BIZONYTALANSÁGKEZELÉSI MODELLEK

Doktori értekezés tézisei

MODELS FOR HANDLING UNCERTAINTY

Ph. D. Thesis

Achs Ágnes

Debreceni Egyetem
Informatikai Kar
Debrecen, 2006.

Tartalomjegyzék – Table of Contents

“Biztosat csak báró Udvarhelyi tud. Hogy minden bizonytalan. Ehhez is mennyit kell tanulni, amíg precízen látja az ember!”

– Rejtő Jenő: A boszorkánymester

Bevezetés

Egy tudásbázis-kezelő rendszerben tények és szabályok reprezentálják a tudást. A tényekből a szabályok segítségével újabb tényekre tudunk következtetni. A klasszikus deduktív adatbázisok elméletében közismert a Datalog-szerű adatmodell. Legáltalánosabb alakjuk megengedi a függvényszimbólumok és a negáció alkalmazását is, dolgozatomban azonban csak a klasszikus, függvényszimbólum-mentes esettel, illetve annak bővítésével foglalkozom. Egy Datalog-szerű program jelentése egy olyan minimális – vagy ha van, legkisebb – modell, amely tartalmazza a tényeket, és kielégíti a szabályokat. Ezt a modellt általában egy fixpontszámítási algoritmussal határozzák meg.

Az emberi tudás jó része azonban nem modellezhető a kétértékű logikán alapuló következtetési rendszerekkel, hiszen ez a tudás gyakran bizonytalan, homályos, esetleg félreérthető vagy hiányos. A bizonytalan információk kezelésére számtalan megoldási javaslat született, ezek közül az egyik ígéretes út a fuzzy logikára épül. Dolgozatomban is ezt az utat választottam.

Néhány évvel ezelőtt kezdtünk el foglalkozni a Datalog szerű nyelvek és a fuzzy logika összekapcsolásával. A Datalog tényeket és szabályokat bizonytalansági szinttel és implikációs operátorral egészítettük ki, és az ily módon kibővített fuzzy Datalog nyelvhez fixpont-szemantikát értelmeztünk. A fuzzy Datalog nyelvet „éles” adatokra definiáltuk, de később kiterjesztettük fuzzy adatokra is. Közben sajnos kimaradt néhány év, de az utóbbi időben egyre erősebben foglalkoztatott a kérdés: nem lehetne-e továbbfolytatni ezt az elképzelést. Így jutottam a fuzzy tudásbázisról alkotott modellhez. Eszerint fuzzy tudásbázist négy elem definiál: a fuzzy Datalog, mint következtetési rendszer, egy háttértudás, amely a termek és predikátumok szinonimáit tartalmazza, egy összekapcsolási algoritmus, amely összeköti az

előbbi két elemet, és dekódoló függvények egy halmaza, melyek segítségével ki tudjuk számolni az eredmény bizonytalansági szintjét.

Kutatásunkkal párhuzamosan, illetve az azóta eltelt években többen foglalkoztak a Prolog és a fuzzy logika összekapcsolásával, különböző javaslatokat adtak a fuzzy egységesítés megvalósítására. A legtöbb megoldási javaslat a hasonlóság fogalmán alapszik. Némelyikük a hasonlóság alapján ekvivalencia-osztályozást valósít meg, és ezeken az osztályokon értelmezi az egységesítést. Van, aki nyelvi változókat definiál, és ezekkel valósítja meg a fuzzy illesztést. Olyan elképzelés is született, amelyben a szerzők az esetleges elírások kezelésére dolgoztak ki egy számítási módszert, az elírásból adódó hibák számával definiálták a termek és predikátumok távolságát, és ezeknek a távolságoknak a felhasználásával értelmezték a fuzzy egységesítést.

A továbbiakban összegzem az eredményeimet.

Fuzzy Datalog

A Datalog nyelv kibővítéseként értelmezhető a fuzzy Datalog (röviden fDATALOG) nyelv. A Datalog tényeit kiegészítjük egy bizonytalansági szinttel, szabályait pedig egy bizonytalansági szinttel és egy implikációs operátorral, és megköveteljük, hogy a szabályhoz tartozó implikáció igazságértéke legalább akkora legyen, mint a szabályhoz tartozó bizonytalansági szint. Ez alapján meg tudjuk határozni a szabályfej bizonytalanságát. Pontosabban fogalmazva:

1. Definíció: fDATALOG szabály egy (r; β; I) hármas, ahol r egy

$$A \leftarrow A_1,...,A_n \qquad (n \geq 0)$$

alakú formula, A atom (a szabály feje), $A_1,...,A_n$ literálok (a szabály törzse), I egy implikációs operátor és $\beta \in (0,1]$ (a szabály bizonytalansági foka vagy szintje).

Ahhoz, hogy mindig véges eredményt kapjunk, szükséges, hogy a program biztonságos legyen. Egy fDATALOG szabály biztonságos, ha a fejben előforduló összes változó szerepel a törzsben is, és az összes olyan változó, amely negatív literálban szerepel, előfordul pozitív literálban is. Egy fDATALOG program biztonságos fDATALOG szabályok véges halmaza. Az ($A \leftarrow$; β; I) alakú szabályokat, ahol A alapatom, tényeknek nevezzük. A továbbiakban ezeket (A;β) módon jelöljük, hiszen az I implikáció alapján az A szintje könnyen kiszámolható.

A fDATALOG szemantikáját rákövetkezési transzformációk fixpontjaként értelmezzük. Két fajta rákövetkezési transzformációt definiáltam, a determinisztikus DT_P és a nem determinisztikus NT_P transzformációt. Ezek fixpontját tekintjük a fDATALOG determinisztikus, illetve nem determinisztikus szemantikájának. Determinisztikus szemantika alapján egy program szabályai párhuzamosan értékelhetők ki, nem determinisztikus esetben egymás után. A szóban forgó transzformációk a következő módon definiálhatók:

2. Definíció: Legyen B_P a P program Herbrand bázisa, és jelölje $F(B_P)$ a B_P fölötti összes fuzzy halmazt. A DT_P: $F(B_P) \rightarrow F(B_P)$ és

NT_P: $F(B_P) \rightarrow F(B_P)$ rákövetkezési transzformációkat a következő módon értelmezzük:

$$DT_P(X) = \{\bigcup\{(A,\alpha_A)\}\} \cup X, \text{ illetve}$$
$$NT_P(X) = \{(A, \alpha_A)\} \cup X,$$

ahol

$$(A \leftarrow A_1,\ldots,A_n ; I ; \beta) \in \textit{ground}(P), (|A_i|,\alpha_{A_i}) \in X, 1 \leq i \leq n,$$
$$\alpha_A = \max(0,\min\{\gamma \mid I(\alpha_{body},\gamma) \geq \beta\}).$$

$|A_i|$ - val az A_i literál magját jelöljük (vagyis azt az atomot, amely, vagy amelynek negáltja a literált alkotja), *ground*(P) pedig a program alapelőfordulása.

Belátható, hogy a tényhalmazból kiindulva, és bizonyos feltételnek eleget tévő implikációs operátorokat alkalmazva, negáció-mentes P programok esetén mindkét transzformációnak van legkisebb fixpontja, amely egyúttal a program minimális modellje is. Az $lfp(DT_P)$, illetve $lfp(NT_P)$-vel jelölt fixpontokat tekintjük a fDATALOG program determinisztikus, illetve nem determinisztikus szemantikájának.

Bizonyítható az is, hogy függvény- és negációmentes program esetén a két szemantika megegyezik, negációt tartalmazó programok esetén azonban a kettő eltérhet egymástól. Az is előfordulhat, hogy a determinisztikus transzformáció fixpontja nem minimális modell, nem determinisztikus esetben azonban bizonyos feltételek mellett biztosítható a minimalitás. Ez a feltétel a rétegzés, amely meghatároz egy kiértékelési sorrendet. Eszerint a negatív literálokat korábban értékeljük ki, mint a nem negáltakat. Nem minden program rétegezhető, de belátható, hogy rétegzett programok esetén létezik olyan kiértékelési sorrend, amely szerint alkalmazva a nemdeterminisztikus rákövetkezési transzformációt, a kapott legkisebb fixpont a program minimális modellje.

1. Példa Tekintsük a következő fDATALOG programot:

```
1.    (r(a); 0.8.)
2.    p(x) ← r(x),¬q(x); 0.6; I2.
3.    q(x) ← r(x); 0.5; I3.
4.    p(x) ← q(x); 0.8; I4.
```

Az egyszerűség kedvéért legyen most az összes implikációs operátor a Gödel féle:

$$I(\alpha,\beta) = \begin{cases} 1 & \text{ha } \alpha \le \beta, \\ \beta & \text{egyébként} \end{cases}$$

A helyes rétegzés {r,q}, {p}, vagyis a kiértékelési sorrend: 1., 3., 2., 4. (Pontosabban: először 1. és 3. tetszőleges sorrendben, majd 2. és 4. tetszőleges sorrendben.)
Ekkor

$lfp(NT_P) = \{(r(a),0.8);\ (p(a),0.5);\ (q(a),0.5)\},$

Vagyis r(a) legalább 0.8 szinten igaz, p(a) és q(a) legalább 0.5 szinten.

Kiértékelési stratégiák

Egy fDATALOG programot különböző stratégiákkal értékelhetünk ki. Ha a tényekből kiindulva az adott szabályok alkalmazásával következtetünk az összes előállítható tényre, vagyis meghatározzuk a determinisztikus (nem determinisztikus) transzformáció fixpontját, akkor bottom-up, vagyis adatvezérelt következtetésről beszélünk. Ez egy egyszerű iteráció, esetenként sok fölösleges számolással. Ennek csökkentése céljából vezettem be a módosított adatvezérelt kiértékelést, amely bizonyos feltételek mellett redukálja az iterációs lépések számát.

