



BIZONYTALANSÁGKEZELÉSI MODELLEK

Doktori (PhD) értekezés

Achs Ágnes

**Debreceni Egyetem
Informatikai Kar
Debrecen, 2006.**

Ezen értekezést a Debreceni Egyetem IK Matematika- és számítástudományok Doktori Iskola Informatika programja keretében készítettem a Debreceni Egyetem IK doktori (PhD) fokozatának elnyerése céljából.

Debrecen, 2006. március 27.

Tanúsítom, hogy Achs Ágnes doktorjelölt 2005-2006. között a fent megnevezett Doktori Iskola Informatika programja keretében irányításommal végezte munkáját. Az értekezésben foglalt eredményekhez a jelölt önálló alkotó tevékenységével meghatározóan hozzájárult. Az értekezés elfogadását javaslom.

Debrecen, 2006. március 27.



Köszönet

Szeretnék köszönetet mondani mindazoknak, akik munkámban segítettek. Először azoknak, akik elindítottak ezen az úton: az ELTE szakinformatikus képzésén oktató tanárainak. Tőlük nem csak a szakmai indíttatást kaptam, hanem meleg emberi hangot is, amely nélkül valószínűleg nem vágtam volna bele ebbe a sok nehézséggel és sok szépséggel járó tevékenységbe. Emberiségükből sok év távlatából még ma is erőt merítetek.

Ugyancsak megköszönöm a Debreceni Egyetem Informatika Karán velem foglalkozó oktatóknak, dolgozóknak a sok segítséget és bátorítást. Külön köszönetet szeretnék mondani Arató Mátyás professzor úrnak, akinek kedves biztatására folytattam újra a korábban elkezdett munkát.

Köszönetet mondok családomnak, barátaimnak is, akik állandóan arra nógattak, hogy a velük való törődés helyett a dolgozatommal foglalkozzam, és szüleimnek, akik már a túlpartról „fogják ceruzámat”.

Végül megköszönöm dolgozatom bírálóinak is a rám fordított energiát.



Tartalomjegyzék

BEVEZETÉS	1
1. TÖRTÉNETI ÁTTEKINTÉS	5
2. ELŐZMÉNYEK	9
<i>Datalog</i>	11
<i>Datalog</i> ⁷	12
<i>Bizonytalanságkezelési alapok – a háromértékű logika alkalmazása</i>	14
3. A DATALOG FUZZY KITERJESZTÉSE	19
3.1. A FUZZY DATALOG SZINTAKTIKÁJA	19
3.2. A FUZZY DATALOG SZEMANTIKÁJA	21
4. KIÉRTÉKELÉSI STRATÉGIÁK	31
4.1. ADATVEZÉRELT KIÉRTÉKELÉS	31
4.2. MÓDOSÍTOTT ADATVEZÉRELT KIÉRTÉKELÉS	33
4.3. MÓDOSÍTOTT ADATVEZÉRELT KIÉRTÉKELÉS RÉTEGZETT FDATALOG ESETÉN	34
4.4. CÉLVEZÉRELT KIÉRTÉKELÉS	34
4.5. CÉLVEZÉRELT KIÉRTÉKELÉS SPECIÁLIS IMPLIKÁCIÓS OPERÁTOR ESETÉN	48
4.6. CÉLVEZÉRELT KIÉRTÉKELÉSI HEURISZTIKÁK.....	54
4.7. AZ ÁLTALÁNOS CÉLVEZÉRELT KIÉRTÉKELÉS RÉTEGZETT FDATALOG ESETÉN.....	56
4.8. KOMBINÁLT KIÉRTÉKELÉS	57
5. A FDATALOG ÉS A FUZZY RELÁCIÓK KÖZÖTTI KAPCSOLAT	59
5.1. FUZZY RELÁCIÓS ALGEBRA.....	59
5.2. A FDATALOG RELÁCIÓS ALGEBRAI KIÉRTÉKELÉSE.....	63
5.3. A RÉTEGZETT FDATALOG RELÁCIÓS ALGEBRAI KIÉRTÉKELÉSE.....	70
6. A FDATALOG FUZZY ADATOKRA VALÓ KITERJESZTÉSE	71
6.1. KITERJESZTETT FDATALOG	71
6.2. TOP-DOWN KIÉRTÉKELÉS KITERJESZTETT FDATALOG PROGRAMOK ESETÉN	79
7. FUZZY TUDÁSBÁZIS	81
7.1. HÁTÉRTUDÁS.....	81
7.2. A TUDÁSBÁZIS	81

7.3. MÓDOSÍTÁSI ALGORITMUSOK	83
7.3.1. Egyszerű módosítás (<i>mA1</i> algoritmus).....	83
7.3.2. Transzformációs módosítás (<i>mA2</i> algoritmus).....	86
7.4. KIÉRTÉKELÉSI STRATÉGIÁK	89
7.4.1. Az <i>mA1</i> algoritmussal összekapcsolt tudásbázis kiértékelése	90
7.4.2. Az <i>mA2</i> algoritmussal összekapcsolt tudásbázis kiértékelése	91
8. KITEKINTÉS	97
ZÁRSZÓ.....	105
SUMMARY	107
THE FUZZY DATALOG	108
EXTENSION OF FDATALOG FOR FUZZY DATA	111
FUZZY KNOWLEDGEBASE	113
IRODALOMJEGYZÉK	115
MELLÉKLET.....	127
ELSŐRENDŰ LOGIKA.....	127
<i>Alapfogalmak</i>	127
<i>Formulák klóz alakja</i>	129
<i>Herbrand tétel</i>	130
<i>Hálóelméleti fixponttétel</i>	131
FUZZY ALAPOK.....	133
<i>A fuzzy halmaz fogalma</i>	133
<i>A fuzzy logika alapelemei</i>	138

Bevezetés

*„...most tükör által homályosan látunk,
most rész szerint van bennem az ismeret...”*

– Pál apostol; I. Korinthus 13:12.

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

– Rejtő Jenő: A boszorkánymester

Az emberi tudás jó része bizonytalan, homályos, esetleg félreérthető vagy hiányos, emiatt nem minden részét lehet modellezni a kétértékű logika „fekete-fehér”, „igaz-hamis” kijelentésein alapuló következtetési rendszerekkel. A bizonytalan információk kezelésére számtalan megoldási javaslat készült, dolgozatomban ezek közül a fuzzy logikára épülő modelleket emelem ki. Erre alapozva tárgyalom a deduktív adatbázis-kezelő nyelv, a Datalog fuzzy kiterjesztését, először csak „éles” adatokra „éles” predikátumokkal, majd további kiterjesztésként fuzzy adatokra, végül szeretnék eljutni a fuzzy Datalogon és a hasonlóságokon alapuló fuzzy tudásbázis fogalmáig. Fejezetekre bontva ez a következőket jelenti:

A disszertáció *első fejezetében* rövid történeti áttekintést nyújtok a logika és a logikai adatbázisok elméletének fejlődéséről. A fejezetben szereplő adatok forrása [D] és [P1], [P2].

A *második fejezetben* szeretném bemutatni, hogy dolgozatom témaköre hogyan illeszkedik a deduktív adatbázisnyelvek elméletébe, a relációs adatbázis fogalmának milyen irányú kiterjesztésével hozható kapcsolatba. Itt természetesen támaszkodom a témával kapcsolatos közismert irodalomra, elsősorban Ullman és az Abiteboul-Vianu szerzőpár munkásságára.

A további fejezetek nagymértékben saját kutatási eredményeket tartalmaznak.

A *harmadik fejezetben* a Datalog egy lehetséges általánosításáról, egy lehetséges fuzziifikálásáról lesz szó. A közönséges Datalog szabályokat kiegészítjük

egy bizonytalansági szintet jelző értékkel és egy implikációs operátorral, amely segítségével következtetni tudunk a szabályfej bizonytalansági szintjére. Ezek a bizonytalansági szintek azt mutatják meg, hogy az illető predikátum, illetve az illető szabály legalább mekkora szinten teljesül. A nagyobb általánosság kedvéért az implikációs operátorok szabályonként változhatnak, vagyis minden programhoz egy operátorhalmaz tartozik, s e halmaz elemei közül kerülnek ki a szabályokhoz rendelt operátorok. Az így értelmezett programnyelvet fDATALOG-nak nevezzük. A programnyelv elnevezése és szintaktikájának értelmezése Kiss Attilától származik ([K1]), szemantikájának megadása azonban saját munkám eredménye. Kétféle szemantikát – egy determinisztikus és egy nem determinisztikus szemantikát – értelmezek, mindkettőt egy-egy rákövetkezési transzformáció fixpontjaként. Belátható, hogy az így kapott szemantikák negációmentes esetben megegyeznek, s a program legkisebb modelljét adják. Negációt tartalmazó esetben megadható olyan nem determinisztikus kiértékelési sorrend, amelyben a szemantikát definiáló legkisebb fixpont a program minimális modelljét adja. Ez a kiértékelési sorrend a rétegzés.

A *negyedik fejezetben* az előzőekben értelmezett fDATALOG kiértékelésével foglalkozom. Ez fontos kérdés, ugyanis „ügyes” stratégiával hatékonyabban lehet elvégezni a következtetéseket. Kétféle stratégiát tárgyalok, a bottom-up és a top-down kiértékelést. Mindkét esetben megadok egy kiértékelési algoritmust is, valamint az egyszerűsítési lehetőségekről is szót ejtek. Az itt közölt módszerek teljes mértékben a saját munkám eredménye. Közöséges Datalog esetén természetesen vannak közismert kiértékelési eljárások ([CGT], [U]), ezek azonban legfőljebb kiindulásul szolgálhatnak, ugyanis minden esetben figyelembe kell venni a bizonytalansági szinteket és az implikációs operátorokat. Az egységes tárgyalásmód kedvéért ebben a fejezetben végig megmaradok a logikai fogalomrendszer keretei között, s csak a következő fejezetben térek rá a relációs algebrai tárgyalásra.

Az *ötödik fejezet* a fDATALOG és a fuzzy relációk közötti kapcsolattal foglalkozik. Itt adom meg a fuzzy relációk fogalmát, és ismertetem a fDATALOG relációsalgebrai kiértékelését. Ebben a fejezetben kapcsolódik össze Kiss Attila fDATALOG-ra vonatkozó kutatása ([K1]) az én munkámmal.

Az előzőekben értelmezett fDATALOG az úgynevezett első típusú fuzzy relációkkal hozható kapcsolatba. Ezek olyan relációk, amelyek azt mutatják

meg, milyen szoros kapcsolat áll fenn az attributumértékek között. Ezen relációk esetén az attributumértékek határozott adatok, nem tartalmaznak bizonytalanságot. Sokszor azonban maguk az értékek is bizonytalanok, vagyis számos esetben fuzzy adatok közötti relációra van szükségünk. Ezért válik szükségessé az úgynevezett második típusú fuzzy relációk bevezetése, melyek segítségével értelmezhető a fDATALOG egy lehetséges kiterjesztése.

Ezt tartalmazza a *hatodik fejezet*. Itt a fDATALOG fuzzy adatokra való egyik lehetséges kiterjesztését tárgyalom. Ez a fejezet saját ötletre és ennek megvalósítására irányuló munkára támaszkodik. Mivel nehézséget jelent a fuzzy adatok illesztése, ezért ezt a szomszédság fogalmának bevezetésével hidalom át. A szomszédsági predikátum beépítésével visszavezethető a probléma a negyedik fejezetben értelmezett fDATALOG szemantikák alkalmazására, s a bizonytalanság kezelése a szomszédsági predikátumokban valósul meg. Ily módon ragaszkodhatunk a szigorú illesztéshez, ugyanakkor fuzzy adatok kezelésére is lehetőségünk nyílik.

A *hetedik fejezet* egy további általánosítási irányt mutat be, a fuzzy tudásbázis fogalmát. Ezt egy négyelemű halmazként definiáljuk, amelynek egyik eleme egy fuzzy következtetési rendszer, a fuzzy Datalog; másik eleme a háttértudás, amelyet a termék és predikátumok közötti szomszédság segítségével adunk meg; harmadik eleme egy összekapcsolási algoritmus, amely összeköti a háttértudást és a következtetési mechanizmust; negyedik eleme pedig a program dekódoló függvényeinek halmaza, amelyek segítségével meg tudjuk határozni az eredmény bizonytalansági szintjét. A fejezetben lehetséges kiértékelési stratégiákat is mutatok.

Végül a *nyolcadik fejezetben* bemutatok néhány egyéb, az irodalomban olvasható eredményt.

A dolgozat témájának tárgyalásához szükséges alapvető logikai ismereteket a *mellékletben* (M) adom meg, felsorolva az idevágó fogalmakat, állításokat. Az adatbázisok elméletében fontos szerepet játszik a Herbrand modell és a hálóelméleti fixponttétel, melyeket szintén ebben a részben ismertetek. A bizonytalan információk kezelésének egy lehetséges elméleti háttéréről, a fuzzy halmazokról és a fuzzy logika fogalmáról, alapvető tulajdonságairól is itt ejtek szót.

1. Történeti áttekintés

A matematikai logika története több mint kétezer évre nyúlik vissza. Alapjait Arisztotelész (i.e.384-322) rakta le szillogisztikus következtetési elméletével. Munkája és – a ma már kevésbé ismert – társainak munkája a modern matematikai logikára is hatást gyakorol.

Az ókortól a tizenhetedik századig gyakorlatilag nem történt változás ezen a területen. Újabb irányzatok csak a XVI-XVII. században alakultak ki Bacon (1561-1626), Descartes (1596-1650) és Leibniz (1646-1716) munkája nyomán. Leibniz foglalkozott elsőként a formális logika fogalmaival. Bár érdemei elvitathatatlanok, a logika igazi fejlődése csak a XIX. században indult meg. Boole (1815-1864) „The Mathematical Analysis of Logic” című könyve volt az arisztotelészi logika átalakításában az első nagy lépés. Ez a munka az alapja a matematikai logika egyik irányzatának, a „logika matematikájának”, a Boole-algebrának. Boole kutatási irányzatához tartozott, és az ő művét fejlesztette tovább Jevons (1835-1882), Peirce (1839-1914), Schröder (1841-1902), Löwenheim (1878-1957).

