Alkalmazott Matematikai Lapok **24** (2007), 79-98.

DATALOG ALAPÚ FUZZY TUDÁSBÁZIS – BIZONYTALANSÁGKEZELÉSI MODELL –

ACHS ÁGNES

Tanulmányunkban egy lehetséges bizonytalanságkezelési modellt szeretnénk bemutatni. A fuzzy tudásbázist négyelemű halmazként definiáljuk, amelynek egyik eleme egy fuzzy következtetési rendszer, a fuzzy Datalog; másik eleme a háttértudás, amelyet a termek és predikátumok közötti hasonlóság segítségével adunk meg; harmadik eleme egy összekapcsolási algoritmus, amely összeköti a háttértudást és a következtetési mechanizmust; a negyedik eleme pedig a program dekódoló függvényeinek halmaza, amelyek segítségével meg tudjuk határozni az eredmény bizonytalansági szintjét. Dolgozatunkban bemutatunk lehetséges kiértékelési stratégiákat is.

1. Bevezetés

Az emberi tudás jó része bizonytalan, homályos, esetleg félreérthető vagy hiányos, emiatt nem minden részét lehet modellezni a kétértékű logikán alapuló következtetési rendszerekkel. A bizonytalan információk kezelésére számtalan megoldási javaslat készült, ezek közül most a fuzzy logikára épülő modelleket emeljük ki.

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 ([6], [1]). A fuzzy Datalog nyelvet „éles" adatokra definiáltuk, de később kiterjesztettük fuzzy adatokra is ([7]).

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. Szinte mindegyikük reflexív, szimmetrikus és tranzitív relációnak tekinti a hasonlóságot. Némelyikük a hasonlóság alapján ekvivalencia-osztályozást valósít meg, és ezeken az osztályokon értelmezi az egységesítést ([5], [16]). Virtanen nyelvi változókat definiál, és ezekkel valósítja meg a fuzzy illesztést ([19], [20]). Az előzőektől eltérő ötlet olvasható [17]-ben. Itt a szerzők az esetleges elírások

kezelésére dolgoznak ki egy számítási módszert, az elírásból adódó hibák számával definiálják a termek és predikátumok távolságát, és ezeknek a távolságnak a felhasználásával értelmezik a fuzzy egységesítést.

Dolgozatunkban szeretnénk folytatni korábbi elképzeléseinket, és a fuzzy Datalog további kiterjesztésén alapuló bizonytalanságkezelő modellt alkotunk. Mi is támaszkodunk a hasonlóság fogalmára, de nem követeljük meg annak tranzitív voltát.

Bár [6], [7], [8] részletesen tárgyalja a fuzzy Datalog fogalmát, a következőkben egy rövid összefoglalást adunk róla.

2. fuzzy Datalog

Egy Datalog program tényekből és szabályokból épül fel. A fuzzy Datalogban a tényeket bizonytalansági szintekkel, a szabályokat bizonytalansági szintekkel és implikációs operátorokkal egészítjük ki. Egy szabályfej bizonytalansági szintje a szabálytörzs bizonytalansági szintjéből és a szabály bizonytalansági szintjéből határozható meg az adott implikációs operátor segítségével.

Ha például a

$$\texttt{szép('Juliska')};\ 0,7.$$
$$\texttt{szereti('Jancsi',X)} \leftarrow \texttt{szép(X)};\ 0,8;\ \texttt{I}.$$

programban I a Gödel-féle implikációs operátort jelenti (vagyis $I(\alpha,\beta) = 1$, ha $\alpha \leq \beta$, $I(\alpha, \beta) = \beta$ egyébként), akkor a szabályfej bizonytalansági szintje a szabálytörzs és a szabály bizonytalansági szintjének minimumaként számolható ki. Esetünkben ez azt jelenti, hogy

$$\texttt{szereti('Jancsi','Juliska'); 0,7,}$$

azaz Jancsi legalább $0,7$ szinten szereti Juliskát.

A továbbiakban pontosan definiáljuk a fuzzy Datalog (fDatalog) program fogalmát:

2.1. Definíció. fDatalog szabály egy r; β; I hármas, ahol r egy

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

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

A fDatalog szabály biztonságos, ha

- a fejben előforduló összes változó szerepel a törzsben is;
- az összes olyan változó, amely negatív literálban szerepel, előfordul pozitív literálban is.

Egy fDatalog program biztonságos fDatalog szabályok véges halmaza. Az $A \leftarrow;\ \beta;\ I$ alakú szabályokat, ahol A alapatom, tényeknek nevezzük. A továbbiakban ezeket (A, β) módon jelöljük, hiszen a legtöbb implikációs operátor esetén a szabályfejhez tartozó bizonytalansági szint megegyezik a szabály bizonytalansági szintjével, de az ettől eltérő tulajdonságú implikációs operátorok esetén is könnyen kiszámolható.

Egy fDatalog programhoz fixpont-szemantikát rendeltünk. Ez alapján a szabálytörzs és a szabály bizonytalansági fokából $(\alpha_{\text{törzs}}, \beta)$ az implikációs operátor (I) alkalmazásával megállapíthatjuk, hogy legalább milyen szinten érvényes a szabályfej. Vagyis a fDatalog szemantikája rákövetkezési transzformációk fixpontjaként értelmezhető.

A rákövetkezési transzformációkat a program Herbrand-bázisának fuzzy részhalmazain értelmeztük. Mivel csak függvénymentes fDatalog programokkal foglalkoztunk, ezért esetünkben egy P program H_P-vel jelölt *Herbrand-univerzumán* a P-ben előforduló konstansokból képzett összes lehetséges alap-term halmazát értjük. A program B_P-vel jelölt *Herbrand-bázisa* pedig a P-ben előforduló predikátumszimbólumokból és H_P-beli alaptermekből képzett összes lehetséges alap-atom halmaza.

Egy szabály alap-előfordulásán egy olyan szabályt értünk, amelyet az eredeti szabályból a szabály változóinak H_P-beli elemekkel való helyettesítésével kapunk.

A program $\mathit{ground}(P)$-vel jelölt alap-előfordulásán összes szabálya összes lehetséges alap-előfordulásának halmazát értjük.

