $\underline{B}'$ önermesinin doğruluk değeri nasıl hesaplanır ? Bu sorunun yanıtı aşağıdadır:

$$\underline{B}' = \underline{A}' \bullet \underline{R} \quad (7.8a)$$

$$\mu_{\underline{B}'}(y) = \vee_{x \in \underline{A}'} (\mu_{\underline{A}'}(x) \wedge \mu_{\underline{R}}(x,y)) \quad (7.8b)$$

$\underline{A}' \bullet \underline{R}$ işlemi, en büyük-en küçük kompozisyonu (max-min composition) olarak adlandırılır ve bu işlem sonunda $\underline{B}'$ bulanık önermesinin doğruluk değeri elde edilir. Bu anlatılanları bir örnekle açıklamaya çalışalım:

Bir üretici firma olalım ve ürettiğimiz malları hangi büyüklükteki pazarlarda satmamız gerektiği sorusuna yanıt arayalım.

$$X=\{\text{malların özgünlüğü}\} = \{1,2,3,4\}$$

$$Y=\{\text{pazar büyüklükleri}\} = \{1,2,3,4,5,6\}$$

$$\underline{A}=\{\text{orta özgün mallar}\} = \{0.6/2,1/3,0.2/4\}$$

$$\underline{B}=\{\text{orta pazar büyüklüğü}\} = \{0.4/2,1/3,0.8/4,0.3/5\}$$

IF $\underline{A}$ THEN $\underline{B}$ olarak tanımlansın. $\underline{R} = (\underline{A} \times \underline{B}) \vee (\neg \underline{A} \times \underline{Y})$ şu biçimde elde edilir:

$$\underline{R} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 0.4 & 0.4 & 0.6 & 0.6 & 0.4 & 0.4 \\ 0 & 0.4 & 1 & 0.8 & 0.3 & 0 \\ 0.8 & 0.8 & 0.8 & 0.8 & 0.8 & 0.8 \end{bmatrix} \quad (7.9)$$

en büyük-en küçük kompozisyonunun yanısıra, bulanık mantıkta adı geçen diğer bazı kompozisyon işlemleri şunlardır [2]:

a) en büyük – çarpım (max-product) :

$$\mu_{\underline{B}'}(y) = \vee_{x \in \underline{A}'} (\mu_{\underline{A}'}(x) \cdot \mu_{\underline{R}}(x,y)) \quad (7.10)$$

b) en küçük - en büyük (min-max) :

$$\mu_{\underline{B}'}(y) = \wedge_{x \in \underline{A}'} (\mu_{\underline{A}'}(x) \vee \mu_{\underline{R}}(x,y)) \quad (7.11)$$

c) en büyük - en büyük (max-max) :

$$\mu_{\underline{B}'}(y) = \vee_{x \in \underline{A}'} (\mu_{\underline{A}'}(x) \vee \mu_{\underline{R}}(x,y)) \quad (7.12)$$

d) en küçük - en küçük (min-min) :

$$\mu_{\underline{B}'}(y) = \wedge_{x \in \underline{A}'} (\mu_{\underline{A}'}(x) \wedge \mu_{\underline{R}}(x,y)) \quad (7.13)$$

e) en büyük – ortalama (max-average):

$$\mu_{\underline{B}'}(y) = \vee_{x \in \underline{A}'} (\mu_{\underline{A}'}(x) + \mu_{\underline{R}}(x,y)) / 2 \quad (7.14)$$

Bu aşamada değinilmesi gereken önemli bir nokta; ilerideki bölümlerde detaylı olarak incelenecek olan Ağırlıklı Bulanık Çıkarım Algoritması'nın, tanıtılan bu kompozisyon işlemlerinden farklılaştığı noktadır. Bu kompozisyon işlemlerinin hiçbirinde, kuralın koşul kısmındaki bulanık önermenin ağırlığı (weight-çıkarım sürecindeki önemi) ve kuralın belirginlik çarpanı (certainty factor) dikkate alınmamaktadır. Ağırlıklı Bulanık Çıkarım Algoritması, kuralların koşul kısmında yeralan önermelerin ağırlıklarını ve kuralların belirginlik çarpanlarının dikkate alarak çıkarım gerçekleştirir. [4]

7.2. Dilsel Pekiştiriciler (linguistic hedges)

Bulanık mantığın uygulama alanlarından biri, doğal dildeki belirsizliğin niceleştirilmesidir. Herhangi bir kişi, doğal dilde söylenmiş bazı ifadeleri bulanık kümeler aracılığıyla matematiksel forma dönüştürebilir. Bu işlem sırasında matematiksel fonksiyonlar kullanılır ve doğal dilde ifade edilmek istenenlerin anlamlarının korunmasına dikkat edilir.

İnsanlar, kullandıkları doğal dile olan alışkanlıklarından dolayı, bu dönüştürme işleminde hiç zorlanmasalar da, doğal dildeki belirsizlik bu işlemi oldukça zorlaştırır. Ancak, bu işlem başarıyla tamamlandıktan sonra bulanık kümeler yardımıyla matematiksel olarak temsil edilen bilgiden başka bilgi parçacıkları elde edilebilir. Bu işlem dilsel pekiştiriciler yardımıyla yapılır. “Çiğdem uzundur” önermesini ele alalım. Bu önerme bulanık bilgi içermektedir çünkü “uzun” sözcüğü kişiden kişiye farklı anlam taşır [2].

Örneğin; $A = \text{“uzun”}$ ve $\mu_{uzun}(\text{Çiğdem}) = .95$ olarak tanımlansın. Dilsel pekiştiriciler (çok, çok çok vb.) değişkenler kullanarak Çiğdem'in sırayla “çok uzun”, “çok çok uzun” vb. bulanık kümelerine üyelik derecesini hesaplayabiliriz. Bu,

$$\mu_{\text{çok uzun}}(\text{Çiğdem}) = \mu_{uzun}^2(\text{Çiğdem}) = (.95)^2 = .9025 \quad (7.15)$$

$$\mu_{\text{çok çok uzun}}(\text{Çiğdem}) = \mu_{uzun}^4(\text{Çiğdem}) = (.95)^4 = .8145 \quad (7.16)$$

şeklinde elde edilir.

Dilsel pekiştiriciler yardımıyla, evrensel kümedeki elemanların bazı bulanık kümelere üyelik derecelerini kullanarak başka bulanık kümelere üyelik dereceleri hesaplanabilir. Dilsel pekiştiricilere örnek olarak [2];

a) Çok (A) :

$$\underline{A}^2 = \sum \mu_{\underline{A}}^2(x_i)/x_i \quad (7.17)$$

b) Çok Çok (A) :

$$\underline{A}^4 = \sum \mu_{\underline{A}}^4(x_i)/x_i \quad (7.18)$$

c) Yaklaşık (A) :

$$\underline{A}^{0.5} = \sum \mu_{\underline{A}}^{0.5}(x_i)/x_i \quad (7.19)$$

verilebilir.



8. AĞIRLIKLIL BULANIK ÇIKARIM ALGORİTMASI VE GELİŞTİRİLEN UZMAN SİSTEM

Kural tabanlı sistemler alanındaki en önemli araştırma konularından biri; bu sistemlerin, belirsizlik altında çıkarım yapabilmesidir. Günlük yaşantımızda kullandığımız bilginin belirsizlik içerdiğini gözönünde bulundurursak, kural tabanlı sistemler için belirsizlik altında çıkarım yapma yeteneğinin gerekli olduğunu görürüz.

Bu yüksek lisans tez çalışmasında da; belirsizlik içeren bilgiyi kullanarak çıkarım yapabilen, genel amaçlı bir bulanık uzman sistem geliştirilmesi hedeflenmiş ve geliştirme sürecinde Shyi Ming Chen tarafından ortaya atılan Ağırlıklı Bulanık Çıkarım Algoritması (Weighted Fuzzy Reasoning Algorithm For Rule Based Systems) kullanılmıştır. Bu algorithmada, kuralların koşul kısmında yeralan önermelerin doğruluk değerleri (truth value) ve ağırlıkları (weight-çıkarım sürecinde sahip oldukları önem), kuralların belirginlik çarpanları (certainty factor-uzman kişinin kuraldan emin olma derecesi), yamuk bulanık sayılar (trapezoidal fuzzy number) aracılığıyla temsil edilir [4].

Ağırlıklı Bulanık Çıkarım Algoritması'nda, kuralların koşul kısımlarındaki önermelerin birbirinden farklı olan ağırlıklarının, çıkarım sürecine katılması esneklik sağlamaktadır. Böylelikle; bilgedeki belirsizliğin, çıkarım sürecinde daha geniş çerçevede kullanımına olanak sağlanır. Örnek olarak şu kuralı ele alalım;

“Eğer nem oranı yüksekse ve barometrik basınç düşükse ve şimşek çakıyorsa yağmur yağacaktır”.

Kuralın koşul kısmında yeralan ilk iki önermenin daha bilimsel bir yaklaşım içerdiği ve çıkarımda, üçüncü önermeye kıyasla daha fazla önem taşıdığı düşünülebilir. Bu önem farkının vurgulanması gereklidir [4].