A másik irányzat, a „matematika logikája” arra törekszik, hogy lehetővé tegye a matematika különböző ágaiban előforduló állítások formális leírását. A matematika axiomatizálásával kapcsolatban Frege (1848-1932), Cantor (1845-1914) és Peano (1858-1932) nevét kell megemlíteni.

A matematikai logika történetében fontos mérföldkő Whitehead (1861-1947) és Russel (1872-1970) „Principia Mathematica” című háromkötetes műve, amelyben egyesítik ezt a két irányzatot. Művük megjelenése után további komoly fejlődést jelentett Hilbert, Post, Ackerman, Gödel munkássága, amely a matematika egyes fejezeteinek ellentmondás-mentességének bizonyítására vonatkozott, és meghatározó volt a logika axiomatizálásában, vagy más szóval élve, bizonyításelméleti megalapozásában. A logika ugyanakkor nyelvészeti, azaz modelleméleti megközelítésben is tárgyalható, vagyis úgy, hogy egy adott szintaxissal és szemantikával rendelkező formális nyelvnek tekintjük. A modelleméleti és bizonyításelméleti megközelítés ekvivalenciájának bizonyítása egy sor algoritmuselméleti és bizonyításelméleti eredményre vezetett. (Church,

Kleene, Lukasewicz, Tarski, Kalmár) A matematikai logika fejlődése eredményezte olyan önálló tudományterületek kialakulását, mint a kiszámításelmélet (Boole, Jeffrey, Cohen), a rekurzív függvények elmélete (Goodstein, Kleene), automatikus tételbizonyítási elméletek (Chang, Lee, Gallier, Loveland, Prawitz). Ez utóbbi terület fejlődése a 30-as években kezdődött Gödel, Skolem, Church, Kleene, Turing, Herbrand, Löwenheim munkássága alapján.

Herbrand 1930-as alapvető tételére építve 1965-ben Robinson megalkotta a rezolúciós elvet, mely az automatikus tételbizonyítás alapja lett. A rezolúciós elv az egységesítés segítségével tud formulákból újabb formulára következtetni. Ez egy cáfoló eljárás, amely teljes abban az értelemben, hogy kielégíthetetlen formulahalmaz esetén mindig ellentmondásra vezet. Később sok változatát vezették be (Chang, Kowalski, Robinson, stb.), melyek a keresési tér csökkentésével növelik az eljárás hatékonyságát.

A rezolúciós elvre épül a logikai programozás elmélete (Deville, Elcock, Kowalski, Lloyd). Az elméletet a 70-es évek elején Colmerauer ültette át a gyakorlatba, létrehozva az első logikai programnyelvet, a PROLOG-ot. (PROgramming in LOGic). A nyelv megalkotásához az elsőrendű logika egy részhalmazát, a Horn-klózzokra épülő Horn-klóz logikát vették alapul. A logikai programozás procedurális interpretációjában minden Horn-klóz egy procedure-leírásnak tekinthető, és ezen procedure-k halmaza alkotja a programot. A „főprogram”-nak a célmegadás felel meg, a program az itt feltett kérdésre keresi meg a választ. A procedure-k az illesztés segítségével aktivizálódnak.

Egy programnyelv Horn-klózzokra való leszűkítése szűkíti az alkalmazhatóság területét is. Ezért a 80-as években élénk kutatás indult meg a logikai programozás körének kiszélesítésére. Vizsgálódások folytak a negatív premisszák bevezetésére vonatkozóan (Apt, Barbuti, Martelli, Lloyd, Clark, Chan, Reiter, Shepherdson, stb.); a deklaratív és procedurális szemantikák kidolgozására (Gelfond, Lifschitz, Kunen, Przymusiński, van Emden, Abiteboul, Vianu, stb.); a teljes elsőrendű logikára való kiszélesítés lehetőségeire (Lobo, Minker, Topor); magasabb rendű logikákra való kiszélesítésre (Maher, Mancarella, Pedreschi) és a logikai programozás különböző alkalmazási területeinek feltérképezésére. Egy további bővítési lehetőség a fuzzy logika alkalmazhatóságának vizsgálata, dolgozatomban is ezzel az iránnyal foglalkozik.

A logikai programozás specializálásának tekinthető a következő fejezetben bevezetendő deduktív adatbázisok témaköre. Ezek olyan logikai programok, amelyek a relációs adatbázisok fogalmát (Codd, Ullman, stb.) általánosítják. Szabályai általában kevésbé összetettek, mint egy általános célú logikai programé, és a függvények szerepe is jóval kisebb. Egy deduktív adatbázis struktúrája is eltérő, hiszen itt a szabályok száma jóval kevesebb, mint a tényeké.

2. Előzmények

A Codd által 1970-ben bevezetett relációs adatmodell segítségével sok, de nem minden probléma oldható meg. A standard relációs adatbázisnyelvek (pl. SQL) nem Turing-kiszámíthatóak, speciálisan nem tudják kezelni a rekurziót, így egy reláció tranzitív lezártját sem lehet meghatározni velük. Többek között ezért vált szükségessé az adatmodell kibővítése, illetve más jellegű modell megalkotása. A relációs kalkulusnál teljesebb a deduktív adatbázisok világa.

Deduktív adatbázisnak nevezzük a következtetési szabályokat is tartalmazó adatbázist. Az adatbázishoz sokszor még bizonyos feltételeket, az adatokra vonatkozó megszorításokat is szoktak fűzni. Bár az előző mondatok intuitíve elég jól meghatározzák a deduktív adatbázis fogalmát, egyes szerzők, mint pl. [GM], [Pr2] mégis igyekeznek precízebb definíciót megfogalmazni. Przymusinski deduktív adatbázison egy

$$D = \langle P, \text{SEM}(P), I \rangle$$

hármast ért, ahol P egy logikai program (adatbázis program), $\text{SEM}(P)$ a P deklaratív szemantikája és I az integritási megszorítások halmaza [Pr2]. Dolgozatomban nem kívánok kitérni a megszorítások tárgyalására, vagyis a továbbiakban deduktív adatbázison csak a logikai programnyelv – szemantika párost értjük.

Ha a lehető legáltalánosabban akarunk közelíteni az adatbázis-programok fogalmához, akkor egy program egy adatbázis-transzformációnak tekinthető, hiszen egy kiinduló adatbázishoz egy másik adatbázist rendel. Természetesen az adatbázis-transzformációknak eleget kell tenniük bizonyos feltételeknek. Ezeket a feltételeket elemzi részletesen Abiteboul és Vianu [AV1], [AV2], [AV3]. Említett tanulmányaikban két transzformációtípust – a determinisztikus és a nem determinisztikus transzformációt – és két programozási nyelvtípust – a procedurális és a deklaratív nyelvet – vizsgálnak. Részletesen és kimerítően tárgyalják az egyes nyelvtípusok kifejezőerejét, összhangba hozva őket egy Turing gép szalag-, illetve időbonyolultságával. Kimutatják, hogy a procedurális és deklaratív nyelvcsalád tagjai milyen kapcsolatban állnak egymással. Ezekre az elemzésekre nem akarok kitérni, de röviden szeretném ismertetni az

általuk definiált deklaratív nyelvcsaládot, melynek meghatározó tagját DL-nek (Declarative Language) nevezték.

2.1. Definíció: Egy DL program

$$A_1, \dots, A_k \leftarrow B_1, \dots, B_n \quad (k > 0, n \geq 0)$$

alakú szabályok halmaza, ahol minden A_i és minden B_i $(\neg)Q_i(x_1, \dots, x_m)$ alakú literál. Az A_1, \dots, A_k kifejezés a szabály feje, B_1, \dots, B_n a törzse.

Egy DL program működése során a szabályok ismételt alkalmazásával új tényekre következtetünk. Addig adunk új tényeket az adatbázishoz, amíg már nem tudunk továbbiakra következtetni, vagyis ameddig el nem jutunk egy fixponthoz. Nem determinisztikus esetben egyszerre csak egy szabályt alkalmazunk egy lehetséges kiértékeléssel, determinisztikus esetben párhuzamosan hajtjuk végre az összes szabályt az összes lehetséges kiértékeléssel. Ha egy szabály fejében van olyan változó, amely csak ott szerepel, akkor a szabálytörzs összes kiértékeléséhez ugyanazt az egy új értéket rendeljük, de a párhuzamosan végrehajtott szabályoknál ezeknek az újonnan bevezetett értékeknek különbözniük kell.

Megjegyzések:

1. Minden $A_1, \dots, A_k \leftarrow B_1, \dots, B_n$ szabályhoz hozzárendelhetjük a

$$\forall \underline{x} \exists \underline{y} (B_1 \wedge \dots \wedge B_n \rightarrow A_1 \wedge \dots \wedge A_k)$$

elsőrendű mondatot, ahol \underline{x} a törzsben előforduló változókat, \underline{y} pedig a csak fejben előfordulókat jelöli.

2. Az $A_1, \dots, A_k \leftarrow B_1, \dots, B_n$ szabályok szimulálhatóak k darab

$$A \leftarrow B_1, \dots, B_n$$

alakú szabállyal.

Ez az állítás determinisztikus esetben triviális, nem determinisztikus esetben nem olyan nyilvánvaló, de [AV3] szerint bizonyítható.

Mivel a DL nyelv megengedi olyan értékek létrehozását, amelyek nem szerepelnek az eredeti adatbázisban vagy a programban, ezért ezek a programok nem biztonságosak. A biztonságosság eléréséhez szintaktikus megszorításokat kell tennünk. A szerzők erős és gyöngé biztonságosságot definiálnak [AV3]. „Erős biztonságosság” fogalmuk megegyezik Ullman „biztonságosság” fogalmával [U].

Egy szabály biztonságos, ha a fejből előforduló összes változó szerepel a törzsben is. Ez a megszorítás azonban erősen korlátozza a nyelv kifejezőerejét.

A nyelvcsaládok kapcsolatát vizsgálva felmerül a szemantikák kapcsolatának kérdése is. Többek között az is, hogy mikor egyezik meg egymással egy determinisztikus és egy nem determinisztikus szemantika. Ez a kérdés azért is izgalmas, mert egy program hatékonyabban értékelhető ki determinisztikus módon, hiszen a szabályok párhuzamosan alkalmazhatóak, így a szemantikák egyezőségét ki tudjuk használni az optimalizálásban. Az egyezőséget [ASV] alapján precízebben megfogalmazva:

2.2. Definíció: Legyen P program, q output predikátum. A P determinisztikus és nem determinisztikus szemantikája megegyezik q -ra vonatkozóan, ha a P minden inputja esetén a determinisztikus szemantikával kapott output megegyezik az összes nem determinisztikus úton kapott értékkel.

Előfordulhat, hogy egy programot nem determinisztikusan kiértékelve az összes kiértékeléssel ugyanazt az eredményt kapjuk, ez azonban eltér a determinisztikus szemantikával kapottól. Ha olyan programokat tekintünk, amelyekben nincs megengedve a negáció, akkor a két szemantika megegyezik. Sajnos azonban a többi nyelvcsalád esetén általában eldönthetetlen kérdés az, hogy egy tetszőleges program esetén megegyezik-e a két szemantika vagy sem. [ASV]

Az adatbázisnyelvként használt logikai programnyelvek közül legismertebb a Datalog, illetve ennek negációval való kibővítése. Ezek a DL különböző megszorításai. Mivel az irodalomban nagyon sok szó esik a Datalog-ról, – az alapvető ismereteket részletesen tárgyalja pl. [CGT], [U] – ezért most csak nagyvonalakban ismertetem.

Datalog

A Datalog Horn-klózek, azaz

$$A \leftarrow B_1, \dots, B_n$$

alakú szabályok halmaza, ahol A, B_i ($i = 1, \dots, n$) pozitív (nem negált) literálok. ([U] megenged úgynevezett beépített predikátumokat is $<, >, =, \neq$).

Természetesen a Datalog programoknál is megköveteljük a biztonságosságot.

Azok a predikátumok, melyekhez tartozó relációkat az adatbázisban tároljuk, az EDB predikátumok, a megfelelő relációk az EDB relációk. Ezeket szokták tényeknek is nevezni. Azokat a predikátumokat (relációkat), melyeket logikai szabályok definiálnak, IDB predikátumoknak (relációknak) nevezzük.

Mint látható, egy Datalog program speciális logikai programnak tekinthető, szabályai azonban általában kevésbé összetettek, mint egy általános célú logikai programé, és az argumentumok között csak változók és konstansok szerepelnek, függvényszimbólumok azonban nem. Ullman kiterjeszti a Datalog fogalmát függvényeket is tartalmazó esetre [U], dolgozatomban azonban nem kívánok evvel a témakörrel foglalkozni.

A Datalog szabályok kiértékelésekor két utat követhetünk. Az egyik módszer szerint minden IDB predikátumhoz meghatározunk egy relációt az EDB predikátumokhoz tartozó relációk segítségével. Rekurzív programok esetén egy Datalog egyenletrendszerhez jutunk, melyet iteratív módon oldhatunk meg. Ezt a módszert preferálja [U]. [CGT] a másik utat részesíti előnyben, amely szerint egy rákövetkezési operátort definiálhatunk, s ez alapján számolhatjuk ki az új tényeket. Mint látjuk, mindkét megoldási algoritmus egy fixpontkeresés, és a legkisebb fixpontot eredményezi. Ez a fixpont egyúttal a Datalog program egyértelmű minimális modellje. Datalog programok esetén a determinisztikus, illetve nem determinisztikus szemantika megegyezik.

Nem ilyen egyszerű a helyzet, ha negációt is megengedünk, hiszen a negatív információk kezelése elég nehéz feladat. A DL és megszorításainak egy része engedi a negáció alkalmazását. Ezeket a megszorításokat az irodalomban sok helyen a Datalog bővítéseként kezelik, ezért a Datalog-hoz kapcsolódó jelöléseket használnak. Mi is ezt a jelölést alkalmazzuk, s a lehetőségek közül csak Datalog^- kerül terítékre.