A P program egy interpretációja Herbrand-bázisának egy fuzzy részhalmaza:

$$\bigcup_{A \in B_P} (A, \alpha_A).$$

α_A-val az A atom igazságértékét jelöltük. A konjunkció és negáció igazságértékét a szokásos módon értelmeztük, vagyis:

$$\alpha_{A_1 \wedge \cdots \wedge A_n} = \min(\alpha_{A_1}, \ldots, \alpha_{A_n})$$
$$\alpha_{\neg A} = 1 - \alpha_A.$$

Egy interpretációt a P program modelljének tekintünk, ha

$$\text{minden } A \leftarrow A_1, \ldots, A_n;\ \beta;\ I \in \mathit{ground}(P) \text{ esetén}$$
$$I(\alpha_{\text{törzs}}, \alpha_{\text{fej}}) \geq \beta,$$

vagyis a szabálytörzs és a szabályfej bizonytalansági értékeiből képzett implikáció értéke nem kisebb a szabály bizonytalanságánál.

Kétfajta rákövetkezési transzformációt is definiáltunk, egy determinisztikust és egy nem determinisztikust. Belátható, hogy bizonyos feltételnek eleget tévő implikációs operátorok esetén mindkét transzformációnak van legkisebb fixpontja, amely egyúttal a program minimális modellje is. Ezeket tekintjük a fDatalog determinisztikus, illetve nemdeterminisztikus szemantikájának. Determinisztikus szemantika

alapján egy program szabályai párhuzamosan értékelhetők ki, nemdeterminisztikus esetben egymás után. Belátható, hogy függvény- és negációmentes program esetén a két szemantika megegyezik.

A két rákövetkezési transzformáció a következő:

2.2. Definíció. Legyen B_P a P program Herbrand-bázisa, és jelölje $F(B_P)$ a B_P fölötti fuzzy halmazok halmazát. A

$$DT_P : F(B_P) \to F(B_P) \quad \text{és} \quad NT_P : F(B_P) \to F(B_P)$$

rákövetkezési transzformációkat a következő módon értelmezzük:

$$DT_P(X) = \Big\{ \bigcup \{(A, \alpha_A)\} \,|(A \leftarrow A_1, \ldots, A_n; \beta; I) \in \mathit{ground}(P),\ (|A_i|, \alpha_{A_i}) \in X,$$
$$1 \le i \le n,\ \alpha_A = \max(0, \min\{\gamma | I(\alpha_{\text{törzs}}, \gamma) \ge \beta\}) \Big\} \bigcup X$$

és

$$NT_P(X) = \{(A, \alpha_A)\} \bigcup X,$$

ahol

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

$|A_i|$-val az A_i literál magját jelöljük (vagyis azt az atomot, amely vagy amelynek negáltja a literált alkotja).

[7]-ben bebizonyítottuk, hogy a tényhalmazból indulva mindkét rákövetkezési transzformációnak létezik fixpontja, mely pozitív P program esetén egyúttal a legkisebb fixpont. Ezeket $\mathit{lfp}(DT_P)$-vel, illetve $\mathit{lfp}(NT_P)$-vel jelöltük.

Azt is beláttuk, hogy $\mathit{lfp}(DT_P)$ és $\mathit{lfp}(NT_P)$ a P program modelljei, így $\mathit{lfp}(DT_P)$-t a fuzzy Datalog program determinisztikus szemantikájaként, $\mathit{lfp}(NT_P)$-t pedig nemdeterminisztikus szemantikaként értelmeztük.

Függvény- és negációmentes fDatalog esetén a két szemantika megegyezik, de negációt tartalmazó esetben eltérhetnek egymástól. Előfordulhat, hogy a determinisztikus fixpont nem minimális modell, viszont – bizonyos feltételek esetén – a nem determinisztikus fixpont minimalitása továbbra is garantálható. Ez a feltétel a rétegzés. A rétegzés meghatároz egy kiértékelési sorrendet, amelyben a negatív literálokat értékeljük ki először, s ily módon minimális modellt kapunk. (A fuzzy Datalog rétegzését részletesen tárgyalja [7].)

2.1. Példa. Adott a következő fDatalog program :

1. $(\mathtt{r(a)}, 0, 8)$.
2. $\mathtt{p(x)} \leftarrow \mathtt{r(x)},\ \neg\mathtt{q(x)};\ 0,6;\ \mathtt{I}$.
3. $\mathtt{q(x)} \leftarrow \mathtt{r(x)};\ 0,5;\ \mathtt{I}$.
4. $\mathtt{p(x)} \leftarrow \mathtt{q(x)};\ 0,8;\ \mathtt{I}$.

A rétegzési sorrend: $\{r, q\}$, $\{p\}$, vagyis a kiértékelési sorrend: 1., 3., 2., 4. (Pontosabban: először 1. és 3. tetszőleges sorrendben, majd 2. és 4. szintén tetszőleges sorrendben.)

A Gödel-féle implikációs operátor és nem-determinisztikus szemantika esetén:

$$lfp(NT_P) = \{(r(a), 0, 8);\ (p(a), 0, 5);\ (q(a), 0, 5)\}.$$

A továbbiakban csak nem-determinisztikus szemantikával foglalkozunk.

3. Háttértudás

Egy fuzzy Datalog program önmagában is leír valamilyen tudást, de sokszor mégsem ad választ a kérdésünkre. Gyakran előfordulhat ugyanis, hogy még egyéb információkra, háttértudásra is szükségünk van a válasz megtalálásához. Ebben a fejezetben a háttértudás egy lehetséges modelljét építjük fel. Ehhez a hasonlóság fogalmát vesszük alapul.

3.1. Definíció. Egy D tartomány fölötti hasonlóság a D egy olyan

$$S_D : D \times D \to [0, 1]$$

fuzzy részhalmaza, amelyre teljesülnek a következő tulajdonságok:

$$S_D(x, x) = 1 \text{ minden } x \in D \text{ esetén} \qquad \text{(reflexivitás)},$$
$$S_D(x, y) = S_D(y, x) \text{ minden } x, y \in D \text{ esetén} \qquad \text{(szimmetria)}.$$

Egy hasonlóság tranzitív, ha

$$S_D(x, z) \geq \max\{\min(S_D(x, y),\ S_D(y, z))\} \text{ minden } x,\ y,\ z \in D \text{ esetén}.$$

A hasonlóságoktól általában nem követeljük meg a tranzitivitást, ugyanakkor azonban egy hasonlóságot leíró mátrixról, vagyis egy olyan mátrixról, amelynek elemei a D elemei közötti hasonlóság mértékét adják meg, eldönthető, hogy tranzitív hasonlóságot ír-e le.