Bu bölümde, öncelikle Ağırlıklı Bulanık Çıkarım Algoritması'nda bilginin (gerçeklerin ve kuralların) nasıl temsil edildiği, çıkarımın nasıl yapıldığı üzerinde

$$A / B = (a_1, b_1, c_1, d_1) / (a_2, b_2, c_2, d_2) = (a_1 / a_2, b_1 / b_2, c_1 / c_2, d_1 / d_2) \quad (8.1c)$$

Belirsizliğin temsil edilmesinde kullanılan dilsel terimler ve bu terimlere karşılık gelen yamuk bulanık sayılar aşağıda listelenmiştir:

unknown	(0,0,0,0)
absolutely-false	(0,0,0,0)
very-low	(0,0,0.02,0.07)
low	(0.04,0.1,0.18,0.23)
medium-low	(0.17,0.22,0.36,0.42)
medium	(0.32,0.41,0.58,0.65)
medium-high	(0.58,0.63,0.80,0.86)
high	(0.72,0.78,0.92,0.97)
very-high	(0.975,0.98,1,1)
absolutely-high	(1,1,1,1)

Yukarıdaki dilsel terimler ile ifade edilen belirsizlik, bu dilsel terimlere karşılık gelen 4 elemanlı dizilişlerle yer değiştirdikten sonra çıkarım sürecine dahil edilir.

8.1.2. Ağırlıklı Bulanık Çıkarım Algoritması'nda Kullanılan Kural Yapısı

Bilgi tabanında yeralan tüm kuralların kümesini $R=\{R_1, R_2, R_3, \dots, R_r\}$ ile gösterelim. Genel kural biçimi;

$$R_i: \text{IF } C_j \text{ THEN } C_k \text{ (CF=f}_{kj}\text{)} \quad (8.2)$$

şeklindedir. Burada C_j ; doğruluk değeri bir bulanık sayıyla gösterilen önermeyi, C_k ; doğruluk değeri hesaplanacak önermeyi, f_{kj} ; kuralın belirginlik çarpanını göstermektedir. C_j önermesinin doğruluk değerini t_j ile gösterirsek, C_k önermesinin doğruluk değeri t_k ;

$$t_k = t_j \times f_{kj} \quad (8.3)$$

denklemleriyle hesaplanır.

Eğer sonuç kısmında C_k önermesinin olduğu iki kural varsa ve C_a , C_b önermelerinin doğruluk değerleri sırasıyla t_a ve t_b ile gösterilirse;

$$R_1: \text{IF } C_a \text{ THEN } C_k \text{ (CF=f}_{ka}\text{)}$$

$$R_2: \text{IF } C_b \text{ THEN } C_k \text{ (CF=f}_{kb}\text{)}$$

$$t_k = (t_a \times f_{ka}) \vee (t_b \times f_{kb}) \quad (8.4)$$

denklemleriyle hesaplanır [4].

Bazı kurallarda, C_j önermesi birden fazla önermenin AND ve/veya OR bağlayıcılarıyla biraraya getirilmesinden oluşabilir. Böyle bir durumda, C_j , bileşik önerme (compound statement) olarak nitelendirilir ve yukarıda sözü edilen hesaplamaların yapılabilmesi için t_j değerinin, C_j 'yi oluşturan alt önermelerin doğruluk değerlerinden hesaplanması gerekir. Bu hesaplama, kuralda AND ve/veya OR bağlayıcılarının kullanılmasına bağlı olarak iki şekilde yapılır [4]:

$$a) \text{ IF } C_{j1} \text{ AND } C_{j2} \text{ AND} \dots \text{ AND } C_{jn} \text{ THEN } C_k \text{ (CF= } f_{kj} \text{):}$$

$C_{j1}, C_{j2}, \dots, C_{jn}$ önermelerinin doğruluk değerleri sırasıyla $t_{j1}, t_{j2}, \dots, t_{jn}$ ve ağırlıkları sırasıyla $w_{j1}, w_{j2}, \dots, w_{jn}$ olsun. $C_j = C_{j1} \text{ AND } C_{j2} \text{ AND} \dots \text{ AND } C_{jn}$ önermesinin doğruluk değeri

$$t_j = (t_{j1} \times w_{j1} + t_{j2} \times w_{j2} + \dots + t_{jn} \times w_{jn}) / (w_{j1} + w_{j2} + \dots + w_{jn}) \quad (8.5)$$

denklemleriyle hesaplanır.

$$b) \text{ IF } C_{j1} \text{ OR } C_{j2} \text{ OR} \dots \text{ OR } C_{jn} \text{ THEN } C_k \text{ (CF= } f_{kj} \text{):}$$

$C_{j1}, C_{j2}, \dots, C_{jn}$ önermelerinin doğruluk değerleri sırasıyla $t_{j1}, t_{j2}, \dots, t_{jn}$ ve ağırlıkları sırasıyla $w_{j1}, w_{j2}, \dots, w_{jn}$ olsun. Bu durumda kural, ayırık kurallara indirgenebilir.

$$\text{IF } C_{j1} \text{ THEN } C_k \text{ (CF=f}_{kj}\text{)} \dots$$

$$\dots \text{IF } C_{jn} \text{ THEN } C_k \text{ (CF=f}_{kj}\text{)}$$

$C_j = C_{j1} \text{ OR } C_{j2} \text{ OR} \dots \text{ OR } C_{jn}$ önermesinin doğruluk değeri (8.6) ile hesaplanır.

$$t_j = [(t_{j1} \times w_{j1}) / (w_{j1} + w_{j2} + \dots + w_{jn})] \vee$$

$$[(t_{j2} \times w_{j2}) / (w_{j1} + w_{j2} + \dots + w_{jn})] \vee$$

...

$$[(t_{jn} \times w_{jn}) / (w_{j1} + w_{j2} + \dots + w_{jn})] \quad (8.6)$$

8.1.3. Ağırlıklı Bulanık Çıkarım Algoritmasının'da Kullanılan Kural Matrisi Ve Doğruluk Değer Matrisi (rule matrix and truth value matrix)

Bilgi tabanında yer alan bir kuralın genel biçimini (8.2) denkleminde vermiş ve C_k önermesinin doğruluk değeri t_k 'nın, (8.3) denklemiyle hesaplanacağını söylemiştik.

Kural tabanındaki kuralların sayısında herhangi bir sınırlama olmadığını düşünürsek, (8.3) denkleminde belirtilen işlemin, herbir kuralın sonuç kısmındaki önermenin doğruluk değerinin hesaplanması için tekrarlanması gerekir. Bu işlem için t_k ve t_j doğruluk değerlerinin, f_{kj} kullanılarak bir şekilde ilişkilendirilmesi zorunludur. Bu ilişkilendirme işlemi F kural matrisi (rule matrix) kullanılarak yapılır.

Tüm kuralların koşul veya sonuç kısımlarında ayrık, toplam m adet önermenin olduğunu varsayalım. F matrisinin boyutu ($\dim(F)$) $m \times m$ olur ve F'in elemanları ($F(k,j)$), (8.2) denklemiyle $k,j = 0, 1, \dots, m$ için (8.7) denklemiyle belirlenir [4].

$$F(k,j)=f_{kj} \quad (8.7)$$

Herhangi iki C_j ve C_k önermeleri arasında bir ilişki yoksa, $F(k,j)$ ="unknown", C_j ve C_j önermeleri arasında birebir aynı olmak durumu sözkonusu olduğundan, refleksifliği sağlamak için $k=0, 1, \dots, m$, $F(k,k)$ = "absolutely-high" olarak atanır.

(8.3) denkleminde belirtilen işlemin yapılabilmesi için gerekli olan diğer matris, T doğruluk değer matrisidir (truth value matrix). T matrisi, $m \times 1$ boyutludur ($\dim(T)= m \times 1$) ve elemanları, önermelerin doğruluk değerlerini temsil eden yamuk bulanık sayılardır [4]. $j = 0, 1, \dots, m$ için

$$T(j)=t_j \quad (8.8)$$

Çıkarım sürecinin başlangıcında C_j önermesinin doğruluk değeri bilinmiyor ise $T(j)$ ="unknown"=(0,0,0,0) değerini alır.

Bu anlatılanları bir örnekle açıklayalım. Kural tabanında;

R_1 : IF enflasyon yüksek THEN fiyat artışı yüksek (CF=high)

R_2 : IF fiyat artışı yüksek THEN alım gücü düşük (CF=very-high)

şeklinde iki kural yeralısın ve “enflasyon yüksek” önermesinin doğruluk değeri, kullanıcı tarafından “very-high” olarak girilsin. Bu durumda F ve T matrisleri şu biçimde elde edilir:

$$F = \begin{bmatrix} \text{absolute} & \text{true} & \text{unknown} & \text{unknown} \\ \text{high} & \text{absolute} & \text{true} & \text{unknown} \\ \text{unknown} & \text{very-high} & \text{absolute} & \text{true} \end{bmatrix} \quad (8.9a)$$

$$T = \begin{bmatrix} \text{very-high} \\ \text{unknown} \\ \text{unknown} \end{bmatrix} \quad (8.9b)$$

8.1.4. Ağırlıklı Bulanık Çıkarım Algoritması'nda Kullanılan Çıkarım Yöntemi (Weighted Fuzzy Reasoning Algorithm)

Ağırlıklı Bulanık Çıkarım Algoritması, adından da anlaşıldığı gibi, bilgisayar ortamında, ağırlıklı bulanık çıkarım yapmak üzere geliştirilmiş bir algoritmadır. Bu algoritma, çıkarım sürecinin başlangıcında girdi (input) olarak, bazı önermelerin doğruluk değerlerini yamuk bulanık sayılar şeklinde alır ve ağırlıklı bulanık çıkarım işlemini gerçekleştirerek, bazı başka önermelerin doğruluk değerlerine yamuk bulanık sayılar şeklinde ulaşır. Bu işlemdeki dikkate değer nokta; kuralların koşul kısmında yeralan önermelerin ağırlıklarının ayrı ayrı değerlendirilmesidir [4].