Datalog⁻

A „negatív” adatok tömege jóval nagyobb, mint a „pozitívaké”, ezért csak a pozitívakat tároljuk, s a negatívakra következtetünk. Többféle következtetési mód is ismert, de mindegyik kapcsolódik valamilyen módon a zárt-világ feltételezéshez (CWA). A CWA lényege, hogy ha egy tény logikailag nem

következik egy klózalmazból, akkor arra következtetünk, hogy a tény negáltja igaz, azaz ha C egy klózalmaz és F alap-literál, akkor

$$C \models^{CWA} F \Leftrightarrow F \text{ pozitív és } C \models F, \text{ vagy } F \text{ negatív és } C \not\models \neg F$$

Ha a CWA elvet a közönséges Datalog-ra alkalmazzuk, akkor is tudunk negatív tényekre következtetni, de ezekből a negatív tényekből már nem határozhatunk meg újabbakat. Hogy tovább tudjunk belőlük következtetni, meg kell engednünk negatív literálok használatát a szabálytörzsben. Így jutunk a Datalog⁻ nyelvhez.

A negatív literálok alkalmazása bizonytalan szabályokhoz vezethet, mint ahogy a $\text{nőtlen}(x) \leftarrow \text{férfi}(x), \neg \text{házas}(x,y)$ példa is mutatja. Ezért a biztonságosság érdekében újabb megszorítást kell tennünk: az összes olyan változónak, amely negatív literálban szerepel, elő kell fordulnia pozitív literálban is. A negáció miatt előfordulhat az is, hogy nem kapunk legkisebb fixpontot, hanem több minimálisat. (Ez hasonlóan több minimális modellt is jelent.) Kérdés, melyiket válasszuk közülük, azaz milyen szemantikát rendeljünk a Datalog⁻-hoz. Kétféle szemantikát szoktak értelmezni, az egyik a rétegzés, a másik az inflációs szemantika.

Rétegzéskor egyfajta rendezést vezetünk be a predikátumok között, s a programokat ebben a sorrendben értékeljük ki, vagyis a deklaratív szabályokhoz egy kis procedurális vonás is társul. Ha ebben a sorrendben végezzük el a kiértékelést, akkor egy negált predikátumra már csak akkor kerül sor, ha előbb minden pozitív előfordulási helyén kiértékeljük. [U]

Az inflációs szemantika megegyezik a DL nyelv szemantikájával. Vagyis – a precíz részletek mellőzésével – addig alkalmazzuk a program szabályait, ameddig újabb adatokat kapunk. Ez a szemantika „hatásosabb”, mint a rétegzés. Ez azt jelenti, hogy található olyan lekérdezés, amely kifejezhető az inflációs Datalog⁻-ban, de nem fejezhető ki a rétegzettben. [CGT]

2.1. Példa: Tekintsük a következő programot:

$$\begin{array}{ll} A \leftarrow \neg A, \neg B & B \leftarrow \neg A, \neg B \\ C \leftarrow A, \neg B & C \leftarrow \neg A, B \\ A \leftarrow C & B \leftarrow C \end{array}$$

Ez egy nem rétegezhető Datalog[¬] program, ezért csak az inflációs szemantika alkalmazható. Nem determinisztikus szemantikával bármely sorrendben is értékeliük ki a szabályokat, mindig {A,B,C}-t kapunk eredményül, míg determinisztikus szemantikával {A,B}-t. Vagyis ebben az esetben a determinisztikus és a nem determinisztikus szemantika eltér egymástól.

Bizonytalanságkezelési alapok – a háromértékű logika alkalmazása

A klasszikus logikával leírható összefüggések, adatmodellek alkalmazhatóságát – és így a Datalog, Datalog[¬] alkalmazhatóságát is – erősen megszorítja az a tény, hogy minden állításban, információban bizonyosnak kell lennünk, azaz egyértelműen el kell tudnunk dönteni azt, hogy ezek igazak-e vagy hamisak. A gyakorlatban azonban sajnos nem ennyire egyszerű a helyzet. Sokszor kerülünk olyan szituációba, sokszor kell olyan információt kezelnünk, amelyről nincs biztos ismeretünk. Időnként maguk a kérdések is rosszul definiáltak. A bizonytalan információk kezelésének egyik lehetséges megközelítési módja, hogy bizonyos tényeket igaznak tekintünk, bizonyosokról biztosan tudjuk, hogy hamisak, és vannak olyanok, amelyek igaz vagy hamis voltáról nem tudunk semmit. Az ilyen típusú adatbázis kezeléséhez a háromértékű logikát hívhatjuk segítségül.

A háromértékű logikában – mint neve is mutatja – három igazságértéket különböztetünk meg: igaz(1), hamis(0) és definiálatlan(0.5). A háromértékű interpretáció a Herbrand bázis atomjaihoz ezen három érték valamelyikét rendeli. Ez az interpretáció formulákra is kiterjeszhető a következő módon:

$$\begin{aligned} I(\neg A) &= 1 - I(A) \\ I(A \wedge B) &= \min(I(A), I(B)) \\ I(A \vee B) &= \max(I(A), I(B)) \\ I(A \rightarrow B) &= 1, \text{ ha } I(B) \geq I(A) \\ &0 \text{ egyébként.} \end{aligned}$$

Egy I interpretáció modellje egy Φ formulahalmaznak, ha $I(\phi) = 1$ minden $\phi \in \Phi$ esetén.

Az igazságértékek alapján értelmezhető az igazságértékek közötti rendezés, így ebben az esetben is beszélhetünk minimális modellről. A minimális modellben minimális az igaz atomok és maximális a hamis atomok halmaza. Ez a kétértékű logikával tárgyalt adatbázisok minimális modell fogalmának általánosítása, ahol a program modelljét képező igaz atomok minimális halmazát határozzuk

meg. Ezt a minimális háromértékű modellt adja meg az adatbázishoz a Van Gelder, Schlipf és Ross által bevezetett, úgynevezett megalapozott (well-founded) szemantika. [GRS] Erről a szemantikáról sok egyéb helyen is olvashatunk, többek között ezekben a művekben: [AHV], [GM], [LLS], [Pr1], [Pr2], [Pr3].

Ez a szemantika is fixpont-keresésen alapszik, csak most „két oldalról” közelítjük meg a fixpontot. Ez nagyvonalakban a következőt jelenti:

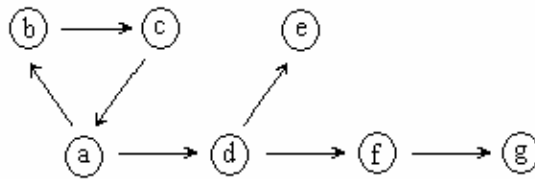
Jelentse T az igaz, F a hamis állítások halmazát. Ekkor egy interpretáció az $I = \langle T, F \rangle$ párral adható meg. Ez az értelmezés a korábbiakban tárgyalt általánosítása, hiszen kétértékű esetben $T \cup F = B_P$, vagyis az igaz és hamis atomok halmaza együtt a program teljes Herbrand bázisát alkotja, míg háromértékű esetben annak csak egy részhalmazát.

Definiálunk egy T_I és egy F_I operátort. $T_I(S)$ az S -ből közvetlenül származtatható igaz tényeket, $F_I(S)$ pedig a zárt világ feltételezés alapján származtatható hamisakat tartalmazza. Mindkét operátor monoton, így mindkettőnek van fixpontja. A P programhoz rendelt $T_P: I \rightarrow I \cup \langle T_I, F_I \rangle$ operátor határértékét (legkisebb fixpontját) tekintjük a P megalapozott modelljének.

Abiteboul, Hull és Vianu egy, az előzőnél hatékonyabb algoritmust definiál a megalapozott szemantika meghatározására. Ráadásul azt is bebizonyítják, hogy minden Datalog^- programhoz létezik megalapozott szemantika [AHV]. Ez az algoritmus szintén fixpont-keresés, egy alternáló $\{I_i\}_{i \geq 0}$ sorozat határértékeként adódik az eredmény. Szintén csak nagyvonalakban, ez a következőt jelenti: Kiindulunk a programhoz kapcsolható összes lehetséges hamis atom halmazából (I_0), ez nyilván felső becslést jelent az eredmény hamis tényeire vonatkozóan. I_1 az I_0 -ból következtethető atomok halmaza. A benne lévő igaz atomok halmaza az eredmény igaz tényeire vonatkozóan felső becslés lesz, míg a hamis atomok halmaza alulról becsli az eredmény hamis tényeinek halmazát. I_2 az I_1 -ből következtethető atomokat tartalmazza, amely az eredmény igaz tényeire vonatkozóan lesz alsó becslés, a hamisakra vonatkozóan pedig felső. Az eljárást folytatva belátható, hogy a páros indexű elemek határértéke, I^* , az eredmény-halmaz igaz tényeit tartalmazza, a páratlan indexűek fixpontja, I_* pedig az eredményhalmaz hamis tényeit. I^* igaz, és I_* hamis tényeinek uniója adja a program megalapozott szemantikáját.

Illusztrációként [AHV] alapján tekintsük a következő példát:

2.2. Példa: A példa egy kétszemélyes játék gráfját szemlélteti. A gráfról leolvasható lehetséges lépéseket a mozgat (m) bináris reláció írja le. Egy játékos veszít, ha olyan állapotba kerül, ahonnan nem tud továbblépni. Kérdés, melyek azok az állapotok, amelyből nyerni lehet. Erre vonatkozóan egy szabályt fogalmazunk meg, amelyben a nyerő állapotot az egy változós „ n ” reláció jelöli. A játék gráfja:



Innen a mozgatási relációk halmaza:

$$\{m(a,b), m(b,c), m(c,a), m(a,d), m(d,e), m(d,f), m(f,g)\},$$

a nyerési lehetőséget megfogalmazó, nem rétegezhető szabály pedig:

$$n(x) \leftarrow m(x,y), \neg n(y).$$

Intuitíve azonnal látszik, hogy azok az állások, amelyekből végállapotba tudunk lépni, biztosan nyerők, hiszen az ellenfél onnan nem tud továbbmenni, a végállapotok pedig biztosan veszítők, hiszen onnan mi nem tudunk továbblépni. A többi állapotról azonban semmi biztosat nem mondhatunk. Vagyis a háromértékű logika nyelvén fogalmazva:

$$\begin{aligned} \text{igaz :} & \quad n(d), n(f) \\ \text{hamis:} & \quad n(e), n(g) \\ \text{ismeretlen:} & \quad n(a), n(b), n(c). \end{aligned}$$

A fent említett fixpont-számítással is ugyanezt az eredményt kapjuk:

Induljunk ki abból a feltételezésből, hogy az m -re vonatkozó összes atom hamis (I_0). Ugyanakkor minden további I_i , $i \geq 1$ esetén az összes ilyen atom igaz, és beletartozik I_i -be, ezért a rövidebb írásmód kedvéért I_i -nek csak az n predikátumra vonatkozó elemeit soroljuk fel.

$$\begin{aligned} I_1 &= \{n(a), n(b), n(c), n(d), \neg n(e), n(f), \neg n(g)\} \\ I_2 &= \{\neg n(a), \neg n(b), \neg n(c), n(d), \neg n(e), n(f), \neg n(g)\} \\ I_3 &= I_1 \\ I_4 &= I_2 \end{aligned}$$

Innen a megalapozott szemantika n-re vonatkozó része:

$$I = \{n(d), \neg n(e), n(f), \neg n(g)\}.$$

A háromértékű logika három csoportba sorolja az adatokat, információkat: a biztosan igaz, a biztosan hamis és a definiálatlan információk csoportjába. Sokszor előfordulhat azonban, hogy ez utóbbi adatokról is tudunk valamit, csak tudásunk bizonytalan. Ezekhez az állításokhoz „bizonytalansági mértéket” rendelhetünk. Megadhatjuk azt is, hogy egy tény mennyire igaz, de azt is, hogy egy következtetési szabály mennyire helytálló. Ezeket az értékeket a gyakorlatban szakértők határozhatják meg tapasztalataik alapján, vagyis a szakértők véleményét, tudását tükrözik.

Mint maga a vizsgált téma, a bizonytalanságot kezelő módszerek sem kristályosodtak ki, így – mivoltukból fakadóan teljesen természetes módon – sokfajta tárgyalásmód található az irodalomban. Vannak, akik bizonytalansági mérték-ként nem is egy, hanem két értéket javasolnak – egy szükségességi és egy lehetőségi mértéket. (pl. [B], [DP1], [DLP], stb.). Ezen értékek között változhat az információ bizonytalansági szintje.

Mások, mint pl. [AK1], [K1], [LL], stb. az információkhoz csak egy [0,1] közötti bizonytalansági értéket rendelnek. Mások megint más bizonytalanság-kezelő módszert javasolnak – pl. halmaz típusú értékek alkalmazását, mint pl. [BPS], [SMF], mi azonban a továbbiakban megmaradunk az előbbi elképzelés mellett, vagyis annál, hogy az információkat egy [0,1] közötti bizonytalansági szinttel látjuk el, és bár az ilyen módon megadott bizonytalanságok kezelésére is több fajta módszer létezik, mi a fuzzy logikát hívjuk segítségül.

Összegezve az eddig elmondottakat: a Codd féle relációs adatbázis fogalmának három lehetséges irányú bővítéséről beszéltünk: a rekurzió kezelésére szolgáló dedukcióról, a negáció fogalmának további tágításáról, illetve megemlítettük a bizonytalanság kezelésére alkalmas harmadik irányt, a fuzzy logika irányába való terjeszkedés lehetőségét. A relációs adatbázis ilyen irányú kiterjesztései ortogonálisnak tekinthetők abban az értelemben, hogy egymástól függetlenül alkalmazható bármelyik irányú bővítés. A bővíthetőséget szépen illusztrálja a mellékelt, úgynevezett Wagner-féle tudáskocka [SS1], [SS2]:

A tudáskocka csúcsai a következőt jelentik:

rDB: relációs adatbázis.

dDB: deduktív adatbázis: a relációs adatbázis szabályokkal és következtetési mechanizmussal kibővíve (pl.: Prolog, Datalog).

rFB: relációs tényhalmaz (factbase): egy pozitív és egy negatív adatokat tartalmazó adatbázis.