Sokszor előfordulhat azonban, hogy nincs szükség a teljes kiértékelésre, hiszen egy konkrét kérdésre keressük a választ, el akarjuk dönteni egy atom (atomok) igaz vagy hamis voltát, illetve bizonytalansági fokát, vagy meg akarjuk tudni, hogy egy atom milyen értékek mellett igaz vagy hamis, milyen értékek mellett igaz legalább β szinten, vagy milyen értékek esetén milyen szinten igaz. Ez azt jelenti, hogy egy cél ismeretében elég „célirányosan" végezni a kiértékelést, vagyis elég csak a cél eléréséhez szükséges szabályokat, tényeket figyelembe venni. Azt a fajta kiértékelési stratégiát, amikor a célból kiindulva a megfelelő szabályok alkalmazásával a tények felé következtetve értékeljük ki a szabályokat, top-down, vagy célvezérelt kiértékelésnek nevezzük.

A célvezérelt kiértékelés általában azt jelenti, hogy kiindulva a célból, újabb részcélokat generálunk úgy, hogy kiválasztjuk az összes olyan szabályt, amelynek feje illeszthető az adott céllal, és a szabálytörzs atomjait újabb részcéloknak tekintjük. Ezt az eljárást addig folytatjuk, amíg el nem jutunk a tényekig. Egy fuzzy Datalog program kiértékelése során is a cél felől haladunk a tényekig, de nem állunk meg, amikor elértük őket, hiszen a cél bizonytalansági szintjét is meg kell határoznunk.

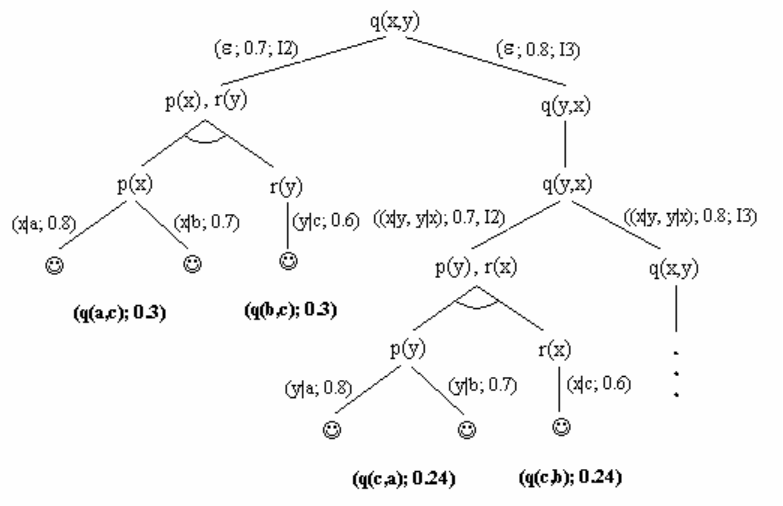
A kiértékelést az úgynevezett kiértékelési gráf segítségével hajthatjuk végre. Ez egy ÉS/VAGY fa-gráf, vagyis egy speciális hipergráf, melynek – a kiértékelendő célt tartalmazó gyökér szintjét 0-nak tekintve – minden páratlan mélységű éle n-edrendű hiperél, a hozzá tartozó n elemből álló halmaz-csúccsal, páros mélységű élei pedig közönséges élek. A gráf minden páros szintjén kiértékelésre váró részcélok, azaz megfelelő módon illesztett szabályfejek, minden páratlan szintjén kirtékelendő szabálytörzsek szerepelnek. A gráf levelei az IGAZ/ HAMIS szimbólumok: ha a részcélt sikerült egy tényállítással illeszteni, akkor az IGAZ, ha egyetlen illeszthető szabályfej sincs, akkor a HAMIS szimbólum.

Megcímkézzük a gráf közönséges éleit, a címke az alkalmazott helyettesítést, az adott éllel reprezentált szabály bizonytalansági szintjét és implikációs operátorát tartalmazza. A lekérdezésre a kiértékelési gráf címkéiből kapjuk meg a választ. Az IGAZ szimbólumban végződő hiperutak mentén a levelekből a gyökér felé haladva a címkék segítségével ki tudjuk számolni a szükséges bizonytalansági értékeket, végül meghatározható a gyökér bizonytalansági szintje is.

2. Példa: Tekintsük a következő szabályokat:

```
(p(a); 0.8).
(p(b); 0.7).
(r(c); 0.6).
q(x,y)  ← p(x),r(y); 0.7; I2.
q(x,y)  ← q(y,x); 0.8; I3.
s(x)    ← q(x,y); 0.9; I3.
```

Célunk (q(x,y), α) meghatározása.



Az ábrán látható ÉS/VAGY gráf alapján a lekérdezés eredménye:
{(q(a,c), 0.3) , (q(b,c), 0.3) , (q(c,a), 0.24) , (q(c,b), 0.24)}.

Bebizonyítottam, hogy adott célhalmaz és véges kiértékelési gráf esetén a top-down kiértékelési algoritmus ugyanazt a megoldást adja, mint a fixpontkeresés. Előfordulhat azonban, hogy a célvezérelt kiértékelés nem terminál. Belátható, hogy ebben az esetben is megadható olyan mélységi korlát, amellyel korlátozva a kiértékelést, az megáll, és megadja a célnak megfelelő összes megoldást. Esetenként ez az algoritmus is sok munkával jár. Bizonyos heurisztikák bevezetésével azonban lerövidíthető, illetve abban a speciális esetben is egyszerűsíthető, ha az összes implikációs operátor a Gödel féle. Ekkor ugyanis a szabályfej bizonytalansági szintje ugyanúgy számlató, mint a törzs bizonytalansági szintje, és a fenti hipergráf helyett elég közönséges kiértékelési gráffal dolgoznunk. Dolgozatomban ezt az egyszerűsített algoritmust is meghatároztam.

A fDATALOG és a fuzzy relációk közötti kapcsolat

Érdemes megvizsgálni a fDATALOG és a fuzzy relációk közötti kapcsolatot.

Egy közönséges Datalog programhoz hozzárendelhető egy relációs adatbázis, vagyis a program relációkkal interpretálható úgy, hogy minden k argumentumú p predikátumhoz hozzárendelünk egy k argumentumú R relációt oly módon, hogy R-be azok a sorok kerüljenek, amelyekre a p predikátum igaz. A program szabályaihoz pedig egy-egy relációs algebrai egyenlet rendelhető. Az ilyen egyenletekből álló egyenletrendszer megoldása függvény- és negációmentes esetben a program legkisebb modelljét, rétegzett Datalog esetén pedig egy – a rétegzés sorrendjétől függő – minimális modelljét adja.

A fenti esethez hasonlóan a fDATALOG predikátumokhoz is rendelhetünk relációkat, illetve a fDATALOG szabályokhoz is hozzárendelhetünk egy-egy fuzzy relációs algebrai kifejezést. A hozzárendelés azonban nem olyan egyszerű, mint a közönséges Datalog esetén, az irodalomban ugyanis többféle módon értelmezik a fuzzy relációkat. Jómagam eddig két relációtípussal foglalkoztam, az úgynevezett első és második típusú relációval. Megmutatható, hogy az első típusú relációk és a rajtuk értelmezett relációs algebrai kifejezések hozzárendelhetőek a fDATALOG programokhoz, míg a második típusú relációk a fDATALOG kiterjesztéséhez vezetnek.

A hagyományos relációs adatbázis tábláinak egy-egy sora vagy beletartozik egy adott relációba, vagy nem. Fuzzy esetben minden sorhoz hozzárendelhetünk egy 0 és 1 közötti értéket, amely a sor relációba való tartozásának mértékét mutatja.

3. Definíció: A D_1, ..., D_n halmazon értelmezett első típusú fuzzy reláció a D_1, ..., D_n Descartes szorzatának fuzzy részhalmaza:

$$\mu_R: D_1 \times ... \times D_n \rightarrow [0,1].$$

$\mu_R(\underline{t})$ a $\underline{t}$ sor R relációba való beletartozásának mértékét mutatja.

3. Példa: A (B(X,Y), μ_B) első típusú fuzzy reláció, (pl. barátság):

B:	X	Y	μ_B
	Jani	Pali	0.1
	Márta	Éva	0.6

Nyilvánvaló, hogy egy fDATALOG P program minden egyes r predikátumához hozzá tudunk rendelni egy első típusú R fuzzy relációt úgy, hogy

$$\underline{t} = (a_1, ...,a_n,\mu_R(\underline{t})) \in (R(A_1, ...,A_n), \mu_R) \Leftrightarrow (r(a_1, ...,a_n),\mu_R(\underline{t})) \in lfp(DT_P) \text{ (vagy } lfp(NT_P)).$$

(A_1,..., A_n az R reláció attribútumait jelöli.)

Belátható, hogy negáció- és függvénymentes fDATALOG program determinisztikus (nem determinisztikus) szemantikája ekvivalens a fDATALOG relációs algebrai kiértékelésével abban az értelemben, hogy adott program esetén mindkét megközelítés ugyanazt a fixpontot eredményezi. Rétegezhető programok és adott rétegzés esetén, ha a fixpontszámítást a rétegzés sorrendjében hajtjuk végre, akkor a logikai és algebrai módszerrel kapott fixpont megegyezik, s ez a fixpont a program minimális modellje.

A fDATALOG fuzzy adatokra való kiterjesztése

Az első típusú fuzzy relációk azt mutatják meg, hogy mennyire szoros kapcsolat áll fenn az attribútum-értékek között. Ilyen relációk esetén az attribútum-értékek határozott adatok, nem tartalmaznak bizonytalanságot. Sokszor azonban maguk az értékek is bizonytalanok, vagyis számos esetben fuzzy adatok közötti relációra van szükségünk. Ezért válik szükségessé a második típusú fuzzy relációk értelmezése, amelyek segítségével megadhatjuk a fDATALOG egy lehetséges kiterjesztését.

4. Definíció: Legyen D1,...,Dn tetszőleges halmaz és F(D1),...,F(Dn), $1 \leq i \leq n$ a rajtuk értelmezett fuzzy részhalmazok! A D1, ..., Dn hal-

mazon értelmezett második típusú fuzzy reláció az F(D1) ×...× F(Dn) egy fuzzy részhalmaza:μR: F(D1) ×...× F(Dn) → [0,1]

4. Példa:

Az

R:

Név	Kor	Fizetés	μ_R
János	31	{(60000, 0.8); (70000, 0.9)}	0.7
Péter	középkorú	80000	0.8
Anna	fiatal	{(80000, 0.7); (85000, 0.6)}	0.9

egy második típusú fuzzy reláció. A harmadik sor például így értelmezhető: Majdnem biztosak vagyunk benne, hogy Anna fiatal, és fizetése talán 80-, esetleg 85 ezer Ft körül lehet.