Kural tabanındaki kuralların, m adet basit önerme ($C_1, C_2, \dots, C_m$) içerdiğini ve bu önermelerin AND ve/veya OR bağlayıcıları ile bağlanarak k adet bileşik önermenin ($C_{m+1}, C_{m+2}, \dots, C_{m+k}$) oluşturulduğunu varsayalım. (8.7), (8.8) denklemleri uyarınca kural matrisi F ve doğruluk değeri matrisi T (8.10a),(8.10b) denklemleriyle elde edilir:

$$F = \begin{bmatrix} f_{11} & f_{12} & \dots & f_{1(m+k)} \\ f_{21} & f_{22} & \dots & f_{2(m+k)} \\ \vdots & \vdots & \dots & \vdots \\ f_{(m+k)1} & f_{(m+k)2} & \dots & f_{(m+k)(m+k)} \end{bmatrix} \quad (8.10a)$$

$$T = \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_{m+k} \end{bmatrix} \quad (8.10b)$$

T^* ; ara doğruluk değer matrisi olarak adlandırılır ve $T^* = F \otimes T$ işlemi ile (8.11) denklemi kullanılarak oluşturulur:

$$T^* = F \otimes T = \begin{bmatrix} (f_{11} \times t_1) \vee (f_{12} \times t_2) \vee \dots \vee (f_{1(m+k)} \times t_{m+k}) \\ (f_{21} \times t_1) \vee (f_{22} \times t_2) \vee \dots \vee (f_{2(m+k)} \times t_{m+k}) \\ \vdots \\ (f_{(m+k)1} \times t_1) \vee (f_{(m+k)2} \times t_2) \vee \dots \vee (f_{(m+k)(m+k)} \times t_{m+k}) \end{bmatrix} \quad (8.11)$$

Ağırlıklı Bulanık Çıkarım Algoritması, $T^* = F \otimes T$ tanımını kullanarak aşağıdaki adımlarla yürütülür [4]:

Adım 1: Önergelerin doğruluk değerleri, T doğruluk değer matrisine atanır. C_i önermesinin doğruluk değeri, çıkarımın başlangıcında bilinmiyor ise $T(i) = \text{"unknown"} = (0,0,0,0)$ değerini alır.

$n = m+k$, en son bileşik önermenin indisi olmak üzere, tüm bileşik önermelerin doğruluk değerleri hesaplanır.

Adım 2: Oluşturulan F kural matrisi kullanılarak $T^* = F \otimes T$ işlemi gerçekleştirilir.

$T^* \neq T$ ise çıkarım henüz tamamlanmamış demektir. Adım 3 gerçekleştirilir.

$T^* = T$ ise çıkarım tamamlanmış demektir. Adım 4 gerçekleştirilir.

Adım 3: Adım 2'deki $T^* = F \otimes T$ işlemiyle, bileşik önermeleri oluşturan basit önermelerin doğruluk değerleri değişmiş olabilir. Tüm bileşik önermelerin doğruluk değerleri, $T = T^*$ ataması yapıldıktan sonra (basit önermelerin en son doğruluk değerleri kullanılarak) tekrar hesaplanır. Adım 2 tekrar gerçekleştirilir.

Adım 4: Çıkarım süreci sona erer. C_i önermesinin doğruluk değeri $T[i]$ 'dir.

Açık olarak;

Adım 1:

for i=1 to m do

$T[i]=t_i$

end:

n=m+k : q=m+1:

while q ≤ n do

if $C_q = C_j \text{ AND } C_k \text{ AND } \dots \text{ AND } C_s$ then

$T[q] = (T[j] \times w_j + T[k] \times w_k + \dots + T[s] \times w_s) / (w_j + w_k + \dots + w_s)$

end:

if $C_q = C_j \text{ OR } C_k \text{ OR } \dots \text{ OR } C_s$ then

$T[q] = [(T[j] \times w_j) / (w_{j1} + w_{j2} + \dots + w_{jn})] \vee \dots \vee [(T[s] \times w_s) / (w_{s1} + w_{s2} + \dots + w_{sn})]$

end:

q=q+1

end:

Adım 2:

$T^* = F \otimes T$

if $T^* \neq T$ then goto Adım 3

else goto Adım 4

end:

Adım 3:

for i=1 to m do

$T[i] = T^*[i]$

end:

q=m+1:

while $q \leq n$ do

if $C_q = C_j$ AND C_k AND ... AND C_s then

$$T[q] = (T^*[j] \times w_j + T^*[k] \times w_k + \dots + T^*[s] \times w_s) / (w_j + w_k + \dots + w_s)$$

end:

if $C_q = C_j$ OR C_k OR ... OR C_s then

$$T[q] = [(T[j] \times w_j) / (w_{j1} + w_{j2} + \dots + w_{jn})] \vee \dots \vee [(T[s] \times w_s) / (w_{j1} + w_{j2} + \dots + w_{jn})]$$

end:

$q = q + 1$

end:

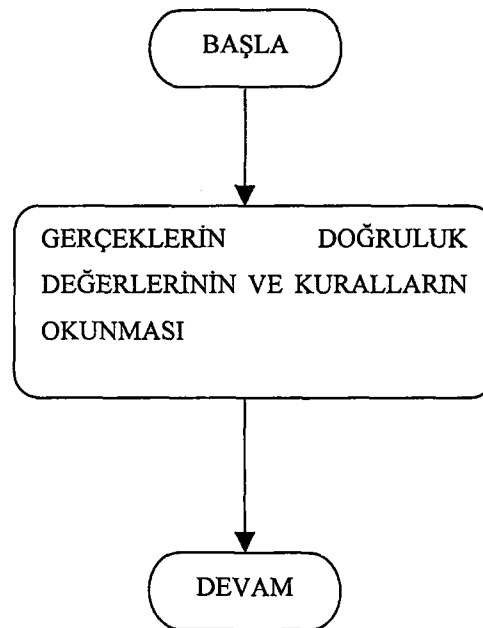
goto Adım 2:

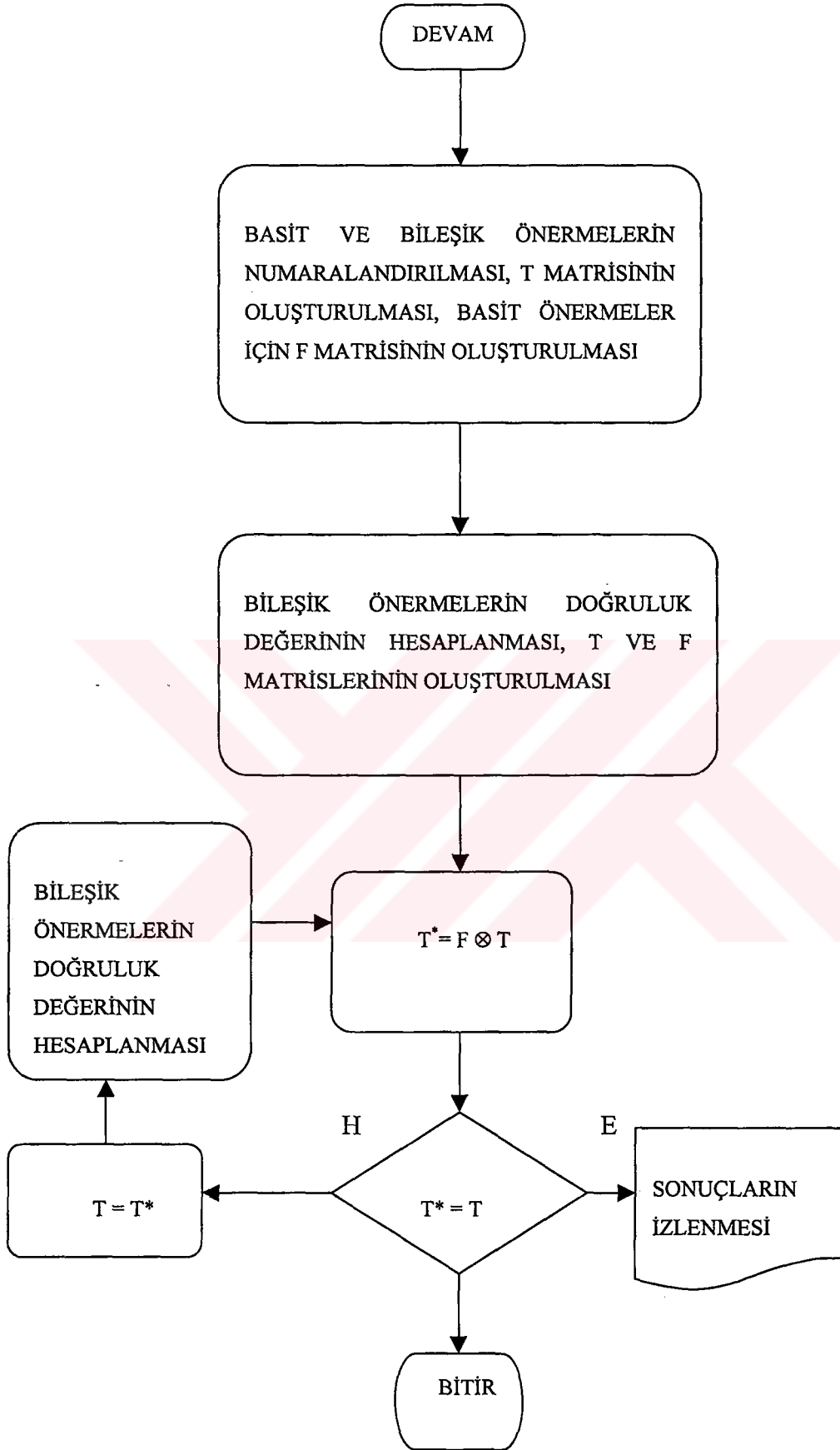
Adım 4:

C_i önermesinin doğruluk değeri $T[i]$ 'dir.