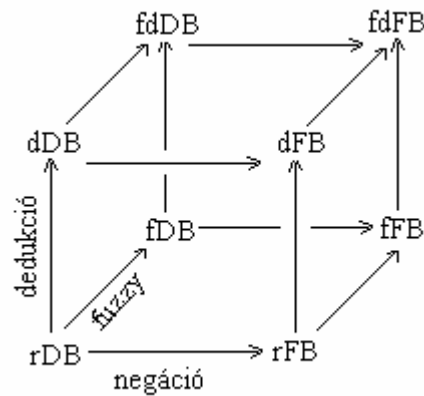
fDB: fuzzy adatbázis: olyan relációs adatbázis, ahol minden adathoz hozzárendelünk egy $[0,1]$ közötti igazságértéket.

dFB: deduktív tényhalmaz (factbase): a relációs tényhalmaz szabályokkal és következtetési mechanizmussal kibővíve.

fFB: fuzzy tényhalmaz: olyan tényhalmaz, ahol mind a pozitív mind a negatív halmaz elemeihez hozzá van rendelve egy $[0,1]$ közötti igazságérték.

fdDB: fuzzy deduktív adatbázis: fuzzy adatbázis szabályokkal és következtetési mechanizmussal kibővíve. Wagner feltételezi, hogy az alkalmazott szabályok nem fuzzy szabályok.

fdFB: fuzzy deduktív tényhalmaz: fuzzy tényhalmaz szabályokkal és következtetési mechanizmussal kibővíve.



Dolgozatomban a kocka bal felső hátsó csúcsának, vagyis fdDB-nek a „környékével” kívánok foglalkozni, de Wagnertől eltérően a szabályokat is bizonytalansági szintekkel látom el.

3. A Datalog fuzzy kiterjesztése

A továbbiakban a Datalog egy lehetséges fuzzyfikálásáról lesz szó. A közönséges Datalog szabályokat kiegészítjük egy bizonytalansági szintet jelző értékkel és egy implikációs operátorral, amely segítségével következtetni tudunk a szabályfej bizonytalansági fokára, ily módon az atomokhoz is bizonytalansági szintet rendelünk. Így azt határozzuk meg, hogy az illető szabály vagy predikátum legalább mekkora szinten teljesül, vagyis gyakorlatilag egy levágást hajtunk végre. Az így kapott programnyelvet fDATALOG-nak nevezzük.

3.1. A fuzzy Datalog szintaktikája

3.1. Definíció: fDATALOG szabály egy $(r; \beta; I)$ hármas, ahol r egy

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

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

Az fDATALOG szabály biztonságos, ha

- a fejből 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.

Egy P program *Herbrand univerzumán* (H_P) általában a P -ben előforduló konstansokból és függvényszimbólumokból képzett összes lehetséges alapatom halmazát értjük, mivel azonban dolgozatomban nem foglalkozom a függvényszimbólumokat is megengedő programokkal, ezért esetünkben egy program Herbrand univerzuma a program konstansainak halmaza. A P *Herbrand bázisa* (B_P) a P -ben előforduló predikátumszimbólumokból képzett összes lehetséges olyan alapatom halmaza, melynek argumentumai H_P elemei. Egy $(r; \beta; I)$ szabály P -beli *alapelőfordulása* egy olyan szabály, amelyet úgy kapunk r -ből, hogy az összes r -beli X változót $\Phi(X)$ -szel helyettesítjük, ahol Φ az r -ben

előforduló változók H_P -be való leképezése. Az $(r; \beta; I)$ szabály összes alapelőfordulásának halmazát $(ground(r); \beta; I)$ -vel jelöljük. A P program alapelőfordulása:

$$ground(P) = \cup_{(r; \beta; I) \in P} (ground(r); \beta; I)$$

3.2. Definíció: Egy P program interpretációja B_P egy fuzzy részhalmaza:

$$N_P \in F(B_P),$$

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

A konjunkció és negáció szintjét a szokásos módon értelmezzük, azaz tetszőleges A_1, \dots, A_n alapatomok esetén :

$$\alpha_{A_1 \wedge \dots \wedge A_n} \stackrel{\text{def}}{=} \min(\alpha_{A_1}, \dots, \alpha_{A_n})$$

$$\alpha_{\neg A} \stackrel{\text{def}}{=} 1 - \alpha_A$$

3.3. Definíció: Egy interpretáció a P program modellje, ha minden

$$(A \leftarrow A_1, \dots, A_n ; \beta; I) \in ground(P)$$

esetén

$$I(\alpha_{A_1 \wedge \dots \wedge A_n}, \alpha_A) \geq \beta.$$

Az M legkisebb modell, ha bármely N modell esetén $M \leq N$, vagyis M az összes modell metszete. M minimális modell, ha nincs olyan N modell, hogy $N \leq M$ teljesüljön, vagyis nem lehet úgy csökkenteni M alaptermjeinek igazságértékét, hogy továbbra is modellt kapjunk.

Az egyszerűség kedvéért $\alpha_{A_1 \wedge \dots \wedge A_n}$ -t $\alpha_{\text{törzs}}$ -szel, és α_A -t α_{fej} -jel jelöljük.

Megjegyzés: Az fDATALOG a közönséges Datalog általánosításának tekinthető, hiszen ha minden szabályhoz az I_7 -tel jelölt (M.1. táblázat), ú.n. Gaines-Rescher implikációs operátort és $\beta = 1$ bizonytalansági szintet rendelünk, akkor Datalog szabályok halmazát kapjuk.

A matematikai logikában az igazság fogalmát két oldalról közelítjük meg, a modellelmélet és a bizonyításelmélet felől. Modellelméleti szempontból egy logikai formula akkor és csak akkor igaz tautologikusan, ha az összes lehetséges interpretációban igaz, vagyis ha a formula minden interpretációja modell.

A bizonyításelméleti megközelítés szerint a formula akkor és csakis akkor tautológia, ha valamely axiómarendszerből adott szabályok szerint levezethető. A matematikai logika egyik fő kérdése a két megközelítés ekvivalenciájának bizonyítása.

Ugyanez a kérdés merül föl a logikai adatbázisok körében is. A következő fejezetben megadjuk a fDATALOG egy bizonyításelméleti megközelítését, és megvizsgáljuk, hogy ez a megközelítés ekvivalens-e a modelleméleti megközelítéssel. Ez azt jelenti, hogy megadunk egy levezetési eljárást, melynek fixpontjáról belátjuk, hogy a vizsgált program modellje, vagyis hogy a két irányú megközelítés ekvivalens. Ezek után definiálhatjuk a fDATALOG program szemantikáját.

3.2. A fuzzy Datalog szemantikája

Két rákövetkezési transzformációt értelmezünk, egy determinisztikus és egy nondeterminisztikus transzformációt. Ennek megfelelően kétféle szemantikát rendelhetünk a fDATALOG programokhoz. Ezek a szemantikák pozitív (negáció mentes) programok esetén megegyeznek, de negációt tartalmazó programoknál különböző eredményt adhatnak.

3.4. Definíció: 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

$$(A \leftarrow A_1, \dots, A_n; \beta; I) \in \text{ground}(P), (|A_i|, \alpha_{A_i}) \in X, 1 \leq i \leq n;$$

$$\alpha_A = \max(0, \min \{ \gamma \mid I(\alpha_{\text{törzs}}, \gamma) \geq \beta \}).$$

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

Megjegyzés: Vegyük észre, hogy az NT_P transzformáció esetén az X halmaz csupán egy legfőbb egy elemű halmazzal bővül, míg a DT_P transzformáció során ez a bővülés sok elemet tartalmazhat.

A vizsgált implikációs operátorok körét le kell szűkítenünk, ugyanis nem minden I esetén létezik a második változó szerinti pszeudo-inverz, azaz nem határozható meg tetszőleges $\alpha_{\text{törzs}}, \beta$ - hoz a kívánt α_A érték.

Értelmezzük az úgynevezett bizonytalansági-szint függvényt:

3.5. Definíció: Az

$$f(I, \alpha, \beta) = \min(\{\gamma \mid I(\alpha, \gamma) \geq \beta\})$$

függvényt bizonytalansági-szint függvénynek nevezzük.

A mellékletben tárgyalt (M.1. tábla) implikációs operátorokra vonatkozóan egyszerű számolással ellenőrizhető a következő állítás:

3.1. Állítás: Bármely $I \in \{I_1, I_2, I_3, I_4, I_5, I_7\}$ esetén tetszőleges $\alpha_{\text{törzs}}, \beta$ értékhez létezik az $f(I_i, \alpha_{\text{törzs}}, \beta)$ ($i = 1, 2, 3, 4, 5, 7$) érték.

A szóban forgó implikációs operátorokhoz tartozó bizonytalansági szint függvények a következő módon határozhatók meg:

$$f(I_1, \alpha, \beta) = \min(\alpha, \beta)$$

$$f(I_2, \alpha, \beta) = \max(0, \alpha + \beta - 1)$$

$$f(I_3, \alpha, \beta) = \alpha \cdot \beta$$

$$f(I_4, \alpha, \beta) = \begin{cases} 0 & \alpha + \beta \leq 1 \\ \beta & \alpha + \beta \geq 1 \end{cases}$$

$$f(I_5, \alpha, \beta) = \max(0, 1 + (\beta - 1)/\alpha), \alpha \neq 0$$

$$f(I_7, \alpha, \beta) = \alpha$$

$I = I_6$ esetén nem értelmezett minden α, β esetén az $f(I_6, \alpha, \beta)$ függvény. Ennek belátásához elég példaként az $\alpha_{\text{törzs}} = 0.6, \beta = 0.7$ értékeket tekinteni, ekkor ugyanis tetszőleges γ esetén $0.4 \leq I_6(\alpha_{\text{törzs}}, \gamma) \leq 0.6$, vagyis nincs olyan γ érték, melyre $I(\alpha_{\text{törzs}}, \gamma) \geq \beta$ teljesülne.

A továbbiakban feltételezzük az I implikációs operátor pszeudo-invertálhatóságát, vagyis azt, hogy tetszőleges $(\alpha_{\text{törzs}}, \beta)$ -hoz meghatározható a kívánt $\alpha_A = f(I, \alpha_{\text{törzs}}, \beta)$ érték.

A tényhalmazból kiindulva egy tetszőleges transzformáció egymás utáni alkalmazásával (vagyis a transzformáció hatványainak meghatározásával) értelmezünk egy halmazsorozatot:

Tetszőleges $T: F(B_P) \rightarrow F(B_P)$ transzformáció esetén legyen

$$T_0 = \{\cup\{(A, \alpha_A)\} \mid (A \leftarrow; \beta; I) \in \text{ground}(P), \alpha_A = \min\{\gamma \mid I(1, \gamma) \geq \beta\}\} \cup \{(A, 0) \mid \exists (B \leftarrow \dots \neg A \dots; \beta; I) \in \text{ground}(P)\},$$

és legyen

$$T_1 = T(T_0)$$

...

$$T_n = T(T_{n-1})$$

...

$$T_\omega = \sup\{T_i \mid i < \omega\}; \text{ ahol } \omega \text{ az első végtelen rendtípus.}$$

A kiindulásul használt T_0 halmazban a ténypredikátumok egy-egy bizonytalansági szinttel együtt szerepelnek. Ezeket az α_A értékeket a tények bizonytalansági mértékéből számoljuk ki. Célszerű lenne olyan implikációs operátorokat használni, amelyek esetén az $(A \leftarrow; \beta; I)$ tényekre vonatkozó bizonytalansági szint és a predikátumhoz kiszámolt bizonytalansági szint megegyezik, vagyis $\beta = \alpha_A$ teljesül. Ennek feltétele, hogy az implikációs operátor kielégítse az $I(1, y) = y$ összefüggést. Egyszerű számolással belátható, hogy az I_k , $k = 1, \dots, 6$ operátorok eleget tesznek ennek a feltételnek, az I_7 azonban nem. Ez utóbbi esetben a kiszámított bizonytalansági szint értéke β -tól függetlenül $\alpha_A = 1$. A továbbiakban – ahol ez nem okoz félreértést – az egyszerűség kedvéért $(A; \alpha_A)$ -val jelöljük a tényeket.

3.2. Állítás: A DT_P és az NT_P transzformációnak van fixpontja, vagyis létezik olyan $X \in F(B_P)$ és $Y \in F(B_P)$, melyre $DT_P(X) = X$, és $NT_P(Y) = Y$.

Ha P pozitív, vagyis negáció-mentes, akkor a két fixpont megegyezik, és ez a közös X érték a legkisebb fixpont. (Azaz bármely $Z = T(Z)$ esetén $X \leq Z$.)

Bizonyítás:

Többek között [CGT], [GS] bizonyítja, hogy egy L teljes hálón értelmezett inflációs T transzformációnak létezik fixpontja. (T inflációs, ha $X \leq T(X)$ minden $X \in L$ esetén.) Mivel DT_P és NT_P inflációs transzformációk, és $\overline{F}(B_P)$ teljes háló, ezért mindkettőnek létezik inflációs fixpontja.

Ha P pozitív, akkor $DT_P = NT_P$. Ez a transzformáció monoton, ezért az M.3. tételben említett, a monoton transzformációkra vonatkozó fixponttétel szerint igaz az állítás. \square

Ezeket a fixpontokat $\text{lfp}(DT_P)$ -vel és $\text{lfp}(NT_P)$ -vel jelöljük. Belátjuk, hogy az így kapott fixpontok a P modelljei, így segítségükkel értelmezni tudjuk egy program jelentését.

3.1. Tétel: $\text{lfp}(DT_P)$ és $\text{lfp}(NT_P)$ a P modelljei.

Bizonyítás:

$T = DT_P$ és $T = NT_P$ esetén $\text{ground}(P)$ -ben a következő alakú szabályok vannak:

a/ $(A \leftarrow; \beta; I)$,

b/ $(A \leftarrow A_1, \dots, A_n; \beta; I); (A, \alpha_A) \in \text{lfp}(T)$ és $(|A_i|, \alpha_{A_i}) \in \text{lfp}(T), 1 \leq i \leq n$,

c/ $(A \leftarrow A_1, \dots, A_n; \beta; I); \exists i : (|A_i|, \alpha_{A_i}) \notin \text{lfp}(T)$.