Célszerű lenne úgy kitrerjeszteni a fDATALOG nyelvet, hogy az alkalmas legyen második típusú fuzzy relációk kezelésére, vagyis úgy, hogy a program predikátumaihoz egy-egy második típusú fuzzy relációt tudjunk rendelni. Ezért megengedjük a fuzzy konstansok alkalmazását és azt, hogy a változók fuzzy adatok valamely halmazán fussanak végig.

Formálisan egy fDATALOG szabály ugyanolyan lesz, mint az előzőekben, csak a kiértékelés módját kell megváltoztatnunk. A nehézséget a fuzzy adatok illesztése okozza. Kérdés, hogyan tudunk két fuzzy értéket illeszteni egymással, azaz hogyan tudunk összehasonlítani két fuzzy halmazt. Mivel ez a kérdés komoly problémákat vet fel, ezért a továbbiakban is ragaszkodunk a szigorú illesztéshez, és a bizonytalanság kezelését szomszédsági predikátumok bevezetésével oldjuk meg.

5. Definíció: Egy D tartomány fölötti $prox_D$: D × D → [0,1] leképezés szomszédsági predikátum, ha reflexív és szimmetrikus, azaz, ha

$prox_D(x,x) = 1$ minden $x \in D$ esetén, és

$prox_D(x,y) = prox_D(y,x)$ minden $x,y \in D$ esetén.

Ha a szomszédság tranzitív, azaz

$$prox_D(x, z) \geq \min(prox_D(x, y), prox_D(y, z))$$

teljesül minden x, y, z esetén, akkor hasonlóságnak nevezzük.

A szomszédságot leíró tényeket is hozzávéve a fDATALOG szabályokhoz, jóval bonyolultabb szabályokat kapunk, melyek átírására megadtam egy algoritmust. Bebizonyítható, hogy tranzitív hasonlósági reláció esetén egyszerűsíthető a kiértékelés.

5. Példa: Az előző példa adataiból kiindulva tegyük fel, hogy az R relációnak megfeleltethető a „személy" predikátum, melynek argumentumai az R sorai. Tekintsük a következő szabályt:

```
főnök(x)←személy(x,40_körüli,80_körüli);0.8;I.
```

Az átírt szabály:

$$\text{főnök}(x) \leftarrow \text{személy}(x_1, y_1, z_1),\ \text{prox}_1(x, x_1),\ \text{prox}_2(40\_\text{körüli}, y_1),\ \text{prox}_3(80\_\text{körüli}, z_1);0.8;I.$$

Természetesen a kiértékeléshez definiálnunk kell az itt szereplő hasonlósági predikátumokat, például $\text{prox}_2(40\_\text{körüli}, \text{középkorú}) = 0.9$, stb.

6. Példa: Bár a javasolt kiterjesztés sok munkával jár, de cserében ilyen jellegű programot is tudunk kezelni:

```
személy('Aladár', sovány, 35); 0.9.
személy('Béla', átlagos, 40); 0.8.
tanú_termet(sovány); 0.6.
tanú_kor(38); 0.7.
tanú_kor(40); 0.8.
gyanús(X)←   személy(X,Y,Z), tanú_termet(Y),
             tanú_kor(Z); 0.8; I.
```

(Jelentése: Nagyjából tudjuk Aladár és Béla termetét, korát, a szemtanúk is emlékeznek valamilyen termetre és valamilyen korra. Egy személy eléggé gyanús, ha életkora és termete olyasmi, mint amilyennek a szemtanúk látták.)
A példában szereplő adatok szomszédsága például az alábbi módon jellemezhető: A személyneveknél megköveteljük az azonosságot, a korok közötti szomszédságot az sz_k, a termetek közötti szomszédságot az sz_t mátrix írja le:

sz_k	35	38	40
35	1	0.8	0.7
38	0.8	1	0.9
40	0.7	0.9	1

sz_t	sovány	átlagos	kövér
sovány	1	0.7	0.2
átlagos	0.7	1	0.6
kövér	0.2	0.6	1

Fuzzy tudásbázis

Kérdés, hogyan lehetne még szélesebb körre kiterjeszteni a fuzzy Datalogot. Ennek vizsgálata vezetett egy lehetséges fuzzy tudásbázis modelljének megalkotásához, melyet a következő négy elem együtteseként értelmezhetünk:

6. Definíció: Fuzzy tudásbázison (fKB) egy négyelemű (P, Bk, Φ_P, mA) halmazt értünk, ahol P egy fuzzy Datalog program, Bk a háttértudás, Φ_P a P dekódoló halmaza, és mA valamilyen módosítási (vagy összekapcsoló) algoritmus.

Hogy érthető legyen ez a definíció, tisztáznunk kell a szükséges fogalmakat:

A szomszédsági reláció segítségével definiálhatunk egy úgynevezett háttértudást úgy, hogy minden lehetséges adat és minden lehetséges predikátumnév esetén megadjuk az ismert szinonimáikat, vagyis a velük szomszédos predikátumneveket, illetve adatokat. Az így képzett szomszédsági halmazok uniója alkotja a háttértudást.

7. Definíció: Legyen $d \in D$ a D tartomány egy eleme! A d szomszédsági halmazán D egy fuzzy részhalmazát értjük:

$$SD_d = \{(d_1 , \lambda_1), (d_2 , \lambda_2), \ldots, (d_n , \lambda_n)\},$$

ahol $d_i \in D$ és $prox_D(d, d_i) = \lambda_i$ minden i = 1,…n esetén.
Legyen C alaptermek, R pedig predikátumszimbólumok egy halmaza. Legyen $prox_C$ és $prox_R$ valamilyen szomszédság C, illetve R fölött. Háttértudáson C és R szomszédsági halmazainak unióját értjük:

$$Bk = \{SC_t \mid t \in C\} \cup \{SR_p \mid p \in R\}$$

Az összekapcsolási algoritmus összekapcsolja a háttértudást a programmal. Az algoritmus segítségével egy módosított programot kapunk. Ezt kiértékelve jutunk az eredmény-tényekhez, de a hozzájuk tartozó bizonytalansági szint értéke még nem megfelelő, hiszen a kifejezésben szereplő predikátumnévhez és az adatokhoz tartozó szomszédsági értékeket is figyelembe kell venni. A probléma az úgynevezett „visszafejtő" vagy dekódoló függvény segítségével oldható meg. Ez a módosított program eredményének bizonytalansági szintjéből és a felhasznált szomszédsági értékekből kiszámítja az eredmény tényleges bizonytalansági szintjét. Elvárható, hogy azonosság esetén a dekódoló függvény ne változtassa meg az eredeti szintet, de egyéb esetekben legfeljebb akkora legyen az értéke, mint az eredeti szint vagy a hasonlóság szintje. Azt is megköveteljük, hogy a dekódoló függvény minden változójában monoton növekvő legyen:

8. Definíció: Dekódoló függvényen egy olyan (n+2)-változós

$$\varphi(\alpha, \lambda, \lambda_1, \ldots, \lambda_n) : (0,1] \times (0,1] \times (0,1] \times \ldots \times (0,1] \rightarrow [0,1]$$

valós függvényt értünk, melyre

$\varphi(\alpha, \lambda, \lambda_1, \ldots, \lambda_n) \leq \min(\alpha, \lambda, \lambda_1, \ldots, \lambda_n)$,

$\varphi(\alpha, 1, 1, \ldots, 1) = \alpha$ és

$\varphi(\alpha, \lambda, \lambda_1, \ldots, \lambda_n)$ minden argumentumában monoton növekvő.

Érdemes megemlíteni, hogy bármely trianguláris norma eleget tesz a dekódoló függvény feltételeinek, pl. a mininimumképzés vagy a szorzás operátorok t-normák.

Természetesen predikátumonként más-más függvényt kell, vagy lehet definiálni, ezek a függvények alkotják az úgynevezett dekódoló halmazt.

Kétféle összekapcsolási algoritmust értelmeztem, egy egyszerű és egy transzformációs módosítást. Az első szerint a programot alakítjuk át úgy, hogy minden predikátumát és alap-termjét a szomszédsági halmazával helyettesítjük, és a programot az eredeti rákövetkezési transzformációval értékeljük ki. Mindkét esetben a rákövetkezési transzformáció legkisebb fixpontjaként definiáljuk a tudásbázis következményét, $C(P, Bk, \Phi_P, mA)$-t. Ez a tudásbázisból levezethető tények halmaza, ahol minden tényállításhoz egy kiszámított érték is tartozik,

amely azt mutatja, hogy legalább milyen szinten érvényes ez az állítás, vagyis hogy „mennyire biztos a bizonytalan”.