Legyen S egy hasonlósági mátrix. A hasonlóság akkor és csak akkor tranzitív, ha

$$S \geq S \cdot S,$$

ahol a mátrix-szorzásban az elemek szorzása helyett azok minimumát képezzük, összeadásuk helyett pedig az elemek maximumát.

Modellünkben a háttértudást a hasonlósági halmazok segítségével definiáljuk.

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

$$S_d = \{(d_1, \lambda_1), (d_2, \lambda_2), \ldots, (d_n, \lambda_n)\},$$
$$\text{ahol } d_i \in D \text{ és } S_D(d, d_i) = \lambda_i \text{ minden } i = 1, \ldots, n \text{ esetén}.$$

3.1. Példa. Könnyen belátható, hogy a következő hasonlósági mátrix tranzitív hasonlóságot ír le.

	a	b	c	d	e
a	1	0,7	0,8	0,7	0,8
b	0,7	1	0,7	0,9	0,7
c	0,8	0,7	1	0,7	0,8
d	0,7	0,9	0,7	1	0,7
e	0,8	0,7	0,8	0,7	1

A hasonlóság fogalmával definiálhatjuk tetszőleges termek és tetszőleges predikátumszimbólumok hasonlóságát, ily módon megalkothatjuk az úgynevezett háttértudást.

3.3. Definíció. Legyen C alaptermek, R pedig predikátumszimbólumok egy halmaza. Legyen SC és SR valamilyen hasonlóság C, illetve R fölött. Háttértudáson C és R hasonlósági halmazainak unióját értjük:

$$Bk = \{SC_t | t \in C\} \bigcup \{SR_p | p \in R\}.$$

4. Fuzzy tudásbázis

Két lépést már megtettünk a fuzzy tudásbázis felépítéséhez vezető úton: definiáltuk a fuzzy Datalog program és a háttértudás fogalmát. A következő lépés a két fogalom összekapcsolása, melyet az összekapcsolási algoritmus segítségével oldunk meg. Ennek eredményeként az eredeti program és a háttértudás figyelembevételével egy módosított fDatalog programot kapunk, amelyet ki tudunk értékelni. A kiértékelés során kapott bizonytalansági szint értékek azonban még nem adják meg a végső választ az eredeti kérdésre, hiszen a végeredmény bizonytalansági szintjét ezekből és a felhasznált hasonlóságok értékéből számíthatjuk ki. Erre szolgálnak a dekódoló függvények. Ezektől elvárhatjuk, hogy azonosság esetén ne változtassák meg a kapott szintet, egyébként pedig legföljebb akkora értéket adjanak, mint az eredeti bizonytalansági szint vagy a hasonlósági értékek. Azt is megköveteljük, hogy a dekódoló függvények minden változójukban monoton növekvők legyenek.

Ezek alapján a dekódoló függvény fogalmát a következő módon definiálhatjuk:

4.1. Definíció. Dekódoló függvényen egy olyan $(n+2)$-változós függvényt értünk, melyre

$$\varphi(\alpha, \lambda, \lambda_1, \ldots, \lambda_n) : (0,1] \times (0,1] \times (0,1] \times \cdots \times (0,1] \to [0,1]$$
$$\varphi(\alpha, \lambda, \lambda_1, \ldots, \lambda_n) \leq \min(\alpha, \lambda, \lambda_1, \ldots, \lambda_n),$$
$$\varphi(\alpha, 1, 1, \ldots, 1) = \alpha \text{ és}$$

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

4.1. Példa.

$$\varphi_1(\alpha, \lambda, \lambda_1, \ldots, \lambda_n) = \min(\alpha, \lambda, \lambda_1, \ldots, \lambda_n);$$
$$\varphi_2(\alpha, \lambda, \lambda_1, \ldots, \lambda_n) = \min(\alpha, \lambda, (\lambda_1 \ldots \lambda_n));$$
$$\varphi_3(\alpha, \lambda, \lambda_1, \ldots, \lambda_n) = \alpha \cdot \lambda \cdot \lambda_1 \cdots \lambda_n$$

dekódoló függvények.

A program összes predikátumához – de legalábbis a fej-predikátumokhoz – dekódoló függvényt kell rendelnünk. Ezek halmaza alkotja a program dekódoló halmazát.

4.2. Definíció. Legyen P egy fuzzy Datalog program, F_P a program funktorainak halmaza (funktornak nevezzük egy atom predikátumszimbólumának és argumentumszámának együttesét, azaz pl. $q(t_1, t_2, \ldots, t_n)$ esetén q/n-t). P dekódoló halmazán a

$$\Phi_P = \{\varphi_q(\alpha, \lambda, \lambda_1, \ldots, \lambda_n) | \forall\, q/n \in F_P\}$$

halmazt értjük.

Tisztáztuk a szükséges fogalmakat, értelmezzük tehát a fuzzy tudásbázist!

4.3. Definíció. Fuzzy tudásbázison egy négyelemű $fKB = \{P, Bk, \Phi_P, mA\}$ halmazt értünk, ahol P egy fuzzy Datalog program, Bk a háttértudás, Φ_P a P dekódoló halmaza, és mA valamilyen módosítási (vagy összekapcsoló) algoritmus.

4.4. Definíció. Legyen $\{P, Bk, \Phi_P, mA\}$ egy fuzzy tudásbázis. A tudásbázis következményén a módosítási algoritmus szerint meghatározott mP program legkisebb fixpontját, $lfp(mP)$-t értjük.

Jelölése: $C(P, Bk, \Phi_P, mA)$.

5. Módosítási algoritmusok

Különböző módosítási algoritmusokat definiálhatunk. Közülük tárgyal egyet-egyet [2], [3], illetve [4], a továbbiakban ezeket foglaljuk össze.

5.1. Egyszerű módosítás ($mA1$ algoritmus)

Legyen P egy fuzzy Datalog program, és Bk a háttértudás. Határozzuk meg az mP módosított fDatalog programot a következőképpen: helyettesítsük a P minden p predikátumát és minden $t \in H_P$ alap-termjét a hasonlósági halmazukkal, SR_p-vel, illetve SC_t-vel, és minden x változót az $X = \{x\}$ halmazzal.