8.2. Geliştirilen Uzman Sistem

Bu başlıkta, Ağırlıklı Bulanık Çıkarım Algoritması başlığı altında anlatılanların, geliştirilen uzman sistemde hayata nasıl geçirildiği açıklanmıştır. Çıkarım sürecinin akış şeması Şekil 8.2'de görülmektedir.





Şekil 8.2 Çıkarım sürecinin akış şeması

8.2.1. Yamuk Bulanık Sayıların Uzman Sistemde Temsil Edilmesi

Çıkarım sürecinde kullanılan ve önceki bölümlerde tanıtılan yamuk bulanık sayıları bellekte tutmak için set_p göstericisi (pointer) ve set yapısı kullanılır. void dec_set(void) fonksiyonu, bellekten gerekli alanı tahsis ederek yamuk bulanık sayıları belleğe yerleştirir. set yapısı aşağıdaki biçimdedir:

```
struct set {  
  
    char set_name[20];    // yamuk bulanık sayının adı  
  
    float set_val[4];      // 4 elemanlı diziliş  
  
};
```

Bellekteki yerleşim Tablo 8.1’de gösterilmiştir.

Tablo 8.1 Yamuk bulanık sayıların bellekteki yerleşimi

Adres	set_name	set_val[0]	set_val[1]	set_val[2]	set_val[3]
set_p	Absolutely-false	0	0	0	0
set_p + 1	Very-low	0	0	0.02	0.07
set_p + 2	Low	0.04	0.1	0.18	0.23
set_p + 3	Medium-low	0.17	0.22	0.36	0.42
set_p + 4	Medium	0.32	0.41	0.58	0.65
set_p + 5	Medium-high	0.58	0.63	0.80	0.86
set_p + 6	High	0.72	0.78	0.92	0.97
set_p + 7	Very-high	0.975	0.98	1	1
set_p + 8	Absolutely-high	1	1	1	1
set_p + 9	Unknown	0	0	0	0

8.2.2. Önermelerin Doğruluk Değerlerinin Uzman Sistemde Temsil Edilmesi

Uzman sistemin çıkarım yapabilmesi için, önermelerin, uzman sistemin kullanıcısı tarafından sisteme girilen bulanık doğruluk değerlerinin, sayısal veriye dönüştürülmesi gerekir. Bu dönüştürme işleminin, Tablo 8.1’deki verinin kullanımıyla gerçekleştirilme şekli, Önermelerin Doğruluk Değerlerinin ve Ağırlıklarının Sayısallaştırılması başlığı altında incelenecektir.

Kullanıcı, önermelerin doğruluk değerlerini, doğal konuşma dilini kullanarak nasıl belirler ? Bu iki şekilde yapılabilir:

- (a) Doğruluk değerlerinin çıkarımın başlangıcında bir gerçek tabanından (factbase) alınması : Bu işlem; kullanıcıya, çıkarım sürecinin başlangıcında sorulan, bir gerçek tabanının kullanılıp kullanılmayacağı yönündeki soruya “evet” yanıtının verilmesi ile gerçekleştirilir. Kullanıcı, önermelerin bulanık doğruluk değerlerini içeren bir metin dosyasının (text file) adını sisteme girer. Önermelerin bulanık doğruluk değerleri, “deffact” anahtar sözcüğü kullanılarak yapılır:

$$\text{deffact (önerme) önermenin-bulanık-doğruluk-değeri} \quad (8.12)$$

Örneğin; “hastanın ateşi çok yüksek” önermesinin doğruluk değeri, kullanıcı tarafından “very-high” olarak değerlendiriliyorsa; bu gerçek, gerçek tabanında şu biçimde yer alır:

deffact (hastanın ateşi çok yüksek) very-high

Bu gerçek, uzman sistem tarafından void init_tr_value(const char *) fonksiyonu, tr_list_p göstericisi ve tr_list yapısı kullanılarak belleğe yerleştirilir. Bellekteki yerleşim Tablo 8.2’de görülmektedir. tr_list yapısı şu biçimdedir:

```
struct tr_list {  
    char st[100];          // önerme  
    char tr_value[20];     // önermenin bulanık doğruluk değeri  
};
```

Tablo 8.2 Bulanık doğruluk değerlerinin bellekteki yerleşimi

Adres	St	Tr_value
tr_list_p	Hastanın ateşi çok yüksek	Very-high

- (b) Doğruluk değerlerinin çıkarım sırasında kullanıcı tarafından uzman sistem ile etkileşimli olarak belirlenmesi : Bu işlem; kullanıcıya, çıkarım sürecinin başlangıcında sorulan, bir gerçek tabanının kullanılıp kullanılmayacağı yönündeki soruya, “hayır” yanıtının verilmesi ile

gerçekleştirilir. `void get_user_input (struct st_list *)` fonksiyonu, kullanıcının, uzman sistem ile etkileşimli olarak girdiği bulanık doğruluk değerlerini, Tablo 8.1'deki sayısal veriye dönüştürerek belleğe yerleştirir.

Her iki durumda da, girilen doğruluk değerlerinin Tablo 8.1'deki değerler arasından seçilmesi zorunludur. Doğruluk değerleri, çıkarım sürecinin ileri aşamalarında sayısallaştırılarak kullanılır.

8.2.3. Kullanılan Kural Tabanı ve Kural Tabanının Biçimi

Uzman sistem, çıkarım sürecinde kullanacağı kuralları, kullanıcı kişinin sisteme adını girdiği metin dosyasından (text file) alır. Bu dosya, uzman kişiden alınmış olan kuralları ve bu kuralların koşul kısımlarında yer alan önermelerin ağırlıklarını içerir. Kurallar ve kuralların koşul kısımlarındaki önermelerin ağırlıkları aşağıdaki biçimde tanımlanır:

(a) Kuralların koşul kısımlarındaki önermelerin ağırlıklarının tanımı :

Bu işlem, “defweight” anahtar sözcüğü kullanılarak kullanılarak yapılır:

defweight (önerme) önermenin-bulanık-ağırlık-değeri (8.13)

Örneğin; “hastanın ateşi çok yüksek” önermesinin ağırlığı, uzman kişi tarafından “medium” olarak değerlendiriliyorsa bu tanımlama, kural tabanında şu biçimde yeralır:

defweight (hastanın ateşi çok yüksek) medium

Burada önemli olan nokta, kuralların koşul kısmında yer alan herbir önermenin ağırlığının, “unknown” ve “absolutely-false” değerlerinden farklı olarak kural tabanına girilmiş olmasının zorunluluğudur. Aksi takdirde, uzman sistem bir uyarı mesajı vererek çıkarıma son verir. Ağırlıkların Tablo 8.1'deki değerler arasından (unknown ve absolutely-false hariç) seçilmesi zorunludur.

(b) Kuralların tanımı :

Bu işlem, “defrule”, “\$and”, “\$or”, “\$not”, “\$cf”, “\$conclude” anahtar sözcükleri kullanılarak şu şekilde yapılır.

$$\text{defrule önerme-dizisi } \$\text{conclude sonuç } \$\text{cf}=\text{bul.bel.}\check{\text{car}}. \quad (8.14)$$

Bu anahtar sözcükler:

- i. \$and : “ve” anlamına gelir.
- ii. \$or : “veya” anlamına gelir.
- iii. \$not : “değil” anlamına gelir.
- iv. önerme-dizisi : \$and ve/veya \$or ve/veya ile birleştirilmiş önermeler
- v. \$conclude : doğruluk değeri hesaplanacak önermeyi gösterir.
- vi. \$cf : Bulanık belirlenlik çarpanı değeridir. Tablo 8.1’deki değerler arasından seçilmesi zorunludur.

Kurallarda “prefix” gösterim kullanılır. Yani, \$and ve \$or operatörleri kendilerinden sonra gelen ilk iki operand, \$not operatörü kendisinden sonra gelen ilk operand üzerinde işlem yapar. Örneğin; uzman kişi tarafından “hastanın ateşi çok yüksek ve hasta 15 yaşından küçük ise Apranax verilmemelidir (bel. çar. = very-high)” şeklinde verilmiş olan bir kural, kural tabanında şu biçimde yeralır:

defrule \$and (hastanın ateşi çok yüksek) (hasta 15 yaşından küçük)

\$conclude (Apranax verilmemeli) \$cf=very-high

\$and, \$or ve \$not anahtar sözcüklerinin kullanımıyla elde edilen bileşik önermelerin doğruluk değerlerinin hesaplanma yöntemi Bileşik Önermelerin Doğruluk Değerlerinin Hesaplanması başlığı altında detaylandırılacaktır.