Az a, b esetben α_A konstrukciója miatt $I(\alpha_{\text{törzs}}, \alpha_A) \geq \beta$ teljesül, c esetén T_0 konstrukciójából adódóan A_i nem negatív, így $\alpha_{A_i} = 0$, ezért $\alpha_{\text{törzs}} = 0$, amiből $I(\alpha_{\text{törzs}}, \alpha_A) = 1 \geq \beta$ következik. Vagyis $\text{lfp}(T)$ modell. \square

Ezek után definiálhatjuk az fDATALOG programok szemantikáját.

3.6. Definíció: $\text{lfp}(DT_P)$ az fDATALOG P program determinisztikus-, $\text{lfp}(NT_P)$ a nemdeterminisztikus szemantikája.

3.3. Állítás: Negációmentes fDATALOG esetén a két szemantika megegyezik.

Bizonyítás:

Mivel ebben az esetben a két transzformáció megegyezik, ezért a két szemantika is. \square

3.4. Állítás: Negációmentes fDATALOG esetén $\text{lfp}(DT_P) (= \text{lfp}(NT_P))$ legkisebb modell.

Bizonyítás:

Közönséges pozitív Datalog program esetén a legkisebb fixpont egyúttal legkisebb modell is ([CGT], [U]). Fuzzy Datalog esetén a rákövetkezési transzformációk definíciója alapján egy szabályfej kiszámított bizonytalansági szintje a legkisebb olyan érték, amely még teljesíti a szabályra vonatkozó modell-kritériumot. Probléma csak abból adódhatna, ha a fixpontszámítás során egy későbbi lépésben alacsonyabb fokszámot rendelhetnénk egy szabályfejhez, mint egy korábbi lépésben, de a transzformációk definíciója alapján a magasabb értéket kellene elfogadnunk. Ilyen eset azonban csak negációt tartalmazó programok körében fordulhat elő. Így az állítás igaz. \square

Belátjuk, hogy pozitív programok és bizonyos tulajdonsággal rendelkező implikációs operátorok esetén létezik algoritmus a legkisebb modell meghatározására, azaz a fixpontszámítási eljárás véges sok lépésben befejeződik. Arra is mutatunk példát, hogy van olyan implikációs operátor, amely alkalmazásakor előfordulhat, hogy a program nem terminál véges lépésszámon belül.

3.5. Állítás: Tegyük fel, hogy az I implikációs operátorhoz tartozó $f(I, \alpha, \beta)$ bizonytalansági-szint függvényre teljesül az $f(I, \alpha, \beta) \leq \alpha, \forall \alpha \in [0,1]$ feltétel. Ekkor minden negációmentes fDATALOG P programhoz létezik olyan n természetes szám, melyre $T = DT_P$ vagy $T = NT_P$ esetén $T_n = T_{n+1} = \dots = \text{lfp}(T)$.

Bizonyítás:

Mivel P véges, ezért a kapott fixpontban csak véges sok alapatom szerepelhet. A bizonytalansági-szint értékek csak rekurzív predikátum esetén növekedhetnek végtelen sok lépésben az eljárás során. Ha a rekurzív predikátum a rá vonatkozó rekurzív szabály törzsében minimális bizonytalansági-szint értékkel rendelkezik, akkor a bizonytalansági-szint függvény előírt tulajdonsága miatt nem kaphatunk nagyobb bizonytalansági értéket a következő lépésben, ha viszont nem ő a minimális bizonytalanságú, akkor a törzs bizonytalansági szintjének kiszámítási módja miatt eleve nem juthatunk a korábinál nagyobb bizonytalansági értékhez. \square

Gyors számolással ellenőrizhető a következő állítás:

3.6. Állítás: Az I_1, I_2, I_3, I_7 operátorok kielégítik a 3.5. állításban megfogalmazott feltételt.

3.7. Állítás: A fixpontszámító algoritmus az I_4 operátor alkalmazásakor is véges sok lépésben terminál.

Bizonyítás:

Bár az operátor nem tesz eleget a 3.5. állítás feltételének, de mivel $f(I_4, \alpha, \beta)$ csak 0 és β értéket vehet fel, ezért a fokszámok ebben az esetben sem növekedhetnek végtelenségig. \square

3.8. Állítás: Az I_5 operátor alkalmazása esetén megadható olyan szabály, hogy a program csak végtelen sok lépésben terminál.

Bizonyítás:

Az $f(I_5, \alpha, \beta) = \max(0, 1 + (\beta - 1)/\alpha)$ függvény $\alpha + \beta \geq 1$ esetén α -ban szigorúan monoton növekvő, és $\beta \geq 0.75$ esetén van olyan α , melyre $1 \geq f(I_5, \alpha, \beta) \geq \alpha$. Ilyen szintekkel rendelkező rekurzív szabály esetén előfordulhat, hogy a szabály újbóli alkalmazásakor az előzőnél nagyobb, de 1-nél kisebb szintértékkal kerül be a megoldáshalmazba a fejpredikátumhoz tartozó alapatom, így a fixpont csak végtelen sok lépéssel érhető el. \square

3.1. Példa: A

$$(p(a); 0.6).$$

$$p(a) \leftarrow p(a); 0.8; I_5.$$

program nem terminál véges sok lépésben. A $p(a)$ alapatomhoz tartozó bizonytalansági szint értékére egy szigorúan monoton növekvő sorozatot kapunk.

Láttuk, hogy pozitív programok esetén a determinisztikus és nondeterminisztikus szemantika azonos eredményre, a program legkisebb modelljéhez vezet. A következő két példa mutatja, hogy negációt tartalmazó programok esetén ez nincs így.

3.2. Példa: Tekintsük az egyetlen szabályból álló programot:

$$p(a) \leftarrow \neg q(b); 0.7; I_1.$$

Ennek a programnak nincs legkisebb modellje, hanem csak két minimális: $M_1 = \{(p(a); 0.7)\}$, illetve $M_2 = \{(q(b); 1)\}$. Az előzőekben definiált fixpontszámítási eljárás mindkét transzformáció esetén M_1 -hez vezet.

3.3. Példa:

1. $(r(a); 0.8)$.
2. $p(x) \leftarrow r(x), \neg q(x); 0.6; I_1$.
3. $q(x) \leftarrow r(x); 0.5; I_1$.
4. $p(x) \leftarrow q(x); 0.8; I_1$.

Ekkor

$$\text{lfp}(DT_P) = \{(r(a),0.8); (p(a),0.6); (q(a),0.5)\}$$

Nem determinisztikus kiértékeléskor különböző megoldásokat kaphatunk, hiszen az függ a kiértékelendő szabályok sorrendjétől. Ha a kiértékelendő szabályok sorrendje 1.,2.,3.,4., akkor

$$\text{lfp}(NT_P) = \{(r(a),0.8); (p(a),0.6); (q(a),0.5)\},$$

míg 1.,3.,2.,4. sorrendben

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

A 3.2. példából látható, hogy negációt tartalmazó esetben több minimális modell is lehet, a 3.3.-ból pedig az, hogy negációt tartalmazó esetben a determinisztikus transzformáció legkisebb fixpontja nem mindig azonos a kívánt minimális modellel, tehát nem tekinthető helyes szemantikának. Nem determinisztikus esetben azonban bizonyos feltételek mellett biztosítható a minimalitás. Ez a feltétel a rétegzés. A rétegzés meghatároz egy kiértékelési sorrendet, amely szerint a negatív literálokat értékeljük ki először, s az ily módon kapott fixpont egyúttal minimális modell.

A rétegzéshez előbb a függőségi gráf fogalmát kell megadnunk. Ez egy olyan irányított gráf, melynek csúcsai a P program predikátumai. Egy p predikátumból akkor vezet él egy q predikátumba, ha P-nek van olyan szabálya, melynek törzsében p vagy $\neg p$ szerepel, és fej-predikátuma q.

Egy program rekurzív, ha függőségi gráfja egy vagy több kört tartalmaz. Az összes olyan predikátum, amely rajta van egy vagy több körön, rekurzív predikátum. A körmentes függőségi gráffal rendelkező programok rekurziómentesek.

Egy program rétegzett, ha az összes olyan esetben, amikor egy szabály fej-predikátuma q, és a törzs negált literálja $\neg p$, a függőségi gráfban nincs út q-ból p-be. Egy P program rétegzése a P predikátumszimbólumainak olyan P_1, \dots, P_n részhalmazokra való partíciója, melyre teljesülnek a következő feltételek:

a/ ha $q \in P_i, p \in P_j$ és nincs él p-ből q-ba, akkor $i \geq j$.

b/ ha $q \in P_i$, $p \in P_j$ és van olyan q fejpredikátumú szabály, melynek törzse tartalmazza $\neg p - t$, akkor $i > j$.

A P_1, \dots, P_n halmazokat rétegeknek nevezzük.

Egy P program többféle módon is rétegezhető. Egy rétegzés meghatároz egy kiértékelési sorrendet. Először azokat a szabályokat értékeljük ki, melyek fejpredikátumai P_1 -ben vannak, azután a P_2 -ben lévőket, és így tovább. Egy programot rétegezhetőnek nevezünk, ha megadható hozzá rétegzés. Nagyon egyszerű rétegzési algoritmust ad meg [CGT] és [U]. Ez alapján egy fDATALOG program is rétegezhető, hiszen az eljárás csak a programban szereplő predikátumokat csoportosítja.

Legyen P egy rétegzett fDATALOG program P_1, \dots, P_n rétegzéssel. Jelölje P_i^* a P_i réteghez tartozó összes P -beli szabály halmazát, azaz az összes olyan szabály halmazát, melynek fejpredikátuma P_i -ben van.

Legyen

$$L_1 = \text{lfp}(NT_{P_1^*}),$$

ahol a fixpontoszámolási eljárás kiindulópontja a korábban definiált T_0 .

Legyen

$$L_2 = \text{lfp}(NT_{P_2^*}),$$

ahol a számolás kiindulópontja az előbb kapott L_1 ,

...

$$L_n = \text{lfp}(NT_{P_n^*}),$$

ahol a kiindulópont L_{n-1} .

Más szóval, először kiszámítjuk a P első rétegéhez tartozó L_1 fixpontot, majd ennek meghatározása után léphetünk a következő rétegekre.

Megjegyzés: $\text{lfp}(NT_{P_i^*}) = \text{lfp}(DT_{P_i^*})$

Indukcióval belátjuk, hogy L_n a P minimális modellje. Ehhez szükségünk van a következő definícióra és lemmára.

3.7. Definíció: Egy fDATALOG P programot szemi-pozitívnak nevezünk, ha negált predikátumai kizárólag tény-predikátumok.

3.1. Lemma: Egy szemi-pozitív P programnak van minimális modellje: $L = \text{lfp}(\text{NT}_P) (= \text{lfp}(\text{DT}_P))$.

Bizonyítás:

Egy szemi-pozitív program nagyon hasonlóan viselkedik, mint egy pozitív. Ha ugyanis p egy negált predikátum P egy szabálytörzsében, akkor a p -re vonatkozó tényállítást, illetve a szabálytörzs megfelelő predikátumát is helyettesíthetjük a $q = \neg p$ -re vonatkozó állítással és a hozzá tartozó bizonytalansági szinttel. Ily módon eliminálhatjuk a programból a negációt. Az így kapott programnak létezik legkisebb fixpontja, ami egyúttal legkisebb modell is. \square

A lemma szerint $L_1 P_1^*$ legkisebb fixpontja. Általában: $L_{i-1} \cup P_i^*$ szemi-pozitív, hiszen P rétegzése miatt az i -edik réteg összes negatív literálja egy alacsonyabb szintű réteg predikátumához tartozik. Így $L_i P_i^*$ legkisebb fixpontja, amely az adott rétegzésre vonatkozóan minimális modell. Ily módon igaz a következő tétel:

3.2. Tétel: Ha P rétegzett fDATALOG program, akkor L_n a P minimális modellje.

Ez azt jelenti, hogy a rétegzés sorrendjében kiértékelve a szabályokat, a program nemdeterminisztikus transzformációjának legkisebb fixpontja a program minimális modellje is. Vagyis:

3.9. Állítás: Rétegzett fDATALOG P program esetén létezik olyan kiértékelési sorrend, amelyben $\text{lfp}(\text{NT}_P)$ a P minimális modellje.

Elképzelhető, hogy egy program többféleképpen rétegezhető. Kérdés, hogy vajon különböző rétegzési sorrend esetén ugyanazt a minimális modellt kapjuk-e. [CGT] kimondja, [AHV] bizonyítja is azt a tételt, amely szerint közönséges rétegzett Datalog programok esetén a kapott minimális modell független az aktuális rétegzéstől, vagyis bármely rétegzés ugyanazt a minimális modellt eredményezi. Mivel bármilyen rétegzési sorrend esetén a rétegzési sorrend csak a predikátumok közötti kapcsolatokat tükrözi, de nincs hatással a bizonytalansági szintekre, ezért fuzzy Datalog esetén is igaz az állítás:

3.3. Tétel: Legyen P rétegezhető fDATALOG program. A tetszőleges rétegzési rendben kapott legkisebb fixpont a P egyértelmű minimális modellje.

4. 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, vagy más szóval adatvezérelt következtetésről beszélünk.

Sokszor előfordulhat azonban, hogy nincs szükség a teljes kiértékelésre, hiszen egy konkrét kérdésre keressük a választ, el akarjuk dönteni egy predikátum (predikátumok) igaz vagy hamis voltát, illetve bizonytalansági fokát, vagy meg akarjuk tudni, hogy egy predikátum milyen értékek mellett igaz vagy hamis, milyen értékek mellett lesz legalább adott szinten igaz, 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. Ilyen kiértékelés esetén azonban elég nehéz a megállási feltétel vizsgálata, illetve időnként sok fölösleges kiértékelési lépést kell végrehajtanunk. Ezért harmadik stratégiaként megemlítünk egy kevert módszert.

A következőkben tárgyalt módszerek mindegyikét függvénymentes esetre dolgoztam ki, bár legtöbbjük általánosítható függvényszimbólumokat tartalmazó programokra is.