A módosítás algoritmusa:

5.1. Algoritmus.

Procedure módosítás(P, mP)
    $mP := \emptyset$
    while nem(üres(P)) do
        $(r; \beta; I) :=$ a P első szabálya
        $(R; \beta; I) :=$ (helyettesített$(r); \beta; I)$
        $mP := mP \bigcup (R; \beta; I)$
        $P := P - \{(r; \beta; I)\}$
    endwhile
endprocedure

function helyettesített(r)
    $\mathrm{Pred}_r :=$ az r predikátumainak halmaza
    $\mathrm{Term}_r :=$ az r alap-termjeinek halmaza
    $\mathrm{Var}_r :=$ az r változóinak halmaza

    while nem(üres(Pred_r)) do
        $q := \mathrm{Pred}_r$ első predikátumszimbóluma
        $Q := SR_q$
        $\mathrm{Pred}_r := \mathrm{Pred}_r - \{q\}$
    endwhile
    while nem(üres(Term_r)) do
        $t := \mathrm{Term}_r$ első alap-termje
        $T := SC_t$
        $\mathrm{Term}_r := \mathrm{Term}_r - \{t\}$
    endwhile
    while nem(üres(Var_r)) do
        $x := \mathrm{Var}_r$ első változója
        $X := \{x\}$
        $\mathrm{Var}_r := \mathrm{Var}_r - \{x\}$
    endwhile
    return helyettesített(r)
endfunction

Az így kapott mP programot ugyanúgy értékeljük ki, mint egy közönséges fDatalog programot, de a kapott fixpont még a hasonlósági halmazokat tartalmazza, ezekből meg kell határoznunk az eredményként kapott alap-termeket. Ezek alkotják a módosított program fixpontját. Mivel $\mathit{lfp}(NT_{mP})$ legkisebb fixpont (ld. [6], [7]), ezért a tagjainak „kibontásával" kapott halmaz is legkisebb fixpont lesz.

5.1. Definíció. A módosított mP program legkisebb fixpontja:

$$\mathit{lfp}(mP) = \bigcup \{(q(t_1, t_2, \ldots, t_n); \varphi_q(\alpha_q, \lambda_q, \lambda_{t1}, \ldots, \lambda_{tn})) |$$
$$\forall (Q((T_1, T_2, \ldots, T_n); \alpha) \in \mathit{lfp}(NT_{mP}),\ (q, \lambda_q) \in Q,\ (t_i, \lambda_{t_i}) \in T_i,\ 1 \leq i \leq n\}.$$

Mivel

$$(q(t_1, t_2, \dots t_n); \alpha) \in \mathit{lfp}(NT_P) \text{ esetén}$$
$$(Q(T_1, T_2, \dots, T_n); \alpha) = (SR_q(SC_{t_1}, SC_{t_2}, \dots, SC_{t_n}); \alpha) \in \mathit{lfp}(NT_{mP}),$$
$$\text{és } \varphi_q(\alpha, 1, 1, \dots, 1) = \alpha,$$

ezért igaz a következő állítás:

5.1. Állítás.
$$\mathit{lfp}(NT_P) \subseteq C(P, Bk, \Phi_P, mA1).$$

5.1. Példa. Tekintsük az 1. 2. példa fDatalog programját, és egészítsük ki az alábbi háttértudással, dekódoló halmazzal!

(r(a), 0, 8).
p(x) ← r(x), ¬q(x); 0, 6; I.
q(x) ← r(x); 0, 5; I.
p(x) ← q(x); 0, 8; I.

	a	b
a	1	0, 8
b	0, 8	1

	p	q	r	s	t
p	1	0, 4			
q	0, 4	1			
r			1	0, 6	0, 7
s			0, 6	1	
t			0, 7		1

$$\varphi_p(\alpha, x, y) = \varphi_q(\alpha, x, y) = \min(\alpha, x, y);$$
$$\varphi_r(\alpha, x, y) = \alpha \cdot x \cdot y$$

1. 2. példa eredménye szerint

$$\begin{aligned}\mathit{lfp}(NT_{mP}) &= \{(R(A),\, 0,8);\ (P(A),\, 0,5);\ (Q(A),\, 0,5)\} = \\ &= \{(\{(r,\, 1), (s,\, 0,6), (t,\, 0,7)\}(\{(a,\, 1), (b,\, 0,8)\}),\, 0,8); \\ &\quad (\{(p,\, 1), (q,\, 0,4)\}(\{(a,\, 1), (b,\, 0,8)\}),\, 0,5); \\ &\quad (\{(q,\, 1), (p,\, 0,4)\}(\{(a,\, 1), (b,\, 0,8)\}),\, 0,5)\}.\end{aligned}$$

Innen

$$\begin{aligned}\mathit{lfp}(mP) = \{&(r(a),\, 0,8), (r(b),\, 0,64), (s(a),\, 0,48), (s(b),\, 0,384), (t(a),\, 0,56), \\ &(t(b),\, 0,448), (p(a),\, 0,5), (p(b),\, 0,5), (q(a),\, 0,5), (q(b),\, 0,5)\}.\end{aligned}$$

5.2. Transzformációs módosítás ($mA2$ algoritmus)

Az előző fejezetben tárgyalt módosítási algoritmus szerint a programot változtattuk meg, és a megváltoztatott programot az eredeti rákövetkezési transzformációval értékeltük ki. Mostani megközelítésünkben a programot változatlanul hagyjuk, és a kiértékelési transzformációt módosítjuk. Az így adódó programot most is mP-vel jelöljük.

Egy P program eredeti rákövetkezési transzformációját a program Herbrand-bázisának fuzzy részhalmazai fölött, vagyis $F(B_P)$-n értelmeztük. A transzformáció módosításához $F(B_P)$ kiterjesztésére van szükségünk. Egészítsük ki a program Herbrand-univerzumát a háttértudásban szereplő összes alap-termmel, az így kapott halmazt módosított Herbrand-univerzumnak nevezzük, és mH_P -val jelöljük. A módosított Herbrand-bázis, mB_P, az összes lehetséges olyan alap-atomot tartalmazza, melynek predikátumszimbóluma eleme a $P \bigcup Bk$ halmaznak, argumentumai pedig a módosított Herbrand-univerzum elemei. A transzformáció segítségével a szabályfejekben lévő atomokra és a hozzájuk hasonló atomokra is tudunk következtetni. Pontosabban:

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

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

ahol

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

($|A_i|$-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 ([4]).

A módosított program fixpontját, vagyis a tudásbázis következményét az mNT_P transzformáció fixpontjaként értelmezhetjük.