8.2.4. Önermelerin Ağırlıklarının Uzman Sistemde Temsil Edilmesi

Uzman kişi tarafından yapılan ağırlık tanımları, uzman sistem tarafından void init_weight(char *) fonksiyonu, value_p göstericisi ve value yapısı kullanılarak belleğe yerleştirilir. 8.2.2 başlığında incelenenler doğrultusunda, uzman sistem, problemle ilgili gerçekleri bir gerçek tabanından okumuşsa, ağırlıkların belleğe yerleştirilme aşamasına geldiğinde, kuralların koşul kısmında yeralan önermelerin bazılarının doğruluk değerleri de biliniyor olabilir. void init_weight(char *) fonksiyonu, Tablo 8.2’de verilen bellek bölgesini tarayarak doğruluk değeri

belirlenmiş olan önermelerin doğruluk değerlerini bulur ve value yapısına yerleştirir. Böylelikle bu önermelerin, doğruluk değerlerinin ve ağırlıkların sayısallaştırılması aşamasında farklı bellek bölgelerine ulaşma ihtiyacı ortadan kaldırılır. Bellekteki yerleşim Tablo 8.3’de görülmektedir. value yapısı şu biçimdedir:

```
struct value {
    char st[100];          // önerme
    char tr_value[20];     // önermenin, tr_list yapısından bulunan
                          // doğruluk değeri
    char weight[20];       // önermenin, kural tabanından okunan
                          // ağırlığı
};
```

Tablo 8.3 Değerlerin bellekteki yerleşimi

Adres	St	Tr_value	Weight
Value_p	Hastanın ateşi çok yüksek	Very-high	Medium

8.2.5. Önermelerin Doğruluk Değerlerinin ve Ağırlıklarının Sayısallaştırılması

Şu aşamaya kadar ele alınan başlıklarda, önermelerin, doğal konuşma dilinde verilen doğruluk değerlerinin ve ağırlıklarının bilgisayar belleğine nasıl yerleştirildiğini ele aldık. Ancak, uzman sistemin matematiksel işlemler yapabilmesi için bu sözel ifadelerin sayısallaştırılması gereklidir. Sayısallaştırma işlemi char index_st(char *,int,char,int) fonksiyonu, st_list_p göstericisi ve st_list yapısı kullanılarak yapılır.

index_st fonksiyonu, kural tabanını baştan sona doğru tarayarak, kural satırları içerisinde yer alan herbir ayrık önermeyi sırasal olarak numaralandırır. Üretilen bu numaralar, Ağırlıklı Bulanık Çıkarım Algoritması’nda Kullanılan Kural Matrisi ve Doğruluk Değer Matrisi başlığında tanımlanan F kural matrisinin oluşturulmasında kullanılır. index_st fonksiyonu, kuralların koşul kısmında yer alan önermeler için Tablo 8.3’de gösterilen bellek bölgesini, kuralların sonuç kısımlarında

yeralan önermeler için Tablo 8.2’de gösterilen bellek bölgesini tarayarak bu önermelerin doğruluk değerlerini ve ağırlıklarını doğal dildeki ifadeler halinde elde eder. Bu değerlere karşılık gelen 4 elemanlı dizilişler (yamuk bulanık sayılar), struct set *get_fuzzy_value(char *) fonksiyonu kullanılarak Tablo 8.1’de gösterilen bellek bölgesinde bulunur ve st_list yapısına yerleştirilir.

st_list yapısı, Ağırlıklı Bulanık Çıkarım Algoritması’nda Kullanılan Kural Matrisi ve Doğruluk Değer Matrisi başlığında tanımlanan T doğruluk değer matrisine karşılık gelmektedir. st_list yapısı aşağıdaki biçimdedir.

```
struct st_list {
    int st_num;           // önerme numarası
    char st[200];         // önerme
    char unknown_flag;    // önermenin doğruluk değerinin bilinip
                        // bilinmediği
    float st_tr_value[4]; // önermenin doğruluk değeri
    float st_weight[4];   // önermenin ağırlığı
    float st_nx_tr_value[4]; // önermenin bir sonraki iterasyondaki
                        // doğruluk değeri
    float mtch_dgr;       // önermenin sözel değeri ile uyuma
                        // derecesi
    char result[20];      // önermenin sözel değeri
};
```

Şu örneği yeniden ele alacak olursak Tablo 8.4’deki durum elde edilir :

Gerçek tabanı : deffact (enflasyon yüksek) very-high

Kural tabanı :

defweight (enflasyon yüksek) high

defweight (fiyat artışı yüksek) high

defrule (enflasyon yüksek) \$conclude (fiyat artışı yüksek) \$cf=high

defrule (fiyat artışı yüksek) \$conclude (alım gücü düşük) \$cf=very-high

Tablo 8.4 Bellekteki sayısallaştırılmış değer matrisi

Adres	St_num	St	St_tr_value[i]	St_weight[i]
st_list_p	0	Enflasyon yüksek	(0.975,0.98,1,1)	(0.72,0.78, 0.92,0.97)
St_list_p + 1	1	Fiyat artışı yüksek	(0,0,0,0)	(0.72,0.78, 0.92,0.97)
St_list_p + 2	2	Alım gücü düşük	(0,0,0,0)	(0,0,0,0)

8.2.6. Bileşik Önermelerin Doğruluk Değerlerinin Hesaplanması

Şu aşamada, kuralları oluşturan ayrık önermelerin doğruluk değerlerini ve ağırlıklarını sayısallaştırmış durumdayız. Ancak, kural tabanının yapısının incelenmesi sırasında değinildiği gibi kural tabanı sadece ayrık, basit önermeleri değil, basit önermelerin \$and, \$or, \$not operatörleri ile birleştirilmesinden elde edilen bileşik önermeleri (compound statements) de içerir.

Bu bileşik önermeler, kuralların koşul kısımlarında yer aldıklarından, çıkarım sürecinin başlangıcında doğruluk değerlerinin hesaplanması gereklidir. Bu hesaplama işlemi void index_comp_st(char *), void dcmp_comp_st(char *,int,int), void eval_comp_st(int) fonksiyonları, dcmp_list_p göstericisi ve dcmp_list yapısı kullanılarak yapılır.

index_comp_st fonksiyonu, kural tabanını baştan sona doğru tarayarak, kural satırları içerisinde \$and, \$or ve \$not anahtar sözcüklerini arar. Bu anahtar sözcükler, bileşik önerme oluşturmakta kullanıldığından doğruluk değer hesaplamasının yapılacağını gösteren birer araç görevi görür. index_comp_st fonksiyonu her bir bileşik önermeyi oluşturup, sırasal olarak numaralandırarak st_list yapısına ekler. Böylelikle bileşik önermeler de Tablo 8.4’de gösterilen bellek bölgesine eklenir. Üretilen önerme numaraları, Ağırlıklı Bulanık Çıkarım Algoritması’nda Kullanılan Kural Matrisi ve Doğruluk Değer Matrisi başlığında tanımlanan F kural matrisinin oluşturulmasında kullanılır.

dcmp_comp_st fonksiyonu, oluşturulan bu bileşik önermeyi parametre olarak alıp, dcmp_list yapısını kullanarak ayrıştırma (decomposition) işlemini gerçekleştirir. Bu işlemin yapılmasındaki amaç bileşik önerme içerisindeki operatörlerin tanınması, operandların ayrıştırılması, bu operandların doğruluk değerlerinin ve ağırlıklarının

st_list yapısından alınması ve Ağırlıklı Bulanık Çıkarım Algoritması'nda Kullanılan Kural Yapısı başlığında tanımlanan doğruluk değeri hesaplama işlemlerinin gerçekleştirilmesidir. dcmp_list yapısı şu biçimdedir:

```
struct dcmp_list {
    char st_type;      // operand'ın tipi
    char st[100];      // operand
    char rezerv_flag;  // operand'ın daha önceki bir hesaplamada
                      // kullanılıp kullanılmadığı
    float st_tr_value[4]; // operand'ın doğruluk değeri
    float st_weight[4];  // operand'ın ağırlığı
};
```

dcmp_list yapısı içerisindeki st_type ve rezerv_flag alanları konusunda belirtilmesi gereken önemli noktalar şunlardır:

(a) st_type : dcmp_list yapısı içindeki elemanın türünü gösterir.

- 0 ; elemanın \$or anahtar sözcüğü
- 1 ; elemanın \$and anahtar sözcüğü
- 2 ; elemanın bir önerme
- 3 ; elemanın \$not anahtar sözcüğü

olduğu durumlara karşılık gelir.

(b) rezerv_flag : dcmp_list yapısı içindeki elemanın daha önceki bir hesaplamada kullanılıp kullanılmadığının gösterir.

- h ; elemanın hesaplamada kullanılmadığı
- e ; elemanın hesaplamada kullanıldığı

durumlara karşılık gelir.

dcmp_comp_st fonksiyonunun, dcmp_list yapısını oluşturmasından sonra eval_comp_st fonksiyonu devreye girip, özyineli (recursive) çalışarak bileşik önermenin doğruluk değerini hesaplar. Her işlemin sonucu, o işlemde kullanılan operatörün üzerinde toplanır. Bu operatör, bir sonraki işlemde operand olarak

yeralabilir. Bu hesaplama sırasında kullanılan algoritma ve algoritmada kullanılan değişkenler aşağıda tanımlanmıştır:

opr : operatör (\$and, \$or, \$not)
i : operatörün, dcmp_list yapısı içindeki indisi
opr_1 : operatörün üzerinde işlem yapacağı 1. eleman indisi
opr_2 : operatörün üzerinde işlem yapacağı 2. eleman indisi
rezerv_index : 2. elemanı bulmakta kullanılan geçici indis değeri
total_dcmp_list : dcmp_list yapısı içindeki toplam eleman sayısı

if opr = \$and or opr = \$or

opr_1 = i + 1;

opr_2 = i + 2;

if opr_1.st_type = '2' and opr_2.st_type = '2'

i.st_tr_value[i] = eval_comp_st(opr_1,opr_2)

end:

if opr_1.st_type = '2' and

(opr_2.st_type = '0' or opr_2.st_type = '1' or opr_2.st_type = '3')

i.st_tr_value[i] = eval_comp_st(opr_1,opr_2)

opr_2.rezerv_flag = 'e'

end:

if (opr_1.st_type = '0' or opr_1.st_type = '1' or opr_1.st_type = '3')

opr_1.rezerv_flag = 'e'

rezerv_index=opr_2;

while rezerv_index < total_dcmp_list do

if (dcmp_list_p+rezerv_index).rezerv_flag = 'h' and

((dcmp_list_p+rezerv_index).st_type = '0' or

(dcmp_list_p+rezerv_index).st_type = '1' or

(dcmp_list_p+rezerv_index).st_type = '3')

```

                                opr_2 = rezerv_index;
                                break;

                                end:

                                rezerv_index = rezerv_index + 1

                                end:

                                opr_2 .rezerv_flag = 'e'

                                i.st_tr_value[i] = eval_comp_st(opr_1,opr_2)

                                end:

else

                                opr_1 = i+1

                                i.st_tr_value[i] = eval_comp_st(opr_1,opr_2)

                                end:

```

Bu anlattıklarımızı, gerçek tabanı ve kural tabanının aşağıdaki biçimde tanımlandığı bir örnekle açıklayalım.