Az egyszerű módosítási algoritmus a program minden predikátum-szimbólumához és termjéhez hozzárendeli a szomszédsági halmazát, majd az így módosított programot az eredeti szemantika alapján értékeljük ki. Az eredményül kapott fixpont még a hasonlósági halmazokat tartalmazza, ezekből a dekódoló függvények segítségével tudjuk meghatározni a végső alaptermeket. Ezek a termek alkotják a módosított program fixpontját:

9. Definíció: Az egyszerű módosítással (mA1) módosított mP program legkisebb fixpontja:

$$C(P, Bk, \Phi_P, mA1)=lfp(mP)=\bigcup\{(q(t_1,t_2,\ldots, t_n); \varphi_q (\alpha_q,\lambda_q,\lambda_{t_1},\ldots, \lambda_{t_n})) \mid$$
$$\forall(SR_q (SC_{t_1}, SC_{t_2}, \ldots, SC_{t_n}) ; \alpha) \in lfp(NT_{mP}),$$
$$(q , \lambda_q) \in SR_q, (t_i , \lambda_{t_i}) \in SC_{t_i}, 1 \le i \le n\}.$$

A transzformációs módosítás szerint a programot változatlanul hagyjuk, és a kiértékelési transzformációt módosítjuk. Ehhez a szomszédsági relációnak megfelelő elemekkel ki kell bővítenünk a program Herbrand univerzumát és Herbrand bázisát, majd a kibővített Herbrand bázis (mB_P) fuzzy részhalmazain definiálhatunk egy rákövetkezési transzformációt. A transzformációs módosítással (mA2) összekapcsolt tudásbázis következménye az alábbi transzformáció legkisebb fixpontja:

10. Definíció: Az mNT_P: $F(mB_P) \rightarrow F(mB_P)$ módosított rákövetkezési transzformációt a következőképpen értelmezzük:

$$mNT_P(X)=\{(q(s_1,\ldots,s_n), \varphi_p(\alpha, \lambda_q, \lambda_{s_1}, \ldots, \lambda_{s_n})|$$
$$(q, \lambda_q) \in SR_p; (s_i, \lambda_{s_i}) \in SC_{t_i}, 1 \le i \le n\} \cup X,$$

ahol

$$(p(t_1,\ldots,t_n) \leftarrow A_1,\ldots,A_k ; I ; \beta) \in \mathit{ground}(P),$$
$$(|A_i|, \alpha_{A_i}) \in X , 1 \le i \le k, \alpha = \max(0,\min\{\gamma \mid I(\alpha_{törzs}, \gamma) \ge \beta\}).$$

($|A_i|$ - val az A_i literál magját jelöljük.)

Belátható, hogy a program tényhalmazából kiindulva, a fent definiált transzformációnak létezik fixpontja. Mivel a módosítás nem befolyásolja a szabályok sorrendjét, ezért az a rétegzésre sincs hatással, vagyis a fenti transzformációnak rétegzett program esetén is van legkisebb fixpontja.

Mindkét módosítás esetén belátható, hogy az eredeti program nemdeterminisztikus fixpontja részhalmaza a tudásbázis következményének. Az is belátható, hogy

$$C(P, Bk, \Phi_P, mA1) \subseteq C(P, Bk, \Phi_P, mA2).$$

7. Példa: Tekintsük a következő tudásbázist:

sz(x,y) $\leftarrow$ jo(y), mu(x); 0.7; I.	(jo(BB), 0.9).
m(x,y) $\leftarrow$ ke(x,y), ko(y); 0.7; I.	(mu(P), 0.8).
(k(V), 0.9).	(ko(B), 1).
(zk(M), 0.8).	(ko(K), 1).

$\phi_{sz} := \phi_{sz}(\alpha, \lambda_{sz}, \lambda_1, \lambda_2) := \min(\alpha, \lambda_{sz}, \lambda_1, \lambda_2)$
$\phi_m := \phi_m(\alpha, \lambda_m, \lambda_1, \lambda_2) := \min(\alpha, \lambda_m, (\lambda_1 \cdot \lambda_2))$
$\phi_k := \phi_k(\alpha, \lambda_k, \lambda) := \alpha \cdot \lambda_k \cdot \lambda$
$\phi_{zk} := \phi_{zk}(\alpha, \lambda_{zk}, \lambda) := \min(\alpha, \lambda_{zk}, \lambda)$
$\phi_{jo} := \phi_{jo}(\alpha, \lambda_{jo}, \lambda) := \min(\alpha, \lambda_{jo}, \lambda)$
$\phi_{mu} := \phi_{mu}(\alpha, \lambda_{mu}, \lambda) := \min(\alpha, \lambda_{mu}, \lambda)$
$\phi_{ko} := \phi_{ko}(\alpha, \lambda_{ko}, \lambda) := \min(\alpha, \lambda_{ko})$, ha $\lambda=1$ 0 ha $\lambda<1$

$$I(\alpha, \beta) = \begin{cases} 1 & \text{ha } \alpha \le \beta, \\ \beta & \text{egyébként} \end{cases}$$

	B	V	H	BB	K	P	M
B	1	0.9	0.7				
V	0.9	1	0.6				
H	0.7	0.6	1				
BB				1	0.8		
K				0.8	1		
P						1	
M							1

	sz	ke	jo	k	mu	zk	ko	m
sz	1	0.8						
ke	0.8	1						
jo			1	0.75				
k			0.75	1				
mu					1	0.6		
zk					0.6	1		
ko							1	
m								1

Ekkor
C(P, Bk, Φ_P, mA1) = {(k(V),0.9); (k(B),0.81); (k(H),0.54); (jo(V),0.675); (jo(B),0.6075); (jo(H),0.405); (zk(M),0.8); (mu(M),0.6); (jo(BB), 0.9); (jo(K),0.8); (k(BB),0.75); (k(K),0.75); (mu(P),0.8); (zk(P),0.6); (ko(B),1); (ko(K),1); (sz(P,BB),0.7); (sz(P,K),0.7); (ke(P,BB),0.7); (ke(P,K),0.7)}.

C(P, Bk, Φ_P, mA2) = {(k(V),0.9); (k(B),0.81); (k(H),0.54); (jo(V),0.675); (jo(B),0.6075); (jo(H),0.405); (zk(M),0.8); (mu(M),0.6); (jo(BB),0.9); (jo(K),0.8); (k(BB),0.75); (k(K),0.75); (mu(P),0.8); (zk(P),0.6); (ko(B),1); (ko(K),1); (sz(M,V),0.6); (sz(M,B),0.6); (sz(M,H),0.405); (sz(M,BB),0.6); (sz(M,K),0.6); (sz(P,V),0.675); (sz(P,B),0.6075); (sz(P,H),0.405); (sz(P,BB),0.7); (sz(P,K),0.7); (ke(M,V),0.6); ke(M,B),0.6), (ke(M,H),0.405); (ke(M,BB),0.6); ke(M,K),0.6); (ke(P,V),0.675); ke(P,B),0.6075); (ke(P,H),0.405); (ke(P,BB),0.7); (ke(P,K),0.7); (m(M,B),0.6); (m(M,K),0.6); (m(P,B),0.6075); (m(P,K),0.7)}.

Megjegyzés: A fenti tudásbázishoz a következő „jelentés” rendelhető: tegyük fel, hogy a muzsikusok (mu) általában (vagyis 0.7 szinten) szeretik (sz) a jó zeneszerzőket (jo). Ha valamelyik zeneszerző műveiből koncertet (ko) rendeznek, akkor egy őt kedvelő (ke) ember általában (azaz 0.7 szinten) elmegy (m) a koncertre. Tudjuk, hogy Márta (M) eléggé (0.8 szinten) kedveli a zenét (zk). Azt is tudjuk, hogy Vivaldi (V) általában (0.9 szinten) kedvenc zeneszerző (k), és

azt is, hogy Vivaldi, Bach (B) és Händel (H) hasonló stílusúak. További információ, hogy Bartók (BB) jó zeneszerző, és stílusa hasonló Kodályéhoz (K). Két koncertet is meghirdetnek, egyet Bach, egyet Kodály műveiből. Vajon következtethetünk-e arra, hogy Márta mennyire kedveli Bachot? Vagy arra, hogy mennyire biztos, hogy a muzsikus Péter (P) meghallgatja a Kodály hangversenyt? Azt tapasztaltuk, hogy ha az mA1 algoritmussal kötjük össze a programot és a háttértudást, akkor ezekre a kérdésekre nem tudunk válaszolni, az mA2 algoritmussal viszont igen.

A fuzzy tudásbázis esetén is felmerül az adatvezérelt, illetve a célvezérelt kiértékelés kérdése. A szomszédsági reláció alkalmazása miatt az adatvezérelt bottom-up kiértékelés esetenként sok fölösleges munkával járhat, a célvezérelt top-down kiértékelésnél viszont az esetleges illesztések kérdését kell végiggondolni. Mind a kétféle tudásbázishoz kifejlesztettem top-down kiértékelési stratégiát is.

Az mA1 algoritmussal összekapcsolt tudásbázisban a módosított mP program ugyanolyan szerkezetű, mint az eredeti fuzzy Datalog program, ezért egy cél ismeretében mP is kiértékelhető célvezérelt módon. Ahhoz, hogy ezt megtegyük, a program módosításához hasonlóan, a szomszédsági halmazok felhasználásával átalakítjuk az adott $(q(x_1, x_2,\ldots x_n); \alpha)$ célt a módosított $(SR_q(SC_{x_1}, SC_{x_2}, \ldots, SC_{x_n}); \alpha)$ céllá. Az így keletkező kérdésre a fDATALOG kiértékelésekor vázolt módon kapunk választ, majd a dekódoló függvények segítségével meghatározhatunk egy válasz-halmazt, amely tartalmazza az eredeti kérdésre nyert választ is.

Az mA2 algoritmussal összekapcsolt tudásbázis bonyolultabb rákövetkezési transzformációra épül, és következménye bővebb, mint az mA1 algoritmussal összekapcsolt tudásbázisé. Emiatt még inkább indokolt a top-down kiértékelés. Ugyanakkor – legalábbis jelenleg – nem megoldott a tisztán felülről lefelé való építkezés, ehhez ugyanis a kiértékelés során meg kellene oldani a fuzzy egységesítést, illetve a dekódoló függvények valamilyen értelmű invertálását. A közönséges fuzzy Datalog kiértékeléséhez hasonlóan most is két irányban – föntről lefelé, majd alulról fölfelé – haladunk a kiértékeléssel, sőt, a

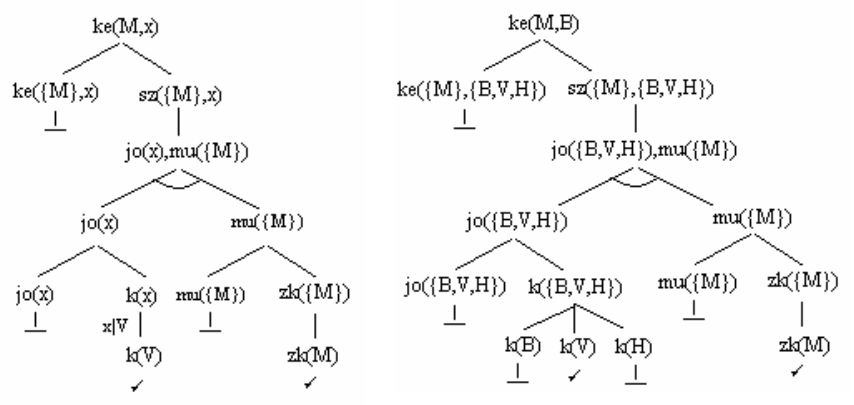
„bottom-up” kiértékelésre támaszkodunk, de „top-down” módon választjuk ki a cél megválaszolásához szükséges kiindulási tényeket.