5.3. Definíció. A módosított mP program legkisebb fixpontja:

$$\mathit{lfp}(mP) = \mathit{lfp}(mNT_P).$$

A fenti konstrukció alapján nyilvánvaló, hogy az $mA2$ algoritmus esetén is fennáll az 5.1. állításban megfogalmazotthoz hasonló tartalmazási reláció:

5.2. Állítás.

$$\mathit{lfp}(NT_P) \subseteq C(P, Bk, \Phi_P, mA2).$$

5.2. Példa. Tekintsük ismét az 5.1. példát, és határozzuk meg a tudásbázis következményét az mNT_P transzformáció segítségével.

Induljunk ki az $X = \{(r(a), 0, 8)\}$ tényhalmazból. A kiértékelési sorrend most is legyen ugyanaz, mint a 2.1. példában, vagyis 1., 3., 2., 4. szabály.

A kiindulási halmaz a hasonlóság figyelembevételével a következőképpen alakul:

$$X = \{(r(a),\, 0,8), (r(b),\, 0,64), (s(a),\, 0,48),\\ (s(b),\, 0,384), (t(a),\, 0,56), (t(b),\, 0,448)\}.$$

A 3. szabály kibővíti ezt a halmazt a $\{(q(a),\, 0,5), (q(b),\, 0,5)\}$ halmazzal, majd az újabb elemekre is figyelembe véve a hasonlóságot, az alapatomok halmaza tovább bővül a $\{(p(a),\, 0,4), (p(b),\, 0,4)\}$ halmazzal.

A 2. szabály alapján a $\{(p(a),\, 0,5), (p(b),\, 0,5)\}$ atomokra következtethetünk. Mivel a további lépések már nem hoznak új eredményt, a két halmaz uniója a transzformáció legkisebb fixpontja, és egyúttal a tudásbázis következménye:

$$\mathit{lfp}(mP) = \{(r(a),\, 0,8), (r(b),\, 0,64), (s(a),\, 0,48), (s(b),\, 0,384), (t(a),\, 0,56),\\ (t(b),\, 0,448), (q(a),\, 0,5), (q(b),\, 0,5), (p(a),\, 0,5), (p(b),\, 0,5)\}.$$

Látható, hogy esetünkben a kétfajta módosítási algoritmus ugyanahhoz a következményhez vezetett. Ez azonban nincs mindig így, általában az $mA2$ algoritmussal definiált tudásbázis következménye bővebb a másiknál. A két konstrukció összehasonlításából nyilvánvalóan adódik a következő:

5.3. Állítás.

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

Ugyancsak egyszerűen belátható, hogy $\mathit{lfp}(mP)$ mindkét esetben a P modellje, de mivel $\mathit{lfp}(NT_P) \subseteq \mathit{lfp}(mP)$, ezért nem minimális modell.

5.3. Példa. Tekintsük a következő tudásbázist:

$lo(x,y) \leftarrow gc(y),\ mu(x);\ 0,7;\ I.$
$(fv(V),\, 0,9).$
$(mf(M),\, 0,8).$

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

	lo	li	gc	fv	mu	mf
lo	1	0,8				
li	0,8	1				
gc			1	0,75		
fv			0,75	1		
mu					1	0,6
mf					0,6	1

	B	V	M
B	1	0,9	
V	0,9	1	
M			1

$\Phi_{lo} := \Phi = \Phi(\alpha,x,y,z) := \min(\alpha,x,y,z)$
$\Phi_{fv} := \Theta = \Theta(\alpha,x,y) := \alpha \cdot x \cdot y$
$\Phi_{mf} := \omega = \omega(\alpha,x,y) := \min(\alpha, x \cdot y)$

Ekkor

$$C(P, Bk, \Phi_P, mA1) = \{(fv(V),\, 0,9); (fv(B),\, 0,81); (gc(V),\, 0,675);\\ (gc(B),\, 0,6075); (mf(M),\, 0,8); (mu(M),\, 0,6)\},$$

$$C(P, Bk, \Phi_P, mA2) = \{(fv(V),\, 0,9); (gc(V),\, 0,675); (fv(B),\, 0,81);\\ (gc(B),\, 0,6075); (mf(M),\, 0,8); (mu(M),\, 0,6);\\ (lo(M,V),\, 0,6); (lo(M,B),\, 0,6); (li(M,V),\, 0,6);\\ (li(M,B),\, 0,6)\}.$$

Megjegyzés. A fenti tudásbázishoz a következő „jelentés" rendelhető: tegyük fel, hogy a muzsikusok (mu) általában (vagyis $0,7$ szinten) szeretik (lo) a jó zeneszerzőket (gc).

Tudjuk, hogy Márta (M) eléggé ($0,8$ szinten) kedveli a zenét (mf). Azt is tudjuk, hogy Vivaldi (V) általában ($0,9$ szinten) kedvenc zeneszerző (fv). Azt is tudjuk, hogy Vivaldi és Bach (B) hasonló zeneszerzők. Vajon következtethetünk-e arra, hogy Márta mennyire kedveli (li) Bachot? Azt tapasztaltuk, hogy ha az $mA1$ algoritmussal kötjük össze a programot és a háttértudást, akkor erre nem tudunk következtetni, az $mA2$ algoritmussal viszont igen.

6. Kiértékelési stratégiák

Egy fuzzy tudásbázis következményét fixpont szemantikával értelmeztük. Ez azt jelenti, hogy a tényekből kiindulva a szabályok és a hasonlóság alkalmazásával következtettünk az összes előállítható atomra. Az ilyen fajta kiértékelést „bottom-up" következtetésnek nevezzük.

Sokszor előfordulhat azonban, hogy nincs szükség a teljes kiértékelésre, hiszen egy konkrét kérdésre keressük a választ, el akarjuk dönteni egy állítás igaz vagy hamis voltát, meg akarjuk állapítani bizonytalansági fokát. Ez azt jelenti, hogy egy cél ismeretében elég „célirányosan" végezni a kiértékelést, vagyis elég, ha csak a cél eléréséhez szükséges szabályokat, tényeket vesszük figyelembe. Azt a fajta kiértékelési stratégiát, amikor a célból kiindulva a megfelelő szabályok kiválasztásával a tények felé következtetve értékeljük ki a szabályokat, „top-down" kiértékelésnek nevezzük. A stratégia alkalmazhatóságához ki kell egészítenünk a tudásbázist a meghatározandó céllal (kérdéssel), vagyis egy $(q(t_1, t_2, \ldots t_n), \alpha)$ párral, ahol $q(t_1, t_2, \ldots t_n)$ atom, α pedig az atom bizonytalansági szintje. A q argumentumai között lehetnek változók is, és α is lehet változó vagy konstans. A továbbiakban mindkét típusú tudásbázis top-down kiértékelését tárgyaljuk.