Gerçek tabanı :

deffact (a) very-high

deffact (b) high

deffact (c) medium

Kural tabanı :

defweight (a) medium

defweight (b) low

defweight (c) high

defrule \$and (a) \$or (b) (c) \$conclude (d) \$cf=high

Bu durumda, dcmp_comp_st fonksiyonun “\$and (a) \$or (b) (c)” bileşik önermesini ayrıştırması sonucunda bellekteki durum Tablo 8.5’deki gibi olacaktır. Tablodaki ok işaretleri, hangi operatörlerin, hangi operandlar ile işleme gireceğini ve

elde edilen değerlerin dcmp_list yapısı içerisinde hangi bellek bölgesine yerleştirildiğini göstermektedir.

Tablo 8.5 dcmp_list yapısındaki yerleşim ve işlem akışı

Adres	St_ type	St	St_tr_value	St_weight
dcmp_list_p	1	\$and	(0.5,0.56,0.66,0.38)	(1.08,1.29,1.68,1.865)
dcmp_list_p+1	2	↓A	(0.975,0.98,1,1)	(0.32,0.41,0.58,0.65)
dcmp_list_p+2	0	\$or	(0.30,0.36,0.48,0.53)	(0.76,0.88,1.1,1.2)
dcmp_list_p+3	2	↓B	(0.72,0.78,0.92,0.97)	(0.04,0.1,0.18,0.23)
dcmp_list_p+4	2	↓C	(0.32,0.41,0.58,0.65)	(0.72,0.78,0.92,0.97)

Sonuçta; “\$and (a) \$or (b) (c)” önermesinin doğruluk değeri (0.5,0.56,0.66,0.38) olarak elde edilir.

8.2.7. Kural Matrisinin Oluşturulması

Ağırlıklı Bulanık Çıkarım Algoritması’nda kullanılan genel kural biçimi (8.2) de verilmişti. Bu denklemden de anlaşılacağı üzere, kural matrisi $F (F[k,j]=f_{kj})$ ’in oluşturulabilmesi için kurallar içinde yeralan önermelerin ardışık olarak numaralandırılması gereklidir. Bu işlemin, basit önermeler için index_st, bileşik önermeler için index_comp_st fonksiyonları tarafından yapıldığını söylemiştik.

index_st ve index_comp_st fonksiyonları, kural tabanından bir kural okuyup, bu kural içindeki basit ve bileşik önermeleri numaralandırdıktan sonra “\$cf” anahtar sözcüğünü kullanarak o kuralın belirginlik çarpanını oluşturur. Bu aşamada, belirginlik çarpanı doğal dilde ifade edilmiş olup, struct set *get_fuzzy_value(char *) fonksiyonu kullanılarak Tablo 8.1’de gösterilen bellek bölgesinden, bu belirginlik çarpanına karşılık gelen 4 elemanlı dizilişi (yamuk bulanık sayı) bulunur.

Geliştirilen uzman sistemde, F kural matrisini bellekte tutmak için rule_mat_p göstericisi ve rule_mat yapısı kullanılır. Bu yapı aşağıdaki biçimdedir:

```
struct rule_mat
{
    int rule_l;          //sıra numarası
    int rule_c;          //sütun numarası
```

```
float rule_cf[4];      // fkj değerine karşılık gelen 4 elemanlı diziliş
};
```

F kural matrisinin köşegeni üzerindeki elemanları “absolutely-high=(1,1,1,1)”, birbirleriyle bir kural aracılığıyla ilişkilendirilmeyen önermelerin bulunduğu satır ve sütunlardaki elemanları “unknown=(0,0,0,0)” ‘dır. Bu değerler, bilgisayarın belleğinden tasarruf etmek amacıyla bellekte tutulmaz ancak çıkarım işlemi sırasında önermelerin numaraları dikkate alınarak hesaplamaya dahil edilir.

Aşağıdaki örneği yeniden ele alacak olursak Tablo 8.6’da verilen durum elde edilir:

```
defrule (enfasyon yüksek) $conclude (fiyat artışı yüksek) $cf=high
```

```
defrule (fiyat artışı yüksek) $conclude (alım gücü düşük) $cf=very-high
```

Tablo 8.6 Bellekteki F kural matrisi

Adres	rule_l	rule_c	Rule_cf[i]
Rule_mat_p	1	0	(0.72,0.78,0.92,0.97)
rule_mat_p +1	2	1	(0.975,0.98,1,1)

8.2.8. Çıkarım İşleminin Gerçekleştirilmesi

Uzman sistemde, Ağırlıklı Bulanık Çıkarım Algoritması’nın ikinci adımında gerçekleştirilen $T^* = F \otimes T$ işlemindeki F ve T matrislerine sırasıyla rule_mat_p ve st_list_p göstericileri ile belirlenen rule_mat ve st_list yapılarındaki bellek bölgelerinin karşılık geldiğini söylemiştik.

void inf_engine(char) fonksiyonu, bu bellek bölgelerindeki veriyi kullanarak (8.8) denklemiyle tanımlanan $F \otimes T$ işlemini gerçekleştirir ve önermelerin yeni doğruluk değerlerini st_list yapısı içindeki st_nx_tr_value[] alanlarına atar. Bu alan, Ağırlıklı Bulanık Çıkarım Algoritması’nda tanımlanan T^* ara değer matrisine karşılık gelmektedir.

inf_engine fonksiyonu, bu işlemi, st_list yapısındaki tüm önermelerin yineleme öncesindeki değerleri ve yineleme sonrasındaki değerleri birbirine eşit oluncaya kadar (yani $T^* = T$) özyineli biçimde tekrarlar. Bileşik önermelerin doğruluk değerleri, kendilerini oluşturan alt önermelerin doğruluk değerleri zamanla değiştiği için her özyinelemenin başlangıcında yeniden hesaplanır.

8.2.9. Çıkarım İşleminin Sonuçlarının Üretilmesi

$T^* = T$, bir başka deyişle önermelerin doğruluk değerlerinin artık bir değişime uğramama noktasına ulaşıldığında özyineli olarak tekrarlanan $F \otimes T$ işlemine son verilir.

Bu noktada, st_list yapısı içinde yeralan önermelerin doğruluk değerleri, 4 elemanlı dizilişler ($st_tr_value[i]$) halinde elde edilmiştir. Yapılması gereken; bu sayısal veriyi, uzman sistemin kullanıcısının yorumlayabileceği şekle dönüştürmektir.

Bu dönüştürme işlemi, st_list_p göstericisi ile belirlenen bellek bölgesinde yeralan ve “amaç nedir ?” sorusuna yanıt olarak verilen ifadeyi içeren her basit önermenin doğruluk değerinin, set_p göstericisi ile belirlenen ve Tablo 8.1’de gösterilen bellek bölgesindeki herbir değerle karşılaştırılmasıyla gerçekleştirilir. Bu önerme, Tablo 8.1’de verilen yamuk bulanık sayıların hangisi ile en fazla derecede uyuyorsa (matched), bu ifadenin dilsel doğruluk değeri ilgili yamuk bulanık sayıdır.

Karşılaştırma işlemi sırasında, iki yamuk bulanık sayının birbirine ne kadar benzediğini belirleyen Bart Kosko Ölçüsü kullanılır. A ve B iki yamuk bulanık sayı olmak üzere Bart Kosko Ölçüsü ($m(A,B)$) (8.11e) denklemiyle tanımlanır:

$$A = (a_1, b_1, c_1, d_1)$$

$$B = (a_2, b_2, c_2, d_2)$$

$$A \cap B = (\min(a_1, a_2), \min(b_1, b_2), \min(c_1, c_2), \min(d_1, d_2)) = (a_3, b_3, c_3, d_3) \quad (8.15a)$$

$$C(A) = a_1 + b_1 + c_1 + d_1 \quad (8.15b)$$

$$C(B) = a_2 + b_2 + c_2 + d_2 \quad (8.15c)$$

$$C(A \cap B) = a_3 + b_3 + c_3 + d_3 \quad (8.15d)$$

$$m(A,B) = C(A \cap B) / \max[C(A), C(B)] \quad (8.15e)$$

9. UYGULAMA

Geliştirilen uzman sistemin, EK A ve EK B’de verilen örnek uygulamalarında, kural tabanı olarak, hastalıkların teşhisinde görev alacak kuralları içeren hastalik.txt dosyası kullanılmıştır. Bu dosya; Kullanılan Kural Tabanı ve Kural Tabanının Biçimi başlığı altında ele alınanlar doğrultusunda oluşturulmuştur. hastalik.txt dosyası aşağıda verilmiştir:

defweight (serum bilirubin duzeyi 2.5mg/dl uzerinde) very-high

defweight (deri ve mukozada sari renk var) very-high

defweight (karaciger buyumesi var) very-high

defweight (transaminaz duzeyi yuksek) very-high

defweight (bulanti ve yorgunluk var) medium-low

defweight (hastalik hepatit olabilir) very-high

defweight (kanda hepatit virusu var) absolutely-high

defweight (hastalik hepatit) absolutely-high

defweight (27nm buyuklukte RNA virusu var) absolutely-high

defweight (42nm den buyuk DNA virusu var) absolutely-high

defweight (kanda HIV antikoru var) absolutely-high

defweight (immun yetmezligi var) high

defweight (pneumocytis carinii var) high

defweight (kaposi sarkomu var) very-high

defweight (plazmadaki glukoz asiri yuksek) high

defweight (pankreas insulin salgisi az) high

defweight (polifaji,polidipsi,poliuri var) medium

defweight (hastalik diabet olabilir) high

defweight (insulin sekresyonu az) high

defweight (insulin verilmediginde ketoasitoz gelisiyor) very-high

defweight (insulin verilmediginde ketoasitoz gelismiyor) very-high

defweight (ates ve kuru oksuruk var) medium

defweight (tuberkulin deri testi pozitif) very-high

defweight (gogus radyografisinde lobuler infiltrasyon var) high

defweight (gece terlemesi,urperme,ates var) medium-low

defweight (hemoptizi var) high

defweight (kandaki lokosit sayisinda artis var) medium

defweight (kemik iligi infiltrasyonu var) high

defweight (trombositopenik kanama,dalak,karaciger,lenf nodullerinde buyume ve enfeksiyon var) absolutely-high

defweight (karin agrisi var) low

defweight (gastrointestinal bozukluklar var) low

defweight (psikolojik bozukluklar var) low

defweight (basdonmesi var) medium

defweight (alopesi var) medium-high

defweight (disetinde siyah cizgiler var) very-high

defweight (irit gelismesi sonucu korluk var) high

defweight (agizda tekrarlayan aftlar var) low

defweight (deride pyoderma,supkutenoz noduller var) medium

defweight (artralji var) low

defweight (trombo flebit var) low

defweight (artrit var) medium

defweight (uretrit var) medium

defweight (konjuktivit var) medium

defweight (iridosiklit var) medium

defweight (yorgunluk var) low

defweight (kirginlik,keyifsizlik var) low

defweight (dusuk ates var) low

defweight (fotosensitivite ile birlikte yuzde leke olusumu var) very-high

defweight (Reynould Fenomeni var) very-high

defweight (eklem agrisi var) medium

defweight (romatoid artrit var) medium

defweight (keratokonjunktivitis sica var) high

defweight (xerostomi var) low

defweight (bronslarda kuruluk var) medium

defweight (trakede kuruluk var) medium

defrule 00001

\$and (serum bilirubin duzeyi 2.5mg/dl uzerinde) \$and (deri ve mukozada sari renk var) \$and (karaciger buyumesi var) \$and (transaminaz duzeyi yuksek)(bulanti ve yorgunluk var) \$conclude (hastalik hepatit olabilir) \$cf=medium

defrule 00002

\$and (hastalik hepatit olabilir) (kanda hepatit virusu var) \$conclude (hastalik hepatit) \$cf=absolutely-high

defrule 00003

\$and (hastalik hepatit) (27nm buyuklukte RNA virusu var) \$conclude (hastalik hepatit A) \$cf=absolutely-high

defrule 00004

\$and (hastalik hepatit) (42nm den buyuk DNA virusu var) \$conclude (hastalik hepatit B) \$cf=absolutely-high

defrule 00005

\$and (kanda HIV antikorü var) \$and (immun yetmezligi var) \$or (pneumocytis carinii var) (kaposi sarkomu var) \$conclude (hastalik AIDS) \$cf=absolutely-high

defrule 00006

\$and (plazmadaki glukoz asiri yuksek) \$and (pankreas insulin salgisi az)(polifaji,polidipsi,poliuri var) \$conclude (hastalik diabet olabilir) \$cf=high

defrule 00007

\$and (hastalik diabet olabilir) \$and (insulin sekresyonu az)(insulin verilmediginde ketoasitoz gelisiyor) \$conclude (hastalik diabet Tip 1)\$cf=very-high

defrule 00008

\$and (hastalik diabet olabilir)(insulin verilmediginde ketoasitoz gelismiyor) \$conclude (hastalik diabet Tip 2) \$cf=very-high

defrule 00009

\$and (ates ve kuru oksuruk var) \$and (tuberkulin deri testi pozitif) \$and (gogus radyografisinde lobuler infiltrasyon var) \$and (gece terlemesi,urperme,ates var)(hemoptizi var) \$conclude (hastalik tuberkuloz) \$cf=absolutely-high

defrule 00010

\$and (kandaki lokosit sayisinda artis var) \$and (kemik iligi infiltrasyonu var)(trombositopenik kanama,dalak,karaciger,lenf nodullerinde buyume ve enfeksiyon var) \$conclude (hastalik losemi) \$cf=absolutely-high

defrule 00011

\$and (karin agrisi var) \$and (gastrointestinal bozukluklar var) \$and (psikolojik bozukluklar var) \$and (basdonmesi var) \$and (alopesi var)(disetinde siyah cizgiler var) \$conclude (hastalik kursun zehirlenmesi) \$cf=absolutely-high

defrule 00012

\$and (irit gelismesi sonucu korluk var) \$and (agizda tekrarlayan aftlar var)
\$and (deride pyoderma,supkutenoz noduller var) \$and (artralji var)(trombo
filebit var) \$conclude (hastalik Behcet Sendromu) \$cf=absolutely-high

defrule 00013

\$and \$and (artrit var)(uretrit var) \$or (konjuktivit var)(iridosiklit var)
\$conclude (hastalik Reiter Sendromu) \$cf=absolutely-high

defrule 00014

\$and (yorgunluk var) \$and (kirginlik,keyifsizlik var) \$and (dusuk ates var)
\$and (fotosensitivite ile birlikte yuzde leke olusumu var) \$and (Reynould
Fenomeni var)(eklem agrisi var) \$conclude (hastalik sistemik lupus
eritamosus) \$cf=absolutely-high

defrule 00015

\$and \$and (romatoid artrit var) \$and (keratokonjuktivitis sica var)(xerostomi
var) \$or (bronslarda kuruluk var)(trakede kuruluk var) \$conclude (hastalik
Sjorgen Sendromu) \$cf=absolutely-high

10. SONUÇLAR VE ÖNERİLER

Bu yüksek lisans tez çalışmasının konusu olan, bir bulanık uzman sistem geliştirme sürecinde, Shyi Ming Chen tarafından ortaya atılan Ağırlıklı Bulanık Çıkarım Algoritması (Weighted Fuzzy Reasoning Algorithm For Rule Based Systems) kullanılmıştır. Bu algoritma, uzman sistemin kural tabanındaki kuralların koşul kısmında yeralan önermelerin doğruluk değerlerinin (truth value) ve ağırlıklarının (weight-çıkarım sürecinde sahip oldukları önem), kuralların belirginlik çarpanlarının (certainty factor-uzman kişinin kuraldan emin olma derecesi), bazı dilsel ifadeler (low, medum, very-high vb.) aracılığıyla ifade edilmesine olanak sağlar. İnsanoğlunun, günlük yaşantısında kullandığı en önde gelen bilgi paylaşım aracı doğal konuşma dili olduğundan, bilginin bu tür dilsel ifadeler kullanılarak ifade edilmesine olanak sağlanması, kuralları oluşturan uzman kişi ve gerçekleri belirleyen uzman sistem kullanıcısı için büyük kolaylık sağlamaktadır.

Eklerde yeralan uygulamalarda İstanbul Üniversitesi Diş Hekimliği Fakültesi 5.sınıf öğrencisi Sn. Onur Çağlar tarafından hazırlanan bilgi tabanı kullanılmış ve Sn.Onur Çağlar ile uzman sistemin doğru, güvenilir sonuçlar ürettiği izlenmiştir.

Bununla birlikte uzman sistemin geliştirme sürecinde bazı zorluklarla karşılaşmıştır.

Bunlardan ilki, program yazımında kullanılan C Programlama Dili'nin bazı yetersizlikleridir. C, kullanımına azami derecede olanak sağladığı göstericiler (pointer) ile dinamik bellek yönetiminde büyük kolaylıklar sağlarken, bir liste işleme dili (list processing language) olmamasından dolayı zorluklara neden olmaktadır. Bu nedenle, kural tabanında yeralan “(A) AND (B) AND (C)” gibi bileşik önermelerin (compound statement) doğruluk değerlerinin hesaplamaları kolaylıkla yapılamamış, bu işlem için karmaşık fonksiyonlar (dcmp_comp_st(),eval_comp_st()) geliştirmek zorunda kalınmıştır. Benzer bir çalışmanın, LISP gibi; listeleri işleyebilen bir programlama dili ile yapılması geliştirme sürecini kısaltacaktır.

İkinci zorluk; Ağırlıklı Bulanık Çıkarım Algoritması'nda yer alan kural matrisi F'in bellekte temsil edilmesinde yaşanmıştır. Kural tabanında toplam m adet (basit veya bileşik) önerme olduğunu varsayacak olursak F matrisinin boyutu m x m olacaktır. Kısacası; çok büyük bilgi tabanları ile çalışıldığında F matrisinin boyu çok büyümektedir. Günümüzde kullanılan kişisel bilgisayarlar büyük bellek miktarlarına sahip olsalar da; bu, sınırlı belleğe sahip bilgisayarlar için bir problem olabilir. Bu problem şu şekilde aşılmıştır: F kural matrisinin köşegeni üzerindeki elemanları “absolutely-high=(1,1,1,1)”, birbirleriyle bir kural aracılığıyla ilişkilendirilmeyen önermelerin bulunduğu satır ve sütunlardaki elemanları “unknown=(0,0,0,0)” ‘dır. Bu değerler, bilgisayarın belleğinden tasarruf etmek amacıyla bellekte tutulmaz ancak çıkarım işlemi sırasında önermelerin numaraları dikkate alınarak hesaplamaya dahil edilir.