4.1. Adatvezérelt kiértékelés

A továbbiakban az egyszerűbb jelölésmód kedvéért a rákövetkezési transzformációkat T_p -vel jelölöm. Ez nem okozhat zavart, hiszen negációmentes programok esetén a két transzformáció megegyezik, negációt tartalmazó esetben pedig a továbbiakban csak a rétegezhető programokról lesz szó, amelyekre a nem determinisztikus transzformációt alkalmaztuk.

A fixpontszámítás gyakorlatilag egy iterációs eljárás, melynek algoritmus a következő:

4.1. Algoritmus:

```
Procedure bottom_up
  régi :=  $T_0$ 
  új :=  $T_P(T_0)$ 
  while régi  $\neq$  új do
    régi := új
    új :=  $T_P(\text{rég})$ 
  endwhile
endprocedure
```

A fenti eljárás azonban nem mindig tekinthető igazi algoritmusnak, hiszen előfordulhat, hogy nem terminál. (3.1. példa) Emiatt a továbbiakban csak olyan programokkal foglalkozunk, amelynek implikációs operátorai garantálják a terminálást. Az ilyen programokat *termináló programoknak*, a bennük szereplő implikációs operátorokat *termináló operátoroknak* nevezzük. A fixpontszámítási algoritmus hátránya, hogy a számolás során sokszor fölöslegesen újra és újra kiértékeli ugyanazokat a szabályokat, így ismételten előállítja ugyanazokat az alapatomokat. Emiatt célszerű oly módon egyszerűsíteni a számolást, hogy a már kiértékelt szabályt ne vegyük ismételten figyelembe akkor, ha nem várhatunk tőle újabb eredményt.

Az, hogy egy szabály adhat-e újabb eredményt, azon múlik, hogy a szabály fej-predikátumához milyen hosszú út vezet a függőségi gráfban. Ha a szabály rekurzív, vagyis ha a függőségi gráfban a fej-predikátumához vezető út kört tartalmaz, akkor a szabály nem hagyható el az algoritmus terminálása, vagyis a fixpont elérése előtt. Ha azonban a fej-predikátumhoz vezető út körmentes, akkor a szabály esetleg a terminálás előtt is elhagyható. Rendeljünk hozzá minden szabályhoz egy számértéket: a függőségi gráfban az illető szabály fej-predikátumához vezető maximális hosszúságú út hosszát. Ha az algoritmus túlhaladta ezt a lépésszámot, akkor a számítási eljárás során ez a szabály már nem ad új eredményt, vagyis elhagyható. Precízebben:

4.2. Módosított adatvezérelt kiértékelés

Legyen $P = \cup \{(r; \beta; I)\}$. Definiáljuk a $h: P \rightarrow N$ függvényt a következőképpen:
 $h(r; \beta; I) = n$, ahol n a függőségi gráfban a fej-predikátumhoz vezető leg-hosszabb körmentes út hossza;
 $h(r; \beta; I) = \infty$, ha a fej-predikátumhoz vezető út tartalmaz kört.

Értelmezzük a $T'_n = T_{P'_n}(T_{n-1})$, $n > 0$; $T'_0 = T_0$ sorozatot, ahol

$$P'_n = P - \{(r; \beta; I) \mid h(r; \beta; I) < n\}$$

A T'_n sorozatnak létezik határértéke, sőt:

4.1. Állítás: Negációmentes termináló P program esetén a T'_n sorozat véges sok lépésben eléri a határértékét, $T'(P)$ -t.

Bizonyítás:

Mivel $P'_n \subseteq P$, és a P -re vonatkozó fixpontszámítás véges sok lépésben véget ér, ezért a T'_n sorozat is véges sok lépésben eléri a határértékét. \square

4.2. Állítás: Negációmentes fDATALOG programok esetén $\text{lfp}(T_P) = T'(P)$.

Bizonyítás:

$T'(P)$ konstrukciójából adódóan $T'(P) \subseteq \text{lfp}(T_P)$.

A fordított tartalmazás belátásához tekintsünk egy tetszőleges $(A, \alpha_A) \in \text{lfp}(T_P)$ elemet. Ekkor van olyan $(r; \beta; I) \in P$, melyre $(A \leftarrow A_1, \dots, A_n; \beta; I) \in \text{ground}(r)$.

Ha $h(r; \beta; I) = k$, akkor $(r; \beta; I) \in P'_k$, így $(A, \alpha_A) \in T'_k \subseteq T'(P)$.

Ha $h(r; \beta; I) = \infty$, akkor $(r; \beta; I) \in P'_k$, $\forall k \in N$, így $(A, \alpha_A) \in T'(P)$. \square

A módosított adatvezérelt kiértékelés algoritmus:

4.2. Algoritmus:

Procedure modositott_bottom_up

$k := 1$

 régi := T_0

 új := $T_P(T_0)$

 while régi \neq új do

$k := k+1$

 régi := új

$P := P - \{(r; \beta; I) \mid h(r; \beta; I) < k\}$

```

új := Tp(régí)
endwhile
endprocedure

```

4.3. Módosított adatvezérelt kiértékelés rétegzett fDATALOG esetén

Az előzőekben ismertetett eljárás rétegzett fDATALOG esetén is alkalmazható. Itt rétegenként végezhetjük a kiértékelést, vagyis:

Legyen P egy rétegzett fDATALOG program P_1, \dots, P_n rétegzéssel. Jelölje P_i^* a P_i réteghez tartozó összes P -beli szabály halmazát, azaz az összes olyan szabály halmazát, melynek fej-predikátuma P_i -ben van.

Legyen

$$L_i = \text{lfp}(T_{P_i^*}),$$

ahol a fixpontoszámítási eljárás kiindulópontja a korábban meghatározott L_{i-1} , és $T_{P_i^*} = NT_{P_i^*} = DT_{P_i^*}$

Mivel P rétegzése miatt az i -edik réteg összes negatív literálja egy alacsonyabb szintű réteg predikátumához tartozik, ezért P_i^* kiértékelése megegyezik egy negációmentes program kiértékelésével, így igaz a következő állítás:

4.3. Állítás: L_i meghatározható az előzőekben definiált módosított adatvezérelt algoritmussal, vagyis $L_i = T'(P_i^*)$.

4.4. Célvezérelt kiértékelés

Sokszor előfordulhat, hogy nincs szükség a teljes kiértékelésre, hiszen csak egy konkrét kérdésre keressük a választ, és nem akarjuk meghatározni a program modelljét képező összes lehetséges alapatomot. Ily módon a fDATALOG program kiegészíthető egy célhalmazzal, s feladatunk a célhalmaznak megfelelő alapatomok meghatározása lesz.

Célnak egy $(Q; \alpha)$ párt nevezünk, ahol Q atom, α pedig az atom bizonytalansági szintje. Q tartalmazhat változókat is, és α is lehet változó vagy konstans. A célhalmazzal kibővített fDATALOG programot *lekérdezésnek* nevezzük. Egy

lekérdezés kiértékelése a célhalmaz összes elemének meghatározását jelenti egymástól független sorrendben.

Egy célt részcélokra bontással tudunk meghatározni. Ez azt jelenti, hogy kiválasztjuk az összes olyan szabályt, melynek fej-predikátuma illeszthető az adott cél-predikátummal, majd a szabálytörzsekben szereplő predikátumokat rész-céloknak tekintve folytatjuk a kiértékelést, amíg el nem jutunk a tényekhez. Az ilyen irányú kiértékelést nevezzük célvezérelt vagy top-down kiértékelésnek.

A továbbiakban ezt a fajta kiértékelési módot részletezzük. A kiértékelési stratégiába egy kis „heurisztikát” is beépítünk, vagyis a bizonytalansági szintek és egyéb megfontolások figyelembevételével megadunk egy célszerű kiértékelési sorrendet.

A precíz megfogalmazáshoz szükségünk van néhány fogalom tisztázására.

4.1. Definíció: Egy θ helyettesítés $x_i|t_i$ alakú helyettesítési komponensek $\{x_1|t_1, \dots, x_n|t_n\}$ véges halmaza, ahol x_i ($i = 1, \dots, n$) egymástól különböző változók, és $t_i \neq x_i$ ($i = 1, \dots, n$) term.

Az x_1, \dots, x_n változók halmazát a θ értelmezési tartományának nevezzük. Ha az összes t_i term konstans, akkor θ - t alaphelyettesítésnek nevezzük. Az üres helyettesítést ε -nal jelöljük.

Ha θ helyettesítés és t term, akkor $t\theta$ jelöli a következőképpen értelmezett termet:

$$t\theta = \begin{cases} t_i & \text{ha } t \mid t_i \in \theta \\ t & \text{egyébként} \end{cases}$$

Ha L literál, akkor $L\theta$ jelöli azt a literált, melyet úgy kapunk L -ből, hogy szimultán helyettesítjük az L -ben előforduló összes x_i változót a megfelelő t_i termmel, ha $x_i|t_i$ eleme θ -nak. $L\theta$ -t az L egy példányának nevezzük.

Ha például $L = \neg p(a, x, y, b)$ és $\theta = \{x|c, y|x\}$, akkor $L\theta = \neg p(a, c, x, b)$.

Ha $(r; \beta; I)$ egy fDATALOG szabály, akkor $(r\theta; \beta; I)$ jelöli azt a szabályt, melyet úgy kapunk, hogy az eredeti szabály összes literáljára szimultán alkalmazzuk a θ helyettesítést, és a szabálytörzsben a helyettesítés után esetleg többszörösen szereplő atomokat egyszeres multiplicitással tekintjük.

4.2. Definíció: Legyen $\theta = \{x_1|t_1, \dots, x_n|t_n\}$ és $\sigma = \{y_1|u_1, \dots, y_m|u_m\}$ két helyettesítés. A θ és σ kompozíciója az a

$$\theta\sigma = \{x_1|t_1\sigma, \dots, x_n|t_n\sigma, y_{i_1}|u_{i_1}, \dots, y_{i_k}|u_{i_k}\}$$

halmaz, ahol $\{y_{i_1}, \dots, y_{i_k}\} = \{y_1, \dots, y_m\} \setminus \{x_1, \dots, x_n\}$, és amelyből elhagyjuk a $z|z$ alakú komponenseket.

A $\theta\sigma$ kompozíciót az $(r; \beta; I)$ szabályra alkalmazva ugyanarra az eredményre jutunk, mintha először a θ helyettesítést alkalmaznánk, majd a kapott $(r\theta; \beta; I)$ példányra a σ -t.

4.3. Definíció: Ha egy L, M literálpárhoz létezik olyan θ helyettesítés, melyre $L\theta = M\theta$, akkor azt mondjuk, hogy L és M egyesíthető, és a θ helyettesítést egyesítő helyettesítésnek nevezzük.

Legyen θ és λ két helyettesítés. Azt mondjuk, hogy θ általánosabb, mint λ , ha létezik olyan σ helyettesítés, melyre $\theta\sigma = \lambda$.

Legyen L és M két literál. Az L és M legáltalánosabb egységesítője olyan egységesítő helyettesítés, amely általánosabb a többinél. L és M legáltalánosabb egységesítőjét $\text{lge}(L, M)$ -mel fogom jelölni.

A legáltalánosabb egységesítő fogalmát függvényszimbólumokat tartalmazó logikai kifejezések egységesítésére vezették be, s meghatározására különböző algoritmusok ismeretesek ([L], [P1], [P2], [U]). Mivel függvénymentes fDATALOG-gal szeretnék foglalkozni, ezért célszerű megadni egy jóval egyszerűbb, függvénymentes atomok egységesítésére szolgáló algoritmust.

Legyen $L = p(t_1, \dots, t_n)$ és $M = p'(t'_1, \dots, t'_m)$ két literál. Az $\text{lge}(L, M)$ meghatározására szolgáló függvényeljárás vagy megadja L és M legáltalánosabb θ egységesítőjét, vagy jelzi, ha az nem létezik. Az egységesítés végrehajthatóságához nyilvánvalóan elengedhetetlen, hogy $p = p'$ és $n = m$ teljesüljön.

4.3. Algoritmus:

function $\text{lge}(L, M)$

 if $p \neq p'$ or $n \neq m$ then L és M nem egyesíthető

 else

$\theta := \varepsilon$

$i := 1$

 egységesíthető := igaz

```

while  $i \leq n$  and egységesíthető do
  if  $t_i\theta \neq t'_i\theta$ 
    then if  $t'_i\theta$  változó
      then  $\theta := \theta\{t'_i\theta \mid t_i\theta\}$ 
      else if  $t_i\theta$  változó
        then  $\theta := \theta\{t_i\theta \mid t'_i\theta\}$ 
        else egységesíthető := hamis endif
      endif
    endif
     $i := i+1$ 
  endwhile
if egységesíthető then  $\text{lge}(L, M) = \theta$ 
  else L és M nem egységesíthető
endif
endif
endfunction

```

Az algoritmusból is látszik, hogy $\text{lge}(L, M) \neq \text{lge}(M, L)$. Erre az aszimmetriára különösen vigyáznunk kell a top-down kiértékelésnél.

A továbbiakhoz szükségünk lesz még egy helyettesítés vetületének és két helyettesítés összekapcsolásának fogalmára.

4.4. Definíció: A $\theta = \{x_1|t_1, \dots, x_n|t_n\}$ helyettesítés $H = \{x_{i_1}, \dots, x_{i_k}\}$ halmazra vonatkozó vetülete a $\theta_H = \{x_{i_1}|t_{i_1}, \dots, x_{i_k}|t_{i_k}\}$ helyettesítés.

4.5. Definíció: Legyen $\theta = \{x_1|t_1, \dots, x_n|t_n\}$ és $\sigma = \{y_1|u_1, \dots, y_m|u_m\}$ két helyettesítés! Ha minden olyan $x_i|t_i, y_j|u_j$ párra, amelyre $x_i = y_j$ igaz, teljesül, hogy $t_i = u_j$, akkor a θ és σ összekapcsolása a

$$\theta \otimes \sigma = \{x_1|t_1, \dots, x_n|t_n, y_1|u_1, \dots, y_m|u_m\}$$

halmaz, amelyből elhagyjuk a többszörösen szereplő komponenseket.

Ha valamely $x_i|t_i, y_j|u_j$ párra $x_i = y_j$, de $t_i \neq u_j$, akkor θ és σ összekapcsolása nem értelmezett.