6.1. Az $mA1$ algoritmussal összekapcsolt tudásbázis kiértékelése

A fuzzy Datalog top-down kiértékelési stratégiájával foglalkozik [1] és [8], fuzzy tudásbázisra való kibővítésével pedig [2] és [3]. Jelen tanulmányban a részletek mellőzésével adunk rövid összefoglalót az ott tárgyaltakról.

A top-down 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 kiértékelendő 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.

A most ismertetett eljárás kiterjeszthető rétegzett fDatalog programokra is.

Az $mA1$ algoritmussal összekapcsolt tudásbázisban a módosított program, mP, ugyanolyan szerkezetű, mint az eredeti fuzzy Datalog program, ezért egy cél ismeretében mP is kiértékelhető top-down módon. Ahhoz, hogy ezt megtegyük, a program módosításához hasonlóan, a hasonlósági halmazok felhasználásával alakítsuk át az adott $(q(x_1, x_2, \ldots x_n); \alpha)$ célt a módosított $(Q(X_1, X_2, \ldots X_n); \alpha)$ céllá. Az így keletkező kérdésre az előzőekben vázolt módon kapunk választ. Innen a dekódoló függvényeken alapuló eljárás segítségével meghatározhatunk egy válaszhalmazt, amely tartalmazza az eredeti kérdésre nyert választ is.

6.1. Algoritmus.

Procedure dekódolás $(Q, P, SR, SC, \Phi_P,$ Válaszok)
    $S :=$ a $(Q; \alpha)$ kérdésre adott válaszok halmaza
    Válaszok $:= \emptyset$
    while nem(üres(S)) do
        $(Q(T_1, T_2, \ldots T_n); \alpha) :=$ az S első eleme
        Válaszok $:=$ Válaszok $\bigcup$ dekódolt$(Q(T_1, T_2, \ldots T_n); \alpha)$
            */*az összes válasz dekódolása*/*
        $S := S - \{(Q(T_1, T_2, \ldots T_n); \alpha)\}$
    endwhile
endprocedure
function dekódolt$(Q(T_1, T_2, \ldots T_n); \alpha)$

```
    Dekódolt_halmaz := ∅
    while nem(üres(Q) do
        (q, λ_q) := Q első eleme
        q_set := ∅
        for all (t_i, λ_{t_ii}) ∈ T_i do
            q_set := q_set ⋃{(q(t_1, t_2, ... t_n); φ_q(α, λ_q, λ_{t_1}, ..., λ_{t_n}))
        endfor
        Dekódolt_halmaz := Dekódolt_halmaz ⋃ q_set
            /*a (Q(T_1, T_2, ... T_n); α) válasz összes „hasonmása"*/
        Q := Q − {(q, λ_q)}
    endwhile
    return Dekódolt_halmaz
endfunction
```

A fenti algoritmus alapján könnyen belátható a következő állítás:

6.1. Állítás. *Legyen Válaszok a 6.1. Algoritmus alapján meghatározott válaszhalmaz. Ekkor*

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

6.2. Az $mA2$ algoritmussal összekapcsolt tudásbázis kiértékelése

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.

A most megtárgyalandó eljárás célja tehát, hogy meghatározza a válasz megadásához szükséges kiindulási tényeket. Emiatt a továbbiakban – legalábbis a kiindulási halmaz meghatározásához – nincs szükségünk a bizonytalansági szintekre, ezért csak közönséges Datalog tényeket és szabályokat értékelünk ki, mégpedig „top-down" módon. Ehhez szükségünk lesz a helyettesítés és az egységesítés fogalmára, amelyet a továbbiakban ismertnek feltételezünk (a szükséges fogalmakat részletesen tárgyalja pl. [1], [10], [15], [18], stb.). Speciális helyettesítésekre is szükségünk van: időnként helyettesíteni kell egy p predikátumot vagy egy t termet a hasonlósági halmazával, S_p-vel, illetve S_t-vel, és időnként helyettesíteni kell egy hasonlósági halmazt az elemeivel.

Az egyszerűbb fogalmazás kedvéért ezentúl bizonytalansági szint nélkülinek tekintjük a célt, szabályokat, tényeket. A mostani kiértékelés során is felépítünk egy ÉS/VAGY gráfot, az úgynevezett keresési fát. Ennek gyökere a cél, levelei pedig az IGAZ/HAMIS szimbólumok. Az IGAZ levelek szülőcsúcsai 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 konstanst a hasonlósági halmazával helyettesítünk, illetve az argumentumokban már szereplő hasonlósági halmazok úgy viselkednek, mint a közönséges egységesítés konstansai.

A hasonlóság alapú egységesítés a részcél predikátumszimbólumát helyettesíti hasonlósági halmaza tagjaival. Az első és az utolsó hasonlóság alapú egységesítés eltér a többitől. Az első a cél alap-termjeit 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.

Az előzőek figyelembevételével a keresési fa a következőképpen épül fel: ha a cél szintjét 0 mélységűnek tekintjük, akkor minden $3k+2$ $(k = 0, 1, \dots)$ mélységben lévő csúcs rákövetkezői ÉS kapcsolatban állnak, a többiek VAGY kapcsolatban. Részletezve:

Induljunk ki a $g(t_1, t_2, \dots t_n)$ célból. Ennek rákövetkezője az összes lehetséges $g'(t'_1, t'_2, \dots t'_n)$ atom, ahol $g' \in S_g$; $t'_i = t_i$, ha t_i változó, és $t'_i = S_{t_i}$, ha t_i alap-term.

Ha a $p(t_1, t_2, \dots t_n)$ atom mélysége $3k$ $(k = 1, 2, \dots)$, akkor rákövetkezője az összes lehetséges $p'(t_1, t_2, \dots t_n)$ atom, ahol $p' \in S_p$.