Geliştirilen uzman sistem, tek bir uzmandan alınan kuralları başarıyla kullanarak sonuca varabilmektedir. İlerideki aşamalarda, birden fazla uzmandan alınmış bilgi kullanılarak üretilen farklı sonuçların birleştirilmesi (fusion) sözkonusu olabilir.

KAYNAKLAR

- [1] **Parsaye, K. and Chignel, M.**, 1988. Expert Systems For Experts, John Wiley & Sons, Inc.
- [2] **Ross, T.J.**,1994. Fuzzy Logic With Engineering Applications, McGraw Hill Book Company.
- [3] **Klir, G. J. and Folger, T. A.**, 1988. Fuzzy Sets, Uncertainty And Information, USA
- [4] **Chen, S. M.**, 1996. A Fuzzy Reasoning Approach For Rule-Based Systems Based on Fuzzy Logic, IEEE Transactions On Systems, Man And Cybernetics Part B: Cybernetics. Vol. 26, No. 5

EK A

Bu uygulamada, rahatsızlıkları olan bir kişi ile ilgili gerçekler (belirtiler), “belirti1.txt” dosyasında yer almaktadır. Çıkarıma başlamadan önce, belirti1.txt dosyasında yer alan gerçeklere ve bu gerçeklerin bulanık doğruluk değerlerine göz atmak istersek:

```
deffact (serum bilirubin duzeyi 2.5mg/dl uzerinde) absolutely-high
deffact (deri ve mukozada sari renk var) high
deffact (karaciger buyumesi var) absolutely-false
deffact (transaminaz duzeyi yuksek) high
deffact (bulanti ve yorgunluk var) low
deffact (kanda hepatit virusu var) absolutely-high
deffact (27nm buyuklukte RNA virusu var) absolutely-high
```

Şimdi, uzman sistemi çalıştıralım.

```
factbase ( y / n ) ?      : y
enter name of factbase   : a:\belirti1.txt
```

total fact number in the factbase = 7

```
-----
fact 0 serum bilirubin duzeyi 2.5mg/dl uzerinde -- absolutely-high
fact 1 deri ve mukozada sari renk var -- high
fact 2 karaciger buyumesi var -- absolutely-false
fact 3 transaminaz duzeyi yuksek -- high
fact 4 bulanti ve yorgunluk var -- low
fact 5 kanda hepatit virusu var -- absolutely-high
fact 6 27nm buyuklukte RNA virusu var -- absolutely-high
```

enter name of rule base: a:\hastalik.txt

INFERENCE COMPLETED

enter goal: hastalik

NUMERICAL TRUTH VALUES OF FACTS

fact: hastalik hepatit incelemesi yap – 0.1876 0.2506 0.3864 0.4466

fact: hastalik hepatit – 0.5989 0.6291 0.6932 0.7233

fact: hastalik hepatit A – 0.7959 0.8146 0.8466 0.8616

fact: hastalik hepatit B – 0.2995 0.3146 0.3466 0.3616

fact: hastalik AIDS – 0.0000 0.0000 0.0000 0.0000

fact: hastalik diabet olabilir -- 0.0000 0.0000 0.0000 0.0000

fact: hastalik diabet tip 1 -- 0.0000 0.0000 0.0000 0.0000

fact: hastalik diabet tip 2 -- 0.0000 0.0000 0.0000 0.0000

fact: hastalik tuberkuloz -- 0.0000 0.0000 0.0000 0.0000

fact: hastalik lösemi -- 0.0000 0.0000 0.0000 0.0000

fact: hastalik kursun zehirlenmesi -- 0.0000 0.0000 0.0000 0.0000

fact: hastalik Behcet sendromu -- 0.0000 0.0000 0.0000 0.0000

fact: hastalik Reiter sedromu -- 0.0000 0.0000 0.0000 0.0000

fact: hastalik sistemik lupus eritamosus – 0.0000 0.0000 0.0000 0.0000

fact: hastalik Sjorgen sedromu -- 0.0000 0.0000 0.0000 0.0000

CONCLUSION

conclusion : hastalik hepatit incelemesi yap – medium-low

conclusion : hastalik hepatit – medium-high

conclusion : hastalik hepatit A -- high

conclusion : hastalik hepatit B – medium-low

conclusion : hastalik AIDS -- unknown

conclusion : hastalik diabet olabilir -- unknown

conclusion : hastalik diabet tip 1 -- unknown

conclusion : hastalik diabet tip 2 -- unknown

conclusion : hastalik tuberkuloz -- unknown

conclusion : hastalik lösemi -- unknown

conclusion : hastalik kursun zehirlenmesi -- unknown

conclusion : hastalik Behcet sendromu -- unknown

conclusion : hastalik Reiter sedromu -- unknown

conclusion : hastalik sistemik lupus eritamosus – unknown

conclusion : hastalik Sjorgen sedromu -- unknown

Çıkarım sonunda görüldüğü gibi; hastalık sırasıyla; “high doğruluk değeriyle hepatit A” ve “medium-low doğruluk değeriyle hepatit B” olarak teşhis edilmiştir.



EK B

Bu uygulamada, rahatsızlıkları olan bir kişi ile ilgili gerçekler (belirtiler), “belirti2.txt” dosyasında yer almaktadır. Çıkarıma başlamadan önce, belirti2.txt dosyasında yer alan gerçeklere ve bu gerçeklerin bulanık doğruluk değerlerine göz atmak istersek:

```
deffact (karin agrisi var) medium
deffact (gastrointestinal bozukluklar var) medium
deffact (psikolojik bozukluklar var) very-low
deffact (basdonmesi var) medium
deffact (alopesi var) absolutely-false
deffact (disetinde siyah çizgiler var) absolutely-false
```

Şimdi, uzman sistemi çalıştıralım.

```
factbase ( y / n ) ?      : y
enter name of factbase   : a:\belirti2.txt
```

total fact number in the factbase = 6

```
-----
fact 0 karin agrisi var -- medium
fact 1 gastrointestinal bozukluklar var -- medium
fact 2 psikolojik bozukluklar var -- very-low
fact 3 basdonmesi var -- medium
fact 4 alopesi var -- absolutely-false
fact 5 disetinde siyah çizgiler var -- absolutely-false
```

enter name of rule base: a:\hastalik.txt

INFERENCE COMPLETED

```
-----
enter goal: hastalik
```

NUMERICAL TRUTH VALUES OF FACTS

fact : hastalik hepatit incelemesi yap -- 0.0000 0.0000 0.0000 0.0000
fact : hastalik hepatit -- 0.0000 0.0000 0.0000 0.0000
fact : hastalik hepatit A -- 0.0000 0.0000 0.0000 0.0000
fact : hastalik hepatit B -- 0.0000 0.0000 0.0000 0.0000
fact : hastalik AIDS -- 0.0000 0.0000 0.0000 0.0000
fact : hastalik diabet olabilir -- 0.0000 0.0000 0.0000 0.0000
fact : hastalik diabet tip 1 -- 0.0000 0.0000 0.0000 0.0000
fact : hastalik diabet tip 2 -- 0.0000 0.0000 0.0000 0.0000
fact : hastalik tuberkuloz -- 0.0000 0.0000 0.0000 0.0000
fact : hastalik lösemi -- 0.0000 0.0000 0.0000 0.0000
fact : hastalik kursun zehirlenmesi -- 0.0642 0.1078 0.1879 0.2305
fact : hastalik Behcet sendromu -- 0.0000 0.0000 0.0000 0.0000
fact : hastalik Reiter sedromu -- 0.0000 0.0000 0.0000 0.0000
fact : hastalik sistemik lupus eritematosus -- 0.0000 0.0000 0.0000 0.0000
fact : hastalik Sjorgen sedromu -- 0.0000 0.0000 0.0000 0.0000

CONCLUSION

conclusion : hastalik hepatit incelemesi yap -- unknown
conclusion : hastalik hepatit -- unknown
conclusion : hastalik hepatit A -- unknown
conclusion : hastalik hepatit B -- unknown
conclusion : hastalik AIDS -- unknown
conclusion : hastalik diabet olabilir -- unknown
conclusion : hastalik diabet tip 1 -- unknown
conclusion : hastalik diabet tip 2 -- unknown
conclusion : hastalik tuberkuloz -- unknown
conclusion : hastalik lösemi -- unknown
conclusion : hastalik kursun zehirlenmesi -- low
conclusion : hastalik Behcet sendromu -- unknown
conclusion : hastalik Reiter sedromu -- unknown
conclusion : hastalik sistemik lupus eritematosus -- unknown

conclusion : hastalik Sjorgen sedromu -- unknown

Çıkarım sonunda görüldüğü gibi; hastalık; “low doğruluk değeriyle kurşun zehirlenmesi” olarak teşhis edilmiştir.



ÖZGEÇMİŞ

Yusuf Barbaros Yonar, 1973 yılında Gölcük, İzmit'te doğdu.1991 yılında İstanbul Eğitim ve Kültür Vakfı, Uluğbey Özel Deneme Lisesi'nden mezun oldu. Aynı yıl İstanbul Teknik Üniversitesi, Fen Edebiyat Fakültesi, Matematik Mühendisliği Bölümü'nde eğitim görmeye hak kazandı. Aynı bölümden 1995 yılında mezun oldu ve İ.T.Ü Fen Bilimleri Enstitüsü Mühendislik Bilimleri Anabilim Dalı Sistem Analizi Programı'nda yüksek lisans öğrenimine başladı. Pamukbank T.A.Ş. Yazılım Geliştirme Bölüm'ünde Sistem Analist olarak çalışmaktadır.