A definícióból láthatóan az összekapcsolás parciális művelet. Ahhoz, hogy az összekapcsolás és a kompozíció műveletét együtt tudjuk alkalmazni, értelmeznünk kell a parciális kompozíció fogalmát is.

4.6. Definíció: A θ és σ helyettesítések parciális kompozíciója $\theta\sigma$, ha mindkét helyettesítés értelmezett, és nincs értelmezve, ha θ és σ valamelyike nem értelmezett.

Először negációmentes fDATALOG programok kiértékelésével foglalkozunk. A kiértékelést az úgynevezett *kiértékelési gráf* segítségével hajthatjuk végre. Ez egy ÉS/VAGY fa-gráf, vagyis egy speciális hipergráf, melynek minden páratlan mélységű éle n-edrendű hiperél, a hozzá tartozó n elemből álló halmazcsúccsal, minden páros mélységű éle pedig közönséges él, a hozzá tartozó egy elemű csúccsal. Pontosabban:

A kiértékelési hipergráf olyan fa-gráf, melynek gyökere a kiértékelendő cél, levelei a \odot és \ominus szimbólumok, csúcsait pedig rekurzív módon definiáljuk. A gyökér szintjét 0-nak tekintve, a gráf minden páros szintjén kiértékelésre váró részcélok, azaz megfelelő módon illesztett szabályfejek, minden páratlan szintjén kiértékelendő szabálytörzsek szerepelnek.

Legyen a $k = 2i$ -edik ($i \in \mathbb{N}$) szint egyik csúcsa a Q atom, és tegyük fel, hogy m db olyan

$$R \leftarrow R_1, \dots, R_n; \beta; I.$$

alakú szabály van, melynek feje illeszthető Q-val. Ekkor ennek a csúcsnak m db leszármazottja van. Ezek a leszármazottak

$n > 0$ esetén $R_1\theta, \dots, R_n\theta$ alakúak, ahol $\theta = \text{lge}(Q, R)$;

$n = 0$ esetén a leszármazott a \odot szimbólum.

(Ekkor a korábban bevezetett egyszerűsítő írásmód alapján az illeszthető szabály $(R; \beta)$ alakú.)

Ha egyetlen illeszthető szabályfej sincs, akkor a leszármazott csúcs a \ominus szimbólum.

Illesztéskor vigyáznunk kell a változóátnevezésre, vagyis arra, hogy az illesztett szabály törzsében szereplő változók változóidegenek legyenek az előző illesztésektől. Ez a legegyszerűbben úgy oldható meg, hogy a kiértékelési gráf szintjével indexeljük a változókat.

Címkézzük meg a gráf $Q \rightarrow R_1\theta, \dots, R_n\theta$ éleit! Az él címkéje

$n > 0$ esetén legyen a $(\theta; \beta; I)$ hármas;

$n = 0$ esetén pedig a $(\theta; \beta)$ pár, ahol β a ténypredikátum bizonytalansági szintje.

Legyen a $k = 2i+1$ -edik ($i \in \mathbb{N}$) szint egy csúcsa a Q_1, \dots, Q_n szabálytörzs!
Ekkor a csúcsból egy n -edrendű hiperél vezet a Q_1, \dots, Q_n csúcsokba.

A hiperélet nem címkézzük meg.

A lekérdezésre a kiértékelési gráf (gráfok) címkéiből kapjuk meg a választ.

A \ominus szimbólumban végződő út nem ad megoldást. Hagyjuk ki a gráfból ezeket az utakat úgy, hogy elhagyjuk az összes olyan csúcsot és élet, amely ehhez a szimbólumhoz vezet, függetlenül attól, hogy ezek a csúcsok közönséges vagy hiperéleken keresztül kapcsolódtak egymáshoz. (Vagyis ha egy hiperél egyik csúcsából vezet él a \ominus szimbólumhoz, akkor az adott hiperélhez tartozó összes csúcsot és rákövetkezőjét töröljük.) Az így kapott gráfot *keresési gráfnak* nevezzük.

A keresési gráf \ominus szimbólumokban végződő útjai mentén határozhatjuk meg a megoldásokat. Ezek uniójaként kapjuk meg az adott célra vonatkozó választ, melyet a 3.5 definícióban értelmezett bizonytalansági-szint függvény segítségével számolhatunk ki.

A \ominus szimbólumokhoz vezető tetszőleges hiperút mentén (mivel egy út hiperéletet is tartalmaz, ezért az út több levélben végződhet) határozzuk meg a θ helyettesítést a következő módon: A gráf páros szintjén lévő csúcsaihoz rendeljük hozzá rendre a leszármazott hipercsúcs kivezető közönséges éleinek címkéjében szereplő helyettesítések összekapcsolását, majd képezzük a csúcsokhoz rendelt helyettesítések parciális kompozícióját.

Ekkor a

$$(Q, \alpha)$$

kérdésre adott egyik válasz:

$$(Q\theta, \alpha_{\text{cél}}),$$

ahol $\alpha_{\text{cél}}$ az $f(I, \alpha, \beta)$ bizonytalansági-szint függvény segítségével számolható a következőképpen:

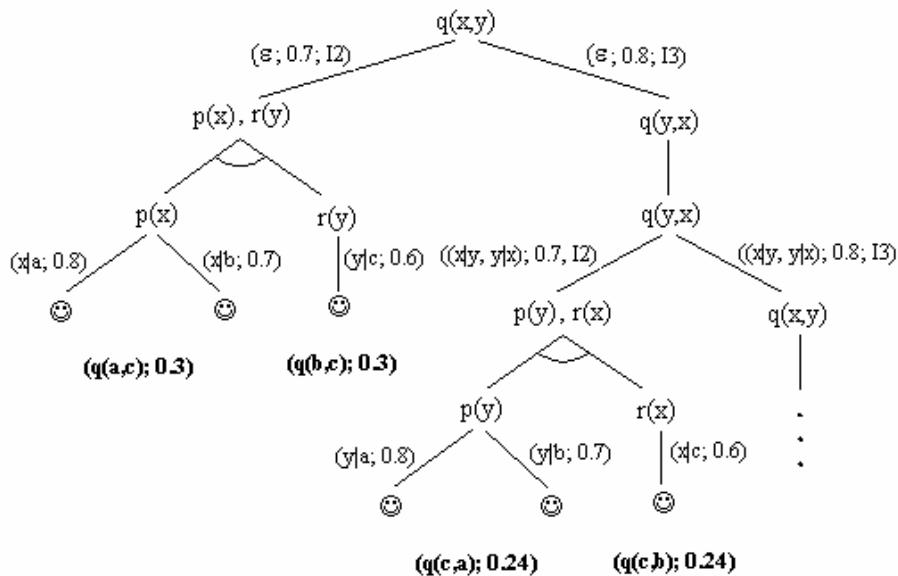
A számolást rekurzív módon definiáljuk. A levelek szülőcsúcsánál kezdjük, melyekhez a levélhez vezető él címkéjében lévő β értéket rendeljük kiinduló értéként. Innen haladunk a gyökér felé. Ha a gráf egy páratlan szintjén lévő csúcsának bizonytalansági szintje α , akkor a szülő csúcsához rendelt bizonytalansági szint legyen $f(I, \alpha, \beta)$, ahol I ; β az él címkéjében szereplő

értékek. Ha egy páratlan szinten levő csúcs leszármazottainak bizonytalansági szintje $\alpha_1, \dots, \alpha_n$, akkor a csúcshoz rendelt bizonytalansági szint legyen $\alpha = \min(\alpha_1, \dots, \alpha_n)$. A gyökérhez ilyen módon rendelt bizonytalansági szint az α_g . Előfordulhat, hogy több, a Q-ból kiinduló hiperút mentén is ugyanazt a Q θ választ kapjuk. A cél bizonytalansági szintje az összes ilyen úton kapott bizonytalansági szintek maximuma: $\alpha_{cél} = \max(\alpha_{g_1}, \dots, \alpha_{g_k})$. (A megfelelő fuzzy halmazok uniója.)

4.1. Példa: Tekintsük a következő szabályokat:

- $(p(a); 0.8) .$
- $(p(b); 0.7) .$
- $(r(c); 0.6) .$
- $q(x, y) \leftarrow p(x), r(y); 0.7; I_2 .$
- $q(x, y) \leftarrow q(y, x); 0.8; I_3 .$
- $s(x) \leftarrow q(x, y); 0.9; I_3 .$

Célunk $(q(x,y), \alpha)$ meghatározása.



Az ábrán látható ÉS/VAGY gráf alapján a lekérdezés eddigi eredménye:

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

Kérdés, hogy vajon az ábra jobboldali ágát tovább folytatva nem módosul-e ez az eredmény. Látható, hogy itt az eredeti lekérdezés ismétlődik meg, új atomot

tehát nem kaphatunk, viszont elképzelhető, hogy módosul a bizonytalansági szint értéke. Példaként tekintsük a $q(a,c)$ esetet. Mivel a $q(x,y)$ kérdés részcélként ugyanúgy értékelhető ki, mint célként, ezért a $q(x,y)$ csúchoz $(x|a, y|b)$ helyettesítés esetén $\alpha = 0.3$ bizonytalansági érték adódik. Innen tovább számolva a gyökér felé az adott I_3 operátorhoz tartozó bizonytalansági függvény alapján, a gyökérhez $\alpha_g = 0.3 \cdot 0.8 \cdot 0.8 = 0.192$ érték adódik, amely nem változtatja meg az eddig elért bizonytalansági szint értékét, 0.3-t. Ha azonban a $q(x,y)$ -ra vonatkozó szabályban az I_3 operátort kicseréljük I_5 -re, akkor $\alpha_g = 0.8$ értéket kapunk, amely befolyásolja az eddigi eredményt, és $q(a,c)$ 0.8-es szinten kerül be a válaszalmazba.

Nem nehéz végiggondolni, hogy a 3.1. példa nem termináló programja, és a hozzá megadott $(p(a), \alpha)$ cél esetén a keresési gráf minden újabb ága módosítja a célhoz tartozó bizonytalansági szint értékét. Belátható azonban, hogy véges kiértékelési gráf esetén az adatvezérelt és a célvezérelt stratégia ugyanazt az eredményt adja. Pontosabban:

4.1. Tétel: Adott célhalmaz és véges kiértékelési gráf esetén a célvezérelt kiértékelési algoritmus ugyanazt a megoldást adja, mint a fixpontkeresés.

Bizonyítás:

A bottom-up és a top-down kiértékelés ekvivalenciáját a kiértékelési gráf mélységére vonatkozó indukcióval bizonyítjuk.

Először tegyük fel, hogy a kiértékelési gráf mélysége egy, vagyis a gyökér összes leszármazottja a \odot vagy a \ominus szimbólum. Ez úgy lehetséges, ha a célpredikátumhoz vagy egyáltalán nem illeszthető szabályfej, vagy csak tényállítás illeszthető hozzá. Az első esetben az adott kérdésre egyik kiértékelési stratégia alapján sincs válasz. A második esetben mindkét stratégia szerint ugyanaz a tényállítás a válasz.

Indukciós feltevésként tegyük fel, hogy minden, legfőljebb n hosszúságú utat tartalmazó kiértékelési gráf esetén igaz az állítás!

Tekintsünk olyan kiértékelési gráfot, amelynek leghosszabb útvonala $n + 1$ hosszúságú! Vizsgáljuk meg a gráf második szintjén lévő részcélokat! Ezek kiértékelési gráfja legfőljebb $n-1$ mélységű, vagyis igaz rájuk az indukciós

feltevés. A célállítás bottom-up módon csak ezekből a részcélokból érhető el. Egy-egy hiperél mentén bottom-up módon följebb lépve az első szintre, megkapjuk azon szabályok szabálytörzsét és a szabálytörzs bizonytalansági szintjét, amelyekből az első szintre mutató élek címkéje, azaz az alkalmazandó szabályok bizonytalansági szintje, implikációs operátora és a megfelelő helyettesítés segítségével megkapjuk a keresett választ.

Tehát az indukciós feltevés szerint igaz az állítás tetszőleges véges mélységű kiértékelési gráfra. □

A gráf kiértékeléséhez megadunk egy általános algoritmust, amely az adott program és az ismert célhalmaz (célok) esetén megadja a válaszalmazt.

Az algoritmus két egymást hívó eljárásra épül, melyek egyike egy célt vagy részcélt értékel ki, a másik pedig egy szabálytörzset.

A célkiértékelő eljárás az összes illeszhető szabály esetén meghatározza az illesztett törzset, és ennek kiértékelésével megadja a célra adott választ, míg a szabálykiértékelő eljárás a szabálytörzs rész céljainak kiértékelésével megadja a törzshöz tartozó illesztést és a szabálytörzs bizonytalansági szintjét.

A célkiértékelés során az illeszhető szabályok sorrendjét, a szabálykiértékelő eljárás során pedig a részcélok kiválasztásának sorrendjét egy-egy kiválasztási függvény segítségével határozzuk meg. A speciális szimbólumokat (\odot , \ominus) nem vesszük be a kiértékelendő részcélok halmazába, hiszen ezek már nem értékelhetőek ki. A \ominus szimbólum esetén nem volt illeszhető szabály, azaz az illeszhető szabályok R halmaza üres, míg a \odot szimbólum esetén az illesztés után kaptunk üres csúcsot, s ebben az esetben azonnal meghatározhatjuk a választ.

Célszerű a helyettesítések összekapcsolását is „top-down” módon megoldani, vagyis nem a részcélok független kiértékelése után végezni el az összekapcsolást, hanem egy „oldalsó információátadás”-sal előre leszűkíteni a vizsgált gráf méretét. Ez azt jelenti, hogy egy rész cél kiértékelésekor válaszként kapott helyettesítést azonnal alkalmazhatjuk a szabálytörzs többi tagjára, evvel esetenként jelentősen lecsökkentve a vizsgálandó utak számát.

Mivel egy rész cél kiértékelésekor az egységesítés során esetleg olyan változókat is helyettesítenünk kell, amelyek nem szerepelnek a rész cél változói között,

ezért elég, ha a kiértékelés eredményeként a kapott illesztésnek csak a rész-cél változóira vonatkozó vetületét tekintjük.