Ha az L atom a $3k+1$ $(k = 1, 2, \dots)$ mélységben van, akkor a rákövetkező csúcspontokban vagy a vele egységesített szabály törzse szerepel, vagy egy vele egységesített tény, vagy a HAMIS szimbólum, ha L egyetlen szabályfejjel vagy ténnyel sem egységesíthető. Kicsit precízebben: ha az $M \leftarrow M_1, \dots, M_n$ $(n > 0)$ szabály feje egységesíthető L-lel, akkor az L rákövetkezője $M_1\theta, \dots, M_n\theta$, ahol θ az L és M legáltalánosabb egységesítője. Ha $L = p(t_1, t_2, \dots t_n)$, és a programban létezik p predikátumszimbólumú tény, akkor L rákövetkezője az összes lehetséges $p(t'_1, t'_2, \dots t'_n)$ atom, ahol $t'_i \in S_{t_i}$, ha $t_i = S_{t_i}$, vagy $t'_i = t_i\theta$, ha t_i változó, és θ a megfelelő illesztés.

Az előzőek alapján háromféle csúcspont lehet a $3k+2$ $(k = 1, 2, \dots)$ mélységű szinten: egy illesztett szabálytörzs; egy egységesített tény közönséges argumentumokkal vagy a HAMIS szimbólum. Az első esetben a rákövetkezők a törzs ÉS kapcsolatban lévő tagjai. Ha a törzs csak egyetlen literált tartalmaz, akkor csökkenthető lenne a kiértékelési út hossza, ez azonban az egységes tárgyalásmód rovására menne.

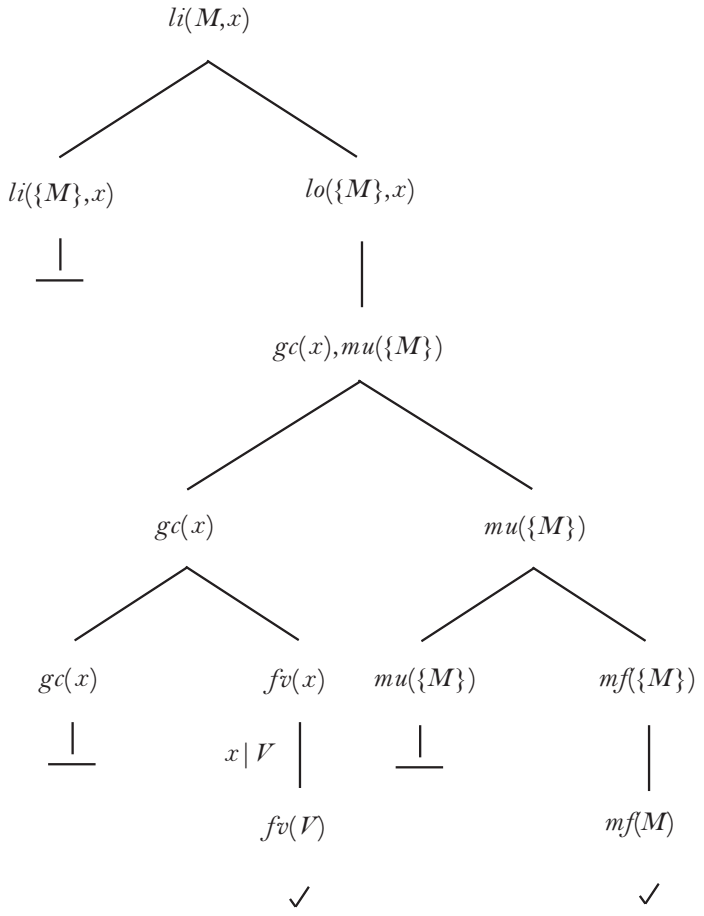
A második esetben a rákövetkezők az IGAZ/HAMIS szimbólumok valamelyike, attól függően, hogy az egységesített tény szerepel-e a program alap-atomjai között vagy sem. A HAMIS címkéjű csúcspontnak nincs utóda.

A keresési gráf konstrukciója alapján nyilvánvaló:

6.2. Állítás. *Legyen X_0 az IGAZ szimbólumok szülő-csúcsaiban lévő tények halmaza. X_0-ból kiindulva, az mNT_P transzformáció fixpontja tartalmazza a kérdésre adható választ.*

Természetesen elképzelhető, hogy ez a fixpont nemcsak a választ tartalmazza, hanem egyéb, fölösleges alap-atomot is, de általában kisebb, mint a tudásbázis teljes következménye. A keresési fa méretének csökkentése, és a fölösleges válaszok kiszűrése egy esetleges további kutatás eredménye lesz.

6.1. Példa. Egészítsük ki az 5.3. példa programját az $li(M, x)$ céllal, ahol x változó. A lekérdezés keresési gráfja:



1. ábra. A lekérdezés keresési gráfja

Az $X_0 = \{(fv(V), 0, 9), (mf(M), 0, 8)\}$ halmazból indulva most ugyanazt a fixpontot kapjuk, mint az 5.3. példában, de nyilvánvaló, hogy az így kapott fixpont általában szűkebb, mint a tudásbázis teljes következménye.

A fentiekben tárgyalt konstrukció a következő algoritmusban foglalható össze:

6.2. Algoritmus.

```
Procedure kiértékelés (g(t), Eredmények)                    /*g(t) a cél*/
      Fejek := {a programban szereplő szabályfejek}
      Tények := {a programban szereplő tények}
      Eredmények := ∅                                       /*az eredményül kapott kiindulási tények*/
      for all t ∈ t do
          if változó(t) then s := t
                        else s := St                        /*St a t hasonlósági halmaza */
          end_if
      end_for

      Csúcsok := {g(s)}                                     /*Csúcsok a kiértékelendő csúcsok halmaza, s az s elemekből álló vektor, az eredeti sorrendben*/
      Új_csúcsok := ∅                                       /*Csúcsok rákövetkezői*/
      while nem_üres(Csúcsok) do
          p(t) := eleme(Csúcsok)
          Spcsúcsok := ∅                                    /*p(t) rákövetkezői*/
          hasonlósági_kiértékelés(p(t), Spcsúcsok)
          Új_csúcsok := Új_csúcsok ⋃ Spcsúcsok
          Csúcsok := Csúcsok − {p(t)}
      end_while

      Csúcsok := új_csúcsok
      Új_csúcsok := ∅
      while nem_üres(Csúcsok) do
          p(t) := eleme(Csúcsok)
          Spcsúcsok := ∅                                    /*p(t) rákövetkezői*/
          szabály_kiértékelés(p(t), Spcsúcsok)
          Új_csúcsok := Új_csúcsok ⋃ Spcsúcsok
          Csúcsok := Csúcsok − {p(t)}
      end_while
end_procedure

procedure hasonlósági_kiértékelés(p(t), Spcsúcsok)
      for all q ∈ Sp do                                     /*Sp a p hasonlósági halmaza*/
          Spcsúcsok := Spcsúcsok ⋃ {q(t)}
      end_for
end_procedure
```

```
procedure szabály_kiértékelés(p(t), Spcsúcsok)
    for all p(v) ∈ Fejek do
        if egységesíthető(p(t), p(v)) then
          Spcsúcsok :=Spcsúcsok ⋃{a p(vθ) fejű      /*θ a megfelelő egysé-
                    szabály törzsének illesz-       gesítés*/
                    tett predikátumai }
        end_if
    end_for

    for all p(v) ∈ Tények do
        if egységesíthető(p(t), p(v)) then
          for all Stθ ∈ vθ do                       /*θ  a megfelelő egysé-
                                                    gesítés*/
            if változó(St) then
              t := Stτ                              /*τ a megfelelő egysé-
                                                    gesítés*/
            else if hasonlósági_halmaz(St) then
              t := eleme(St)
            end_if
          end_for
        end_if
        for all lehetséges t do                     /*t a t elemekből álló
                                                    vektor a megfelelő sor-
                                                    rendben*/
          if p(t) ∈ Tények then
            Eredmények := Eredmények ⋃{p(t)}
          end_if
        end_for
    end_for
end_procedure
```