Az eljárás célja tehát, hogy meghatározza a válasz megadásához szükséges kiindulási tényeket. A kiindulási halmaz meghatározásához nincs szükségünk a bizonytalansági szintekre, ezért a keresési gráf most csak közönséges Datalog tényekre és szabályokra épül. A mostani kiértékelés során is felépítünk egy ÉS/VAGY gráfot, az úgynevezett keresési fát, melynek gyökere a cél, levelei pedig az IGAZ/HAMIS szimbólumok. Az IGAZ levelek szülő-csúcsai a keresett kiindulási tények. Ez a keresési fa a hasonlóságon és a szabályokon alapuló egységesítés váltakozásával épül fel.

A szabályokon alapuló helyettesítés a részcélt a vele illeszthető szabályfejekkel egységesíti, és a kiértékelést a szabálytörzzsel folytatja. Az egységesítés során alkalmazott helyettesítés speciális abban az értelemben, hogy egy konstansot a hasonlósági halmazával helyettesítünk, illetve az argumentumokban már szereplő hasonlósági halmazok úgy viselkednek, mint a közönséges egységesítés konstansai.

A hasonlóság alapú egységesítés a rész-cél predikátumszimbólumát helyettesíti hasonlósági halmaza tagjaival. Az első és az utolsó hasonlóság alapú egységesítés eltér a többitől. Az első a cél alap-term-jeit egységesíti a hasonlósági halmazukkal – ezek a halmazok a kiértékelés végéig konstansként viselkednek. Az utolsó ennek a fordítottját végzi: az eredmény tények argumentumaiban szereplő hasonlósági halmazokat helyettesíti az elemeikkel.

8. Példa: Egészítsük ki az előző példa programját a ke(M,x), illetve a ke(M,B) céllal, ahol x változó, M és B konstans. A lekérdezések keresési gráfjai:



Mindkét esetben az X_0 = {(k(V),0.9), (zk(M,0.8)} halmazból kell kiindulnunk a fixpontszámoláskor. Így szűkebb megoldáshalmazt kapunk, mint az előző feladatban, de ez a halmaz tartalmazza a kérdésekre adható választ.

Összegzés

Dolgozatomban bizonytalan információk kezelésével kapcsolatos modelleket igyekeztem kidolgozni. E célból definiáltam a Datalog nyelv kibővítéseként értelmezhető fuzzy Datalog nyelvet, annak szintaktikáját, szemantikáját, különböző kiértékelési stratégiáit. A fuzzy Datalog „éles" adatokkal és „éles" predikátumnevekkel dolgozik. Ez azonban a szomszédsági reláció segítségével kiterjeszthető fuzzy adatokra, és egymással csak bizonyos szinten hasonló predikátumnevekre is. Ily módon jutunk egy lehetséges fuzzy tudásbázishoz. Dolgozatomban a fuzzy tudásbázis értelmezésén túl annak néhány lehetséges kiértékelésével is foglalkoztam.

A dolgozat megírása után is maradtak nyitott kérdések: milyen összekapcsolási algoritmusokat lehetne még definiálni egy tudásbázishoz; milyen esetleges további tulajdonságokkal kell rendelkezniük a dekódoló függvényeknek; hogyan lehetne hatékonyabbá tenni a kiértékelést, akár az esetleges fuzzy illesztés segítségével; hogyan lehetne ügyesen strukturálni a háttértudást; stb. Ezekre a kérdésekre vagy egy részükre talán valamikor még sikerül választ találni.

"Only baron Udvarhelyi knows for certain. That every is uncertain. How much one has to learn even to see that precisely!"

– Rejtő Jenő: The witch-master

Introduction

In knowledge-base systems there are given some facts representing certain knowledge and some rules, according which one can deduce other kinds of information. In classical deductive database theory the Datalog-like data model is widely spread. Its most general type allows the use of both function symbols and negation, but in this dissertation the use of function symbols is not allowable. The meaning of a Datalog-like program is the least (if it exists) or a minimal model, which contains the facts and satisfies the rules. This model is generally computed by a fixpoint algorithm.

However the large part of human knowledge can't be modelled by pure inference systems, because this knowledge is often ambiguous, incomplete and vague. The study of inference systems is faced in literature with several and often very different approaches. When knowledge is represented as a set of facts and rules, this uncertainty can be handled by means of fuzzy logic. In my dissertation also this way was discussed.

A few years ago we were started to deal with a possible combination of Datalog-like languages and fuzzy logic. In our works there was introduced the concept of fuzzy Datalog by completing the Datalog-rules and facts by an uncertainty level and an implication operator. The semantics of fuzzy Datalog was defined as a fixpoint-semantics. We gave an extension of fuzzy Datalog to fuzzy relational databases too. Unfortunately I missed a few years, but recently a question have been arisen: how I can continue this former concept. So I attained a possible model of fuzzy knowledgebase, which is defined as a quadruple of any background knowledge, defined by the proximity of predicates and terms; a deduction mechanism: a fuzzy Datalog program; a connecting algorithm, which connects the background

knowledge with the program and a decoding set of the program, which help us to determine the uncertainty level of the results.

Parallel with these works, there were researches on possible combination of Prolog language and fuzzy logic. Several solutions were arisen for this problem. These solutions propose different methods for handling uncertainty. Most of them use the concept of similarity, but in various ways. Some of them take any classification according to similarities and use this equivalence classes to make unification and fuzzy resolution. Others deal with linguistic variables and their linguistic values, and give definition of the fuzzy unification algorithm by means of these values. There is such a concept, according which the authors deal with the problem of missing parameters and mismatching predicates and parameters. They define the edit distance of terms, which is compound according to the number of mismatching, and give the unification algorithm according to these distances.

In the next I give a short summary of my results.

Fuzzy Datalog

In fuzzy Datalog (fDATALOG), we can complete the facts by an uncertainty level, the rules by an uncertainty level and an implication operator, which means, that evaluating the fuzzy implication connecting to the rule, its truth-value according to the implication operator is at least the uncertainty level of the rule. According to this operator we can compute the uncertainty level of the rule-head. More precisely:

Definition 1. A fDATALOG rule is a triplet r;β;I, where r is a formula of the form

$$A \leftarrow A_1,...,A_n \quad (n \geq 0).$$

A is an atom (the head of the rule), $A_1,...,A_n$ are literals (the body of the rule); I is an implication operator and $\beta \in (0,1]$ (the level of the rule).

For getting finite result, all the rules in the program must be safe.
A fDATALOG rule is safe if all variables occurring in the head also occur in the body, and all variables occurring in a negative literal also occur in a positive one. A fDATALOG program is a finite set of safe fDATALOG rules. There is a special type of rule, called fact. A fact has the form $A \leftarrow ;\beta;I$. From now on, we refer to facts as (A,β), because according to implication I, the level of A easily can be computed.

The semantics of fDATALOG is defined as the fixpoints of consequence transformations. Depending on these transformations we can define two kind of semantics for fDATALOG. The deterministic semantics is the least fixpoint of deterministic transformation DT_P, the nondeterministic semantics is the least fixpoint of nondeterministic transformation NT_P. According to the deterministic transformation, the rules of a program are evaluated parallel, while in non-deterministic case the rules are considered independently one after another. These transformations are the following:

Definition 2. Let B_P the Herbrand base of the program P, and let $F(B_P)$ denote the set of all fuzzy sets over B_P. The consequence transformations DT_P: $F(B_P) \rightarrow F(B_P)$ and NT_P: $F(B_P) \rightarrow F(B_P)$ are defined as

$$DT_P(X) = \{\bigcup\{(A,\alpha_A)\}\} \cup X, \text{ and}$$
$$NT_P(X) = \{(A, \alpha_A)\} \cup X,$$

respectively, where

$$(A \leftarrow A_1,\ldots,A_n ; I ; \beta) \in \textit{ground}(P), (|A_i|,\alpha_{A_i}) \in X, 1 \le i \le n,$$
$$\alpha_A = \max(0,\min\{\gamma \mid I(\alpha_{body},\gamma) \ge \beta\}).$$

$|A_i|$ denotes the kernel of the literal A_i, (that is it is the ground atom A_i, if A_i is a positive literal, and $\neg A_i$, if A_i is negative), *ground*(P) is the ground instance of P.

It was proven, that starting from the set of facts, both DT_P and NT_P has fixpoints, which are the least fixpoints in the case of positive (negation-free) P. These fixpoints are denoted by $lfp(DT_P)$ and $lfp(NT_P)$. It was also proved, that $lfp(DT_P)$ and $lfp(NT_P)$ are models of P, so we could define $lfp(DT_P)$ as the deterministic semantics and $lfp(NT_P)$ as the nondeterministic semantics of fDATALOG programs.

For negation-free fDATALOG the two semantics are the same, but they are different if the program has any negation. In this case the set $lfp(DT_P)$ is not always a minimal model, but the nondeterministic semantics – $lfp(NT_P)$ – is minimal under certain conditions. This condition is the stratification. Stratification gives an evaluating sequence in which the negative literals are evaluated first. Not all of the programs are stratifiable, but it was proved, that in the case of stratified programs, in the order of stratification, the least fixpoint of nondeterministic transformation is the minimal model of the program.