Ugyanez az algoritmus rétegzett fDatalog program esetén is használható, ha a szabálytörzs rákövetkezőjének meghatározásakor figyelmen kívül hagyjuk a negációt.

7. Összesítés

Tanulmányunkban a bizonytalan információk kezeléséhez kívántunk lehetséges modellt nyújtani. Modellünket a deduktív adatbázis-kezelő nyelvre, a Datalogra és a fuzzy logikára alapozva képzeltük el. Ezért a fuzzy tudásbázist egy négyelemű halmazként értelmeztük, amelynek egyik eleme egy fuzzy következtetési rendszer, a fuzzy Datalog; másik eleme a háttértudás, amelyet a termek és predikátumok közötti hasonlóság segítségével adtunk meg; harmadik eleme egy összekapcsolási

algoritmus, amely összeköti a háttértudást és a következtetési mechanizmust; a negyedik eleme pedig a program dekódoló függvényeinek halmaza, amelyek segítségével meg tudjuk határozni az eredmény bizonytalansági szintjét. A tudásbázis következtetési rendszerét többféle elképzelés alapján kapcsolhatjuk össze a háttértudással, erre vonatkozóan két összekapcsolási algoritmust definiáltunk, és össze is hasonlítottuk a segítségükkel definiált tudásbázisok következményeit. Mindkét algoritmus esetén megadtunk egy top-down kiértékelési algoritmust, amely javítja a hatékonyságot. Különösen a második algoritmus esetén ez a hatékonyság még kívánnivalókat hagy maga után. A kiértékelés hatékonyságának javítása, illetve egy jobb, vagy az eddigiektől eltérő módosítási algoritmus megtalálása egy esetleges későbbi kutatás célja lehet.

Hivatkozások

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

[2] Ágnes Achs: *Fuzzy Datalog with background knowledge.* (Accepted in Teaching Mathematics and Computer Science, Debrecen)

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

[4] Ágnes Achs: *Computed answer from uncertain knowledge.*

[5] F. Arcelli – F. Formato – G. Gerla: *„Fuzzy Unification as Foundations of Fuzzy Logic programming"* in Logic Programming and Soft Computing, Ed. RSP-Wiley, England, 1998

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

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

[8] Achs Ágnes – Kiss Attila: *A Datalog fuzzy kiterjesztése.* Alkalmazott Matematika Lapok, Budapest, 1998, (111-138.)

[9] J. F. Baldwin – T. P. Martin: *Learning Uncertain Logic Programs from Examples.* Logic Programming and Soft Computing (LPSC98), Manchester

[10] S.Ceri – G.Gottlob – L.Tanca: *Logic Programming and Databases.* Springer-Verlag Berlin, 1990

[11] Didier Dubois – Henri Prade: *Fuzzy sets in approximate reasoning, Part 1.* Inference with possibility distributions, Fuzzy Sets and Systems **40**, 1991, (143-202.)

[12] Yuri Gurevich – Saharon Shelah: *Fixed-point extensions of first-order logic.* IEEEE Symp. on FOCS, 1985, (346-353.)

[13] J. W. Lloyd: *Foundations of Logic Programming* Springer-Verlag, Berlin, 1987

[14] Vilém Novák: *Fuzzy sets and their applications.* Adam Hilger Bristol and Philadelphia, 1989

[15] Pásztorné Varga Katalin: *A matematikai logika és alkalmazásai.* Tankönyvkiadó, Budapest, 1986

[16] Maria I. Sessa: *Approximate reasoning by similarity-based SLD resolution.* Theoretical Computer Science **275**, 2002, (389-426.)

[17] Michael Schroeder – Ralf Schweimeier: *Arguments and Misunderstandings: Fuzzy.* Unification for Negotiating Agents, Electronic Notes in Theoretical Computer Science, **Vol. 70** (5), 2002, Elsevier, Proceedings of the ICLP workshop CLIMA02, Copenhagen, Aug. 2002.

[18] J. D. Ullman: *Principles of database and knowledge-base systems.* Computer Science Press, Rockville, 1988

[19] Harry E. Virtanen: *Fuzzy unification.* 5th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, Paris, 1994. July 4-8

[20] Harry E. Virtanen: *Vague Domains, S-Unification and Logic Programming.* Electronic Notes in Theoretical Computer Science **66** (No. 5), 2002, URL: http://www.elsevier.nl/locate/entcs/volume66.html, 18 page

(Beérkezett: 2005. július 15.)

ACHS ÁGNES
PTE-PMMK
PÉCS, BOSZORKÁNY U. 2.
achs@witch.pmmf.hu

FUZZY KNOWLEDGE-BASE WITH FUZZY DATALOG

Ágnes Achs

In this paper there is given a possible model of handling uncertain information by defining fuzzy knowledge-base as a quadruple of a background-knowledge, a deduction mechanism, a decoding set and some modifying algorithm which connects the background knowledge to the deduction mechanism. We defined two kind of modifying algorithm, and gave evaluation strategies of them. Possibly this evaluation strategy can be improved, or maybe there is a better modifying algorithm. To find these better solutions will be the work of a possible further development.