Example 1. Consider the fDATALOG program:

1. (r(a); 0.8)
2. p(x) ← r(x),¬q(x); I_1; 0.6
3. q(x) ← r(x); I_1; 0.5
4. p(x) ← q(x); I_1; 0.8

For simplicity let all of the implication operators be the Gödelian:

$$I(\alpha,\beta) = \begin{cases} 1 \text{ if } \alpha \le \beta, \\ \beta \text{ otherwise,} \end{cases}$$

The stratification is {r,q}, {p}, so the evaluation order is: 1., 3., 2., 4. (Precisely: firstly 1. and 3. in arbitrary order, then 2. and 4. in arbitrary order.) Then

$lfp(NT_P) = \{(r(a),0.8);\ (p(a),0.5);\ (q(a),0.5)\}.$

Evaluation Strategies

A fDATALOG program can be evaluated according to different strategies. The *bottom-up evaluation* starts from the facts, applies the rules, so it can deduce all computable facts. It is a simple iteration, possibly with many superfluous computing. There was given a modified bottom-up evaluation, which maybe can reduce the number of iterating steps.

In many cases however there is no need for all facts of the fixpoint, we want to get answer only for a concrete question. If a goal is specified together with a fDATALOG program, it is enough to consider only the rules and facts which are necessary to reach the goal. The *top-down evaluation* starts from the goal, and applies the suitable rules to reach the given facts of the program. Generally this kind of evaluations works through sub-queries. This means, that there are selected all possible rules, whose head can be unify with the given goal, and the atoms of the body are considered as new sub-goals. This procedure continues until obtaining the facts.

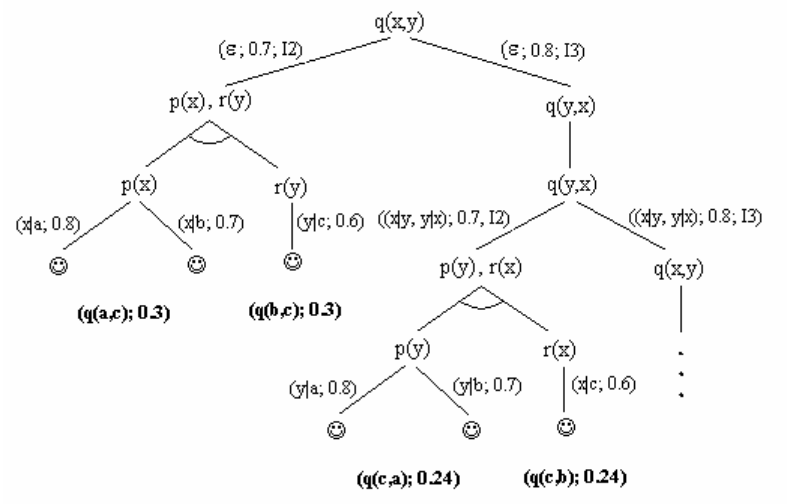
In the case of fuzzy Datalog, the evaluation doesn't terminate obtaining the facts, because we need to determine the uncertainty level of the goal. The evaluation is executed by the aid of evaluation tree. This is a special hyper-graph, based on the AND/OR tree concept. The root of the tree is the goal-atom, every odd edge of this tree is an n-order hyper-edge with the set-node of n elements, and every even edge is an ordinary edge with one node. On every even level of the graph there are sub-goals, which are suitably unified rule-heads, and on every odd level there are rule-bodies. The leaves of the tree are the symbols YES or NO: if the sub-goal is unified with a fact, than there is YES, else there is NO.

The ordinary edges of this graph are labelled: this label contains the applied unification, the uncertainty level and the implication operator of the suitable rule, represented by this edge. The answer can be obtain from the labels being on the hyper-path ending in symbol YES: starting from the leaves, going backward to the root, the uncertainty levels can be calculated from these labels.

Example 2. Consider the next rules:

```
(p(a); 0.8).
(p(b); 0.7).
(r(c); 0.6).
q(x,y)  ← p(x),r(y); 0.7; I2.
q(x,y)  ← q(y,x); 0.8; I3.
s(x)    ← q(x,y); 0.9; I3.
```

We want to get an answer for query (q(x,y); α).



According the above AND/OR tree, the answer is:

{(q(a,c), 0.3) , (q(b,c), 0.3) , (q(c,a), 0.24) , (q(c,b), 0.24)}.

It was proven, that in the case of final evaluation graph, the top-down evaluation gives the same answer as the fixpoint-algorithm. However it can be happen, that the top-down algorithm doesn't terminate. It was proven also, that in such case there is a depth limit, according with restricting the evaluation, it stops, and gives all answers for the query. This algorithm can be improved by some heuristics.

As in the case of Gödelian implication operator the uncertainty level of the rule-head can be computed in the same way as the uncertainty level of the rule-body, therefore the above top-down evaluation can be simplified if all implication operators of the program are Gödelian. This evaluation algorithm is also given in my dissertation.

Connection between fDATALOG and fuzzy relations

The ordinary Datalog program P may be interpreted also by relations. To every predicate of P corresponds a relation so, that an instance of the database predicate is true if and only if the corresponding relation has that fact as a tuple. Each rule of a Datalog program can be translated into an equation of relational algebra, and the fixpoint of these equations forms a model of P.

Similarly to this, it is possible to associate relations with fDATALOG predicates. The problem is, that in the literature there are different definitions of fuzzy relations. In my dissertation there was examined the equivalence of relational algebraic and logical evaluation of negation-free or stratified fDATALOG programs. To do this, we were supported by the concept of first type fuzzy relation:

Definition 3. A first type fuzzy relation R in D_1, ..., D_n is characterised by the membership function: $\mu_R : D_1 \times ... \times D_n \rightarrow [0,1]$

Example 3. $(F(X,Y), \mu_B)$ is a first type fuzzy relation, (e.g. friendship):

F:

X	Y	μ_B
John	Paul	0.1
Martha	Eve	0.6

It is obvious, that to each predicate r of a fDATALOG program P, there is a first type fuzzy relation R, so, that
$\underline{t} = (a_1, ...,a_n,\mu_R(\underline{t})) \in (R(A_1, ...,A_n), \mu_R) \Leftrightarrow (r(a_1, ...,a_n),\mu_R(\underline{t}))\in lfp(DT_P)$ (or $lfp(NT_P)$).
(A_1,..., A_n are the attributes of R.)

It was proven, that the deterministic or nondeterministic semantics of a fDATALOG program is equivalent with the solution of the suitable system of equations in relational algebra, even if the case of stratified programs.

Extension of fDATALOG to Fuzzy Data

The first type fuzzy relations show the closeness of the connection among crisp data. However, sometimes we need to use fuzzy data. So in the seventh chapter there was defined the second type fuzzy relation and was given a possible extension of fDATALOG on these relations.

Definition 4. Let D_1 ,..., D_n be n universal sets and $F(D_1)$, ..., $F(D_n)$ $1 \le i \le n$ theirs fuzzy sets. Then a second type fuzzy relation R is defined by the membership function: $\mu_R : F(D_1) \times ... \times F(D_n) \rightarrow [0,1]$

Example 4.

R:

Name	Age	Salary	μ_R
John	31	{(70000, 0.8), (75000, 0.7)}	0.7
Tom	middle aged	82000	0.8
Ann	young	{(60000, 0.6), (70000, 0.8)}	0.9

R is a second type fuzzy relation. The meaning of the third row for example is: we are almost sure, that Ann is young, and her salary is maybe 80-, possibly around 85 thousand Ft.

Our aim is to extend the fDATALOG programs so, that the predicates of the programs can be related to second type fuzzy relations. Therefore the fuzzy data are allowed. Formally a fDATALOG rule is the same as in the earlier chapters, but the unification of fuzzy data would cause difficulties. To avoid this, the rules are completed by proximity predicates, which show the closeness of the program's data. So the unification is crisp and the uncertainty is expressed by the proximity.

Definition 5. A proximity on a domain D is a fuzzy subset
$prox_D$: $D \times D \rightarrow [0,1]$ of $D \times D$ such that the following properties hold:

$prox_D(x,x) = 1$ for any $x \in D$ (reflexivity)

$prox_D(x,y) = prox_D(y,x)$ for any $x,y \in D$ (symmetry).

If a proximity is transitive, that is $S_D(x,z) \geq \min(S_D(x,y), S_D(y,z))$ for any $x,y,z \in D$, then it is called similarity.

Adding the facts according to proximity to the fDATALOG rules, we get more complicated program. I gave an algorithm for rewriting the rules according to proximity, and it was proven, that the evaluation can be simplified in the case of similarity.

Example 5. For the data of previous example, let us correspond to the relation R the predicate "person" with arguments of R's tuples. Consider the next rule:

```
head(x) ← person(x,about_40,about_80); 0.8; I.
```

The rewritten rule is

```
head(x) ← person(x1,y1,z1), prox1(x,x1),
      prox2(about_40,y1), prox3(about_80,z1); 0.8; I.
```

Of course it is necessary to define the above proximity predicates, for example $prox_2$(about_40, middle aged)= 0.9, etc.

Example 6. Although the suggested extension is difficult, but we can deal with such kind of programs:

```
person('Aladár', thin, 35); 0.9.
person('Béla', average, 40); 0.8.
```

```
witness_build(thin); 0.6.
witness_age(38); 0.7.
witness_age(40); 0.8.
suspect(X) ←
person(X,Y,Z),witness_build(Y),witness_age(Z);0.8;I.
```

(It's meaning: The build and age of Aladár and Béla is roughly known. The eyewitnesses remember any age and any build. A person is fairly suspect, if his age and build is something like that the eyewitnesses saw.)
The proximity of the example's atoms can be characterized for example in the next way: In the case of personal names the identity is required. The proximity of ages is written by the matrix sz_k, the proximity of builds is written by the matrix sz_t:

sz_k	35	38	40
35	1	0.8	0.7
38	0.8	1	0.9
40	0.7	0.9	1

sz_t	sovány	átlagos	kövér
sovány	1	0.7	0.2
átlagos	0.7	1	0.6
kövér	0.2	0.6	1

Fuzzy knowledgebase

Continuing the concept of the previous part, there is a further extension of fDATALOG to a possible model of fuzzy knowledge-base. This is defined as the next quadruple:

Definition 6. A fuzzy knowledge-base (fKB) is a quadruple (Bk, P, Φ_P, mA), where Bk is a background knowledge, P is a fuzzy Datalog program, Φ_P is a decoding set of P and mA is any modifying (or connecting) algorithm.

To understand this definition we have to determine the necessary concepts: Based on proximity relation we can define a so called background knowledge, which contains all possible known synonyms of predicate symbols and data.

Definition 7. Let $d \in D$ any element of domain D. The proximity set of d is a fuzzy subset over D: $SD_d = \{(d_1, \lambda_1), (d_2, \lambda_2),\ldots, (d_n, \lambda_n)\}$, where $d_i \in D$ and $prox_D(d, d_i) = \lambda_i$ for i = 1,…n.
Let C be any set of ground terms and R any set of predicate symbols. Let SC and SR any proximity over C and R respectively. The background knowledge is the union of proximity sets of C and R:

$$Bk = \{SC_t \mid t \in C\} \cup \{SR_p \mid p \in R\}$$

The connecting algorithm connects the background knowledge with the program. This algorithm gives a modified fDATALOG program, which is the extension of the original one according to proximity. Evaluating this program, the resulting uncertainty levels are not yet the final ones; those can be computed from these levels and from the proximity values of actual predicates and theirs arguments. To do this, we need the concept of decoding functions. It is expectable, that in the case of identity the decoding functions should not change the original level, but in other cases the final level should be less or equal then the original level or then the proximity values. Furthermore we require the decoding function to be monotone increasing.

Definition 8.
A decoding function is an (n+2)-ary function :

$$\varphi(\alpha, \lambda, \lambda_1 ,\ldots, \lambda_n) : (0,1] \times (0,1] \times (0,1] \times \ldots \times (0,1] \rightarrow [0,1]$$

so that

$\varphi(\alpha, \lambda, \lambda_1 ,\ldots, \lambda_n) \leq \min(\alpha, \lambda, \lambda_1 ,\ldots, \lambda_n)$,
$\varphi(\alpha, 1, 1 ,\ldots, 1) = \alpha$, and
$\varphi(\alpha, \lambda, \lambda_1 ,\ldots, \lambda_n)$ is monotone increasing in all arguments.

It is worth mentioning that any triangular norm is suitable for decoding function, for example the above min and product operators are t-norms.
We have to order decoding functions to all – but at least to the head – predicates of the program. The set of decoding functions will be the decoding set of the program.

Two kind of modifying algorithm was defined. According the first one we modify the program and use the original consequence transfor-

mation for evaluation; in the second case a modified consequence transformation is applied for the original program. In both cases the least fixpoint of modified program is regarded as the consequence of the knowledgebase, denoted by C(Bk, P, Φ_P, mA), which is the set of facts and levels, deducible from knowledgebase.

The simple modification replaces each predicate symbol and term of the program by their proximity sets, then we can evaluate the modified program according to the original semantics. The resulting fixpoint contains the proximity sets, from which we have to decide the final ground terms. These terms form the fixpoint of modified program.

Definition 9. The least fixpoint of mP, modified according simple modification (mA1) is:

$$C(P, Bk, \Phi_P, mA1)=lfp(mP)=\bigcup\{(q(t_1, t_2, \ldots, t_n); \varphi_q (\alpha_q, \lambda_q, \lambda_{t_1},\ldots, \lambda_{t_n})) \mid$$
$$\forall(SR_q (SC_{t_1} , SC_{t_2} , \ldots, SC_{t_n}) ; \alpha) \in lfp(NT_{mP}),$$
$$(q , \lambda_q) \in SR_q, (t_i , \lambda_{t_i}) \in SC_{t_i} , 1 \le i \le n\}.$$

In the other case a modified consequence transformation is applied for the original program. To define the modified transformation's domain we have to extend P's Herbrand universe and Herbrand base with elements according to proximities. The modified consequence transformation is defined over this modified Herbrand base (mB_p), so we can deduce to the atoms being in the rule-heads and the atoms being similar to them. The consequence of knowledgebase connected by transformation modification (mA2) is the least fixpoint of the next transformation.

Definition 10. The modified consequence transformation
$mNT_P : F(mB_P) \rightarrow F(mB_P)$ is defined as

$$mNT_P(X)=\{(q(s_1,\ldots,s_n), \phi_p(\alpha, \lambda_q, \lambda_{s_1} ,\ldots, \lambda_{s_n})|$$
$$(q, \lambda_q) \in SR_p;\ (s_i, \lambda_{s_i}) \in SC_{t_i} , 1 \le i \le n\} \cup X,$$

where $(p(t_1,\ldots,t_n) \leftarrow A_1,\ldots,A_k ; \beta; I) \in$ *ground*(P),
$(|A_i|, \alpha_{A_i}) \in X , 1 \le i \le k, \alpha = \max(0,\min\{\gamma \mid I(\alpha_{body}, \gamma) \ge \beta\})$.
$|A_i|$ denotes the kernel of the literal A_i.

It was proven, that the fixpoint of modified programs contains the fixpoint of original program, moreover the fixpoint of the program according to mA2 is wider than the fixpoint according to mA1:

$$C(P, Bk, \Phi_P, mA1) \subseteq C(P, Bk, \Phi_P, mA2).$$

Example 7. Consider the next knowledgebase:

sz(x,y) ← jo(y), mu(x); 0.7; I.	(jo(BB), 0.9).
m(x,y) ← ke(x,y), ko(y); 0.7; I.	(mu(P), 0.8).
(k(V), 0.9).	(ko(B), 1).
(zk(M), 0.8).	(ko(K), 1).

$\phi_{sz} := \phi_{sz}(\alpha, \lambda_{sz}, \lambda_1, \lambda_2) := \min(\alpha, \lambda_{sz}, \lambda_1, \lambda_2)$
$\phi_m := \phi_m(\alpha, \lambda_m, \lambda_1, \lambda_2) := \min(\alpha, \lambda_m, (\lambda_1 \cdot \lambda_2))$
$\phi_k := \phi_k(\alpha, \lambda_k, \lambda) := \alpha \cdot \lambda_k \cdot \lambda$
$\phi_{zk} := \phi_{zk}(\alpha, \lambda_{zk}, \lambda) := \min(\alpha, \lambda_{zk}, \lambda)$
$\phi_{jo} := \phi_{jo}(\alpha, \lambda_{jo}, \lambda) := \min(\alpha, \lambda_{jo}, \lambda)$
$\phi_{mu} := \phi_{mu}(\alpha, \lambda_{mu}, \lambda) := \min(\alpha, \lambda_{mu}, \lambda)$
$\phi_{ko} := \phi_{ko}(\alpha, \lambda_{ko}, \lambda) := \min(\alpha, \lambda_{ko})$, ha $\lambda=1$ 0 ha $\lambda<1$

$$I(\alpha, \beta) = \begin{cases} 1 & \text{ha } \alpha \le \beta, \\ \beta & \text{egyébként} \end{cases}$$

	B	V	H	BB	K	P	M
B	1	0.9	0.7				
V	0.9	1	0.6				
H	0.7	0.6	1				
BB				1	0.8		
K				0.8	1		
P						1	
M							1

	sz	ke	jo	k	mu	zk	ko	m
sz	1	0.8						
ke	0.8	1						
jo			1	0.75				
k			0.75	1				
mu					1	0.6		
zk					0.6	1		
ko							1	
m								1

Then
C(P, Bk, Φ_P, mA1) = {(k(V),0.9); (k(B),0.81); (k(H),0.54); (jo(V),0.675); (jo(B),0.6075); (jo(H),0.405); (zk(M),0.8); (mu(M),0.6); (jo(BB), 0.9); (jo(K),0.8); (k(BB),0.75); (k(K),0.75); (mu(P),0.8); (zk(P),0.6); (ko(B),1); (ko(K),1); (sz(P,BB),0.7); (sz(P,K),0.7); (ke(P,BB),0.7); (ke(P,K),0.7)}.

C(P, Bk, Φ_P, mA2) = {(k(V),0.9); (k(B),0.81); (k(H),0.54); (jo(V),0.675); (jo(B),0.6075); (jo(H),0.405); (zk(M),0.8); (mu(M),0.6); (jo(BB),0.9); (jo(K),0.8); (k(BB),0.75); (k(K),0.75); (mu(P),0.8); (zk(P),0.6); (ko(B),1); (ko(K),1); (sz(M,V),0.6); (sz(M,B),0.6); (sz(M,H),0.405); (sz(M,BB),0.6); (sz(M,K),0.6); (sz(P,V),0.675); (sz(P,B),0.6075); (sz(P,H),0.405); (sz(P,BB),0.7); (sz(P,K),0.7); (ke(M,V),0.6); ke(M,B),0.6), (ke(M,H),0.405); (ke(M,BB),0.6); ke(M,K),0.6); (ke(P,V),0.675); ke(P,B),0.6075); (ke(P,H),0.405); (ke(P,BB),0.7); (ke(P,K),0.7); (m(M,B),0.6); (m(M,K),0.6); (m(P,B),0.6075); (m(P,K),0.7)}.

Note: The above knowledge base could have the following interpretation. Let us suppose that music listeners (mu) "generally" (level 0.7) are fond of (sz) the greatest composers (jo). If there is a concert (ko) from the works of a composer, then his devotee (ke) generally (level 0.7) goes (m) to the concert. Assume furthermore that Mary is a "rather devoted" (level 0.8) fan of classical music (zk), and Vivaldi (V) is "generally accepted" (level 0.9) as a "great composer" (k). It is also widely accepted that the music of Vivaldi, Bach (B) and Handel (H) are fairly "similar", being related in overall structure and style. A further information, that Bartók (BB) is a great composer, and his style is similar to Kodály (K). There are two concerts, one from the works of Bach, the other from the works of Kodály. On the basis of the above information, how strongly state that Mary likes Bach? Or how sure, that Peter, who is a musician will listen the concert of Kodály? Our experience is: according to the algorithm mA1 we can't reply to these questions, but there are answers in the case of algorithm mA2.

I developed top-down evaluation for both kind of knowledgebase.

In the knowledgebase connected according to algorithm mA1, the modified program, mP has the same structure than the original one, so in the case of a given goal, mP also can be evaluated in top-down manner. To do this, using the similarity sets, we have to modify the original goal $(q(x_1, x_2,\ldots, x_n); \alpha)$, to the modified goal $(SR_q(SC_{x_1},SC_{x_2},\ldots,SC_{x_n});\alpha)$. For this query we get an answer according to the above procedure. Applying the suitable decoding functions we can decide an answer-set, which contains the required answer.

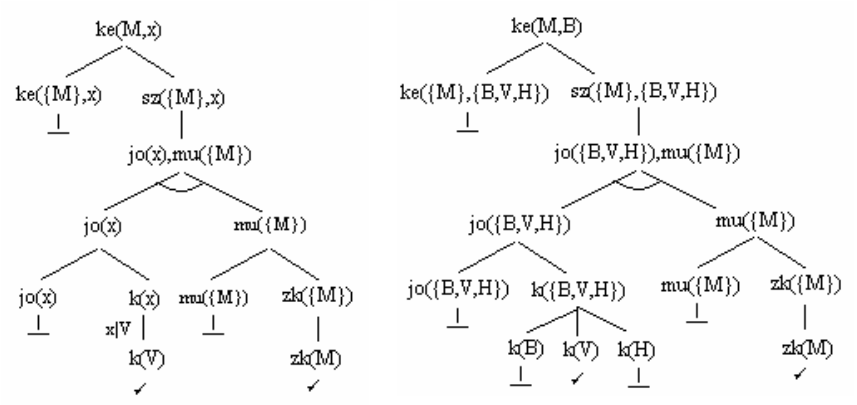
The knowledgebase connected according to algorithm mA2 is based on more complicated transformation, and its consequence is wider then the consequence of the first kind of knowledgebase. Therefore in this case the top-down evaluation is more desirable. Recently the pure top-down evaluation is not solved yet, because to do this, we have to solve the problem of fuzzy unification and the inversion of decoding functions. Similarly to the evaluation of ordinary fuzzy Datalog, our evaluation will have two direction, a top-down and a bottom up turn, moreover we rather rely on the bottom up evaluation, but the selection of required starting facts takes place in a top-down way.
Thus the aim of the further procedure is to decide the required starting facts, from which we can reply for the query. As only these facts are searched, therefore in the top-down part of the evaluation there are no need for the uncertainty levels, so we search only among the ordinary facts and rules.

An AND/OR tree arise during the evaluation, this is the searching tree. Its root is the goal; its leaves are one of YES or NO. The parent-nodes of YES are the required starting facts. This tree is build up by alternation of similarity-based and rule-based unification.

The rule-based unification unifies the sub-goals with the head of suitable rules, and continues the evaluating by the bodies of these rules. This unification is special in the sense, that during this unification a constant can be substitute with its similarity set, and the similarity sets of terms behave as ordinary constants.

The similarity-based unification unifies the predicate symbols of sub-goals by the members of its similarity set, excepting the first and last unification. The first similarity-based unification unifies the ground terms of the goal with their similarity sets, and the last one unifies the similarity sets among the parameters of resulting facts with their members.

Example 8. Let us complete the knowledgebase of the previous example by the goal ke(M,x), and the goal ke(M,B), where x is a variable, M and B are constants. The searching graph of the query is:



Starting from the set X_0 = {(fv(V), 0.9), (mf(M), 0.8)}, we get a fixpoint which contains the answer for our question, and this fixpoint is tighter than the full consequence of knowledgebase.

Summary

In my dissertation there was discussed some model for handling uncertain information. For this reason there was introduced the concept of fuzzy Datalog by completing the Datalog-rules and facts by an uncertainty level and an implication operator. The semantics of fuzzy Datalog was defined as a fixpoint-semantics. I gave an extension of fuzzy Datalog to fuzzy data based on the proximity

relation. So there was introduced the concept of a possible fuzzy knowledgebase. I also have presented some possible evaluation strategy.

After finishing the dissertation there are remaining questions: which kind of other modifying algorithms may be developed; which kind of other possible property are necessary to order to the decoding function; how can improve the efficiency of evaluation algorithms; how can give a good structure for background knowledge; etc. Maybe these questions will be the subject of further investigations.

Publikációk – Publications

Referált cikkek – Referred articles:

[1] Ágnes Achs - Attila Kiss: Fixpoint query in fuzzy Datalog, Annales Univ. Sci. Budapest, Sect. Comp. 15. (1995.), (223-231.)

[2] Ágnes Achs - Attila Kiss: Fuzzy extension of Datalog, Acta Cybernetica 12 (1995.), Szeged, (153-166.)

[3] Achs Ágnes - Kiss Attila: A Datalog fuzzy kiterjesztése, Alkalmazott Matematikai Lapok 19 (1998.), (111-138.)

[4] Ágnes Achs:Evaluation Strategies of Fuzzy Datalog, Acta Cybernetica 13 (1997.), Szeged, (85-102.)

[5] Ágnes Achs: Fuzzy Datalog with Background Knowledge, Teaching Mathematics and Computer Science, 3(2005)2 Debrecen, (257-283)

Referált konferenciakiadványok – Referred conference proceedings:

[6] Ágnes Achs: Fuzzy knowledge-base with fuzzy Datalog – a model for handling uncertain information, IEEE 4th International Conference on Intelligent System Design and Application, Budapest, 2004. aug. 26-28, (55-60.)

[7] Ágnes Achs: Computed answer based on fuzzy knowledgebase – a model for handling uncertain information EUSFLAT-LFA 2005 – Fourth Conference of the European Society of Fuzzy Logic and Technology, September 7-9. 2005., Barcelona, (94-99.)

Tankönyv, jegyzet – Textbook, note:

[8] Achs-Fekete-Sárvári-Tóth: Matematikai feladatgyűjtemény
(főiskolai jegyzet) PMMF Mat. és Szám.tech. Int., 1979., 268 o.

[9] Achs-Fekete: Tanulásirányító útmutató a matematikához
(főiskolai jegyzet) PMMF Mat. és Szám.tech. Int.,, 1979., 54 o.

[10] Achs Ágnes: Numerikus matematika
(főiskolai jegyzet) PMMF Mat. és Szám.tech. Int., 1981., 148 o.

[11] Achs-Fekete: Tanulásirányító útmutató a matematikához
(főiskolai jegyzet) PMMF Mat. és Szám.tech. Int., 1982., 139 o.

[12] Achs-Blázsovics: Számítástechnikai feladatgyüjtemény
(főiskolai jegyzet) PMMF Mat. és Szám.tech. Int., 1984., 109 o.

[13] Achs Ágnes: Sík és térgörbék approximációja
(főiskolai jegyzet), PMMF, 1988., 124 o.

[14] Achs Ágnes: Gyakorlati szövegszerkesztés
(főiskolai jegyzet), Krónika kiadó, Pécs, 2002., 66 o.

Elektronikus tananyagok – Electronical textbooks:

[15] Achs Ágnes: A mesterséges intelligencia alapjai
(html formátumú lektorált főiskolai jegyzet), 1997.

[16] Achs Ágnes: Java alapok
(html formátumú lektorált főiskolai jegyzet), 2000.

[17] Achs Ágnes: A Prolog nyelv alapjai
(html formátumú lektorált főiskolai jegyzet), 2001.

Konferencia előadások – Conference lectures:

[18] Achs Ágnes: CAD oktatás az informatikus képzésben
Műszaki és Tanárképző Főiskolák Mat.-, Szám.tech.-, Fizika Oktatóinak Országos Konferenciája, Szombathely, 1992.

[19] Achs Ágnes: Logikai adatmodellek
Informatika a Felsőoktatásban Konferencia, Debrecen, 1993.

[20] Achs Ágnes: A De Montfort University of Milton Keynes egyetem oktatási struktúrája, az egy hónapos Tempus ösztöndíj tapasztalatai
Főiskolák Fizika, Matematika és Számítástechnika oktatóinak XVIII. Országos Tanácskozása, Szeged, 1994. augusztus 22-24.

[22] Achs Ágnes: Bizonytalan információk kezelése logikai adatmodellekben
Főiskolák Fizika, Matematika és Számítástechnika oktatóinak XIX. Országos Tanácskozása, Pécs, 1995. szeptember

[23] Achs Ágnes: Bizonytalan információk kezelése logikai adatmodellekben
(előadás – az előzőtől eltérő tartalommal)
Informatika a Felsőoktatásban Konferencia, Debrecen, 1996.

[24] Achs Ágnes: Mennyire biztos a bizonytalan? - Gondolatok egy fuzzy tudásbázisról
(előadás + konferenciakiadvány)
Informatika a felsőoktatásban Konferncia, Debrecen, 2005., 6 o.

Egyéb írások – Other publications:

[25] Achs Ágnes: Tallózás folyóiratokban a számítástechnika és a társadalom kapcsolatának ürügyén.
Pollack Krónika, 1981. 1. sz.

[26] Achs Ágnes: Számítógép és grafika, sík és térgörbék konstruálása
PMMF, Tanulmányok 1986/2., 69 o.

[27] Számítástechnikai feladatok 2000-ig I-II.
Szerk. Dr. Hetényi Pálné, OMIKK, Budapest, 1988., 271 o.
(szerzői munkaközösség tagjaként)

[28] Achs-Vörös: CAD alkalmazás az informatikus képzésben
OTKA pályázat, 1989.

[29] Achs Ágnes: CAD oktatás az informatikus képzésben
PMMF tudományos közleményei 1990. 3. k., (62-71.).

Folyamatban lévő publikációk – Publications in process

[30] Achs Ágnes: Datalog alapú fuzzy tudásbázis – bizonytalanság kezelési modell
ígért megjelenés: 2006. május vége
Alkalmazott Matematikai lapok

[31] Achs Ágnes – Fazekas Gábor: The Role of Logic in Informatics Education – on the pretext of a logic textbook
elfogadva
Teaching Mathematics and Computere Science, Debrecen

[32] Ágnes Achs: Computed answer from uncertain knowledge – a model for handling uncertain information –
elküldve az EJOR számára

[33] Ágnes Achs: Creating and evaluating of fuzzy knowledgebase
felkérésre elküldve a JUCS számára.

[34] Ágnes Achs: Proximity based knowledge with evaluating algorithm
elküldve az INES 2006. konferenciára, Londonba