Az esetenként szükséges változóátnevezést a *helyettesítőterm-halmaz* fogalmának bevezetésével oldjuk meg. A $\theta = \{x_1|t_1, \dots, x_n|t_n\}$ helyettesítés helyettesítőterm-halmaza a $\{t_1, \dots, t_n\}$ halmaz.

4.4. Algoritmus:

Kiértékelés:

begin

 megoldáshalmaz := \emptyset

 while nem_üres(célhalmaz) do

 cél := eleme (célhalmaz)

 célhalmaz := célhalmaz - {cél}

 célválasz := \emptyset

 céلكiértékelés (célatom, célválasz)

 while nem_üres (célválasz) do

$(\theta, \alpha\text{cél}) := \text{eleme}(\text{célválasz})$

 célválasz := célválasz - $\{(\theta, \alpha\text{cél})\}$

 megoldáshalmaz := megoldáshalmaz $\cup \{(\text{célatom}\theta, \alpha\text{cél})\}$

 endwhile

 endwhile

end

Procedure céلكiértékelés (célatom, célválasz)

 célváltozó := {a célatom változóinak halmaza}

 R := $\{(r; \beta; I) \mid \text{szabályfej}(r) \text{ illeszthető a célatommal}\}$

 if R = \emptyset then return

/ befejeződik, ha már nincs illeszthető szabály */*

 while nem_üres(R) do

$(r; \beta; I) := \text{szabályválasztás}(R)$

 R := R - $\{(r; \beta; I)\}$

 törzs := szabálytörzs(r)

$\theta := \text{lge}(\text{célatom}, \text{szabályfej}(r))$

/ legáltalánosabb egységesítő helyettesítés */*

 for all változó $\in r$ do

```

    if változó ∈ helyettesítőterm( $\theta$ )
        then változó := újnév(változó)
            /* esetleges változóátnevezés */
    endfor
    törzs := törzs $\theta$ 
        /* a helyettesítés alkalmazása a szabálytörzsre */
     $\alpha$ törzs := 1
     $\theta$ törzs :=  $\varepsilon$ 
        /* kezdőértékek a szabálykiértékeléshez:
            $\alpha$ törzs a törzs bizonytalansági foka
            $\theta$ törzs a törzsre alkalmazandó helyettesítés */

    if törzs =  $\emptyset$  then célválasz := célválasz  $\cup$  {( $\theta$ ,  $\beta$ )}
        /* tényt illesztünk */
    else szabálykiértékelés(törzs, $\alpha$ törzs, $\theta$ törzs,célválasz,célváltozó,I, $\beta$ )
        /* szabály illesztése esetén meghívjuk a szabálykiértékelést */
    endif
endwhile
endprocedure

```

Procedure szabálykiértékelés(törzs, α törzs, θ törzs,célválasz,célváltozó,I, β)

```

    atom := atomválasztás(törzs)
    új $\theta$ törzs := törzs - {atom}
    válasz :=  $\emptyset$ 
    célkiértékelés (atom, válasz)
        /* kiértékeljük a szabálytörzs egy atomját */
    if válasz =  $\emptyset$  then return
        /* ha van egy kiértékelhetetlen atom, akkor megszakad
           az eljárás */

    while nem_üres(válasz) do
        ( $\theta$ ,  $\alpha$ atom) := eleme(válasz)
            /* minden lehetséges válasszal kiértékeljük a törzs
               további részét */
        válasz := válasz - {( $\theta$ ,  $\alpha$ atom)}
        új $\theta$ törzs :=  $\theta$ törzs $\theta$ 
    endwhile

```

```

/* a maradék törzsre alkalmazandó helyettesítés kezdőértéke a
   törzsre eddig alkalmazott helyettesítés és a vizsgált atomra
   kapott helyettesítés kompozíciója */
 $\alpha$ újtörzs := min( $\alpha$ törzs,  $\alpha$ atom)
if újjtörzs  $\neq \emptyset$  then
    újjtörzs := újjtörzs $\theta$ 
    /* „oldalsó” információátadás */
szabálykiértékelés(újjtörzs, $\alpha$ újtörzs, $\theta$ újtörzs,célválasz,célváltozó,I, $\beta$ )
    /* ezzel az egy válasszal végigértékeljük a törzs további
       részét, így lehet elérni, hogy mindent mindennel párosítsunk,
       de ne legyenek fölösleges párosítások */
endif

if újjtörzs =  $\emptyset$  then
     $\theta$  := vetület( $\theta$ törzs, célváltozó)
    célválasz := célválasz  $\cup \{(\theta, f(I, \alpha$ törzs,  $\beta))\}$ 
    /* ha végig kiértékeljük a szabálytörzset, akkor megkaptuk a
       célra alkalmazandó helyettesítést, illetve a cél bizonytalansági fokát */
endif
endwhile
endprocedure

```

Megjegyzés: Mivel negációmentes fDATALOG esetén a determinisztikus és a nemdeterminisztikus szemantika megegyezik, ezért a top-down kiértékelés sorrendje (vagyis az illesztendő szabályok és a kiértékelendő részcélok kiválasztási sorrendje) lényegtelen. Ez a sorrend azonban befolyásolhatja az algoritmus hatékonyságát.

A válaszadás során minden – a célatomnak megfelelő – alapatomhoz meghatároztunk egy bizonytalansági szintet. Ha a $(Q; \alpha)$ célban szereplő bizonytalansági szint egy változó, akkor a megoldásban ez a változó az így meghatározott értéket veszi fel. Ha az α konstans, akkor az algoritmus végrehajtásakor kapott atom csak akkor kerül be a megoldáshalmazba, ha a kiszámított bizonytalansági szintje nagyobb α -nál vagy egyenlő vele. Ebben az esetben azonban fölösleges figyelembe venni a program összes szabályát, elég csak azokkal

foglalkozunk, melyek bizonytalansági szintje nagyobb vagy egyenlő α -nál. Ily módon a kiértékelési gráf mérete csökkenthető.

Mint az 4.1. példa is mutatja, egészen egyszerű esetekben is előfordulhat, hogy a célvezérelt kiértékelés nem terminál. Ennek oka a rekurzióban keresendő, hiszen nem rekurzív predikátumok kiértékelése véges sok lépésben befejeződik. Ha azonban a rekurzív predikátumokhoz hozzárendelünk egy, a kiértékelés mélységére vonatkozó korlátot, akkor – terminál, vagyis megfelelő implikációs operátorokkal rendelkező programok esetén – az eljárás megállítható. A korlát a következő módon határozható meg:

Jelölje h a függőségi gráfban az illető predikátumot tartalmazó körök maximális hosszát, t pedig az illető predikátumba vezető körmentes utak maximális hosszát. Vezessük be a rekurziós távolság fogalmát. Ez az az érték, ahány lépésben bottom-up kiértékeléssel megkapjuk az illető atomra vonatkozó fixpontot. A rekurziós távolság függ a program konstansainak számától, és természetesen függ a vizsgált predikátum „tartalmától”. Például az

$$u(x,y) \leftarrow e(x,z), u(z,y); \beta_1; I.$$

$$u(x,y) \leftarrow e(x,y); \beta_2; I.$$

program esetén, ahol az e ténypredikátum, és a program konstansainak száma c , az $u(x,y)$ atom rekurziós távolsága $c-1$.

Jelöljük az predikátumhoz tartozó rekurziós távolságot r -rel! Ekkor a mélységi korlát $k=2h(r-1)+2t+1$.

4.4. Állítás: Egy termináló P program minden egyes rekurzív predikátumához hozzárendelve az előzőekben értelmezett mélységi korlátot, a top-down eljárás terminál, és megadja a célnak megfelelő összes megoldást.

Bizonyítás:

Mivel a célkiértékelést visszavezettük részcélok kiértékelésére, ezért elegendő megmutatni azt, hogy egyetlen rekurzív atom esetén helyes megoldást kapunk.

Ha a függőségi gráfban egy predikátumhoz nem juthatunk el körmentes úton is, akkor az illető szabály kiértékelhetetlen, vagyis az illető predikátumhoz tartozó alapatomok nem kerülnek be a fixpontba. Ezért elég csak azokkal az esetekkel foglalkoznunk, ahol létezik ilyen út.

Mivel a kiértékelési gráfot két különböző lépésből – a szabálytörzs meghatározásából és a törzs részcélokra bontásából – álló lépéssorozattal építhetjük fel, ezért ha a függőségi gráfban t hosszúságú úton jutunk el a vizsgált predikátumhoz, akkor itt ez az út kétszer olyan hosszú lesz, s ehhez adódik még a végszimbólumokba vezető él hossza.

A korlátban szereplő $2t$ hosszúságú úton a vizsgált atomtól tehát egy ténypredikátumhoz tartozó atomhoz jutunk, amelyet az adott ténypredikátumoknak megfelelően ki tudunk értékelni.

A kiértékelési gráfban a szóban forgó atom – esetleg más változókkal – $2h$ lépéssel mélyebben ismét előfordul. Az újbóli kiértékelésnek csak akkor van értelme, ha evvel az eredeti kérdésre újabb választ kapunk. Ez viszont a rekurziós távolságnál többször – a terminálási feltételnek eleget tévő implikációs operátorok esetén – nem fordulhat elő. Mivel az első ismétlődéskor már a második rekurziós szintre jutottunk, ezért elég, ha csak $r-1$ -szer engedjük meg az ismétlődést.

Mivel egy rész cél kiértékelésekor az összes lehetséges szabályfejjel illesztjük az illető részelt, ezért megkapjuk az összes lehetséges adott rekurziós mélységű választ. Mivel a megadott korlátnál mélyebben már nem kapunk újabb eredményt, ezért ezek az ágak elhagyhatóak. (Újabb alapatomot nem, de újabb bizonytalansági szintet kaphatunk, ez a bizonytalansági szint azonban kisebb az előzőnél, így az unióképzésben értéke felülíródik.)

Minden egyes rész célhoz bevezetve a megfelelő korlátot, az algoritmus terminál, és megadja a célnak megfelelő összes megoldást. \square

Megjegyzés: A rekurziós távolság nem minden esetben határozható meg olyan egyszerűen, mint az említett példában, azonban nem lehet nagyobb c^n -nél, ahol c a program konstansainak, n pedig az atom argumentumainak száma.

Ha a top-down algoritmus célkiértékelési eljárását kiegészítjük egy mélységi korlát-vizsgálattal, akkor az algoritmus terminál. Az algoritmus termináló implikációs operátorok esetén mélységi korlát bevezetése nélkül is átalakítható úgy, hogy minden esetben termináljon. Ebben az esetben nem rekurzív, hanem iteratív módon célszerű számolnunk, azaz egy rész cél egyetlen megtalált válaszával azonnal „végigmenni” a szabálytörzs többi rész célján, ily módon

azonnal megadni egy, a célra vonatkozó választ. Ezt az iterációt addig folytathatjuk, amíg a válaszalmaz változik.

4.5. Célvezérelt kiértékelés speciális implikációs operátor esetén

A top-down kiértékelési stratégiában a kiértékelés elvégzéséhez egy hipergráfot kellett vizsgálnunk, ezen kellett megkeresnünk a megoldást. A hiperéleket azért kellett megkülönböztetni a közönséges élektől, mert a szabálytörzs bizonytalansági szintjét a benne szereplő atomok bizonytalansági szintjének minimumaként határozzuk meg, a közönséges élek mentén viszont az adott implikációs operátortól függő módon számolunk. Az egyik speciális implikációs operátor, az

$$I(x, y) = \begin{cases} 1 & \text{ha } x \leq y, \\ y & \text{egyébként} \end{cases}$$

esetén azonban a bizonytalansági szint meghatározására szolgáló $f(I, \alpha, \beta)$ függvény értéke $f(I_1, \alpha, \beta) = \min(\alpha, \beta)$ módon számolható, azaz nincs különbség a szabálytörzs és a rész cél bizonytalansági szintjének kiszámítása között. Ezért abban az esetben, ha a program összes szabályában az I_1 implikációs operátor szerepel, akkor a hipergráf helyett elég közönséges gráfot vizsgálnunk.

A közönséges kiértékelési gráf egy olyan fa-gráf, melynek gyökere a kiértékelendő cél, levelei a \odot és \ominus szimbólumok, csúcsait pedig rekurzív módon definiáljuk.

Legyen a k -adik csúcs Q_1, \dots, Q_n , ahol Q_1, \dots, Q_n atomok. Ha valamely $i \in N$ esetén m db olyan

$$R \leftarrow R_1, \dots, R_s; \beta; I_1$$

alakú szabály van, melynek feje illeszthető valamelyik Q_i -vel, akkor a k -adik csúcsnak m db leszármazottja van, és ezek a leszármazottak

$s > 0$ esetén $Q_1\theta, \dots, Q_{i-1}\theta, R_1\theta, \dots, R_s\theta, Q_{i+1}\theta, \dots, Q_n\theta$ alakúak, ahol $\theta = lge(Q_i, R)$;

$s = 0, n > 1$ esetén $Q_1\theta, \dots, Q_{i-1}\theta, Q_{i+1}\theta, \dots, Q_n\theta$ alakúak, ahol $\theta = lge(Q_i, R)$;

$s = 0, n = 1$ esetén a leszármazott a \odot szimbólum.

Ha egyetlen Q_i sem illeszthető egyetlen szabályfejjel sem, akkor a leszármazott csúcs a \ominus szimbólum.

A levelekhez vezető utakon címkézzük meg a gráf éleit! Tegyük fel, hogy az út k -adik csúcsából az $(r; \beta; I_1)$ szabály alkalmazásával jutottunk az út $k+1$ -edik

csúcsába! Ekkor a $k \rightarrow k+1$ él címkéje legyen a $(\theta; \beta)$ pár, ahol θ a $k \rightarrow k+1$ élen alkalmazott legáltalánosabb egységesítő helyettesítés.

A lekérdezésre ebben az esetben is a kiértékelési gráf (gráfok) címkéiből kapjuk meg a választ. A \ominus szimbólumban végződő út nem ad megoldást, míg a \odot -ban végződő utak alapján meghatározott megoldások uniója megadja az adott célra vonatkozó választ.

A

$$(Q, \alpha)$$

kérdésre adott válasz:

$$(Q\theta, \alpha_{\text{cél}}),$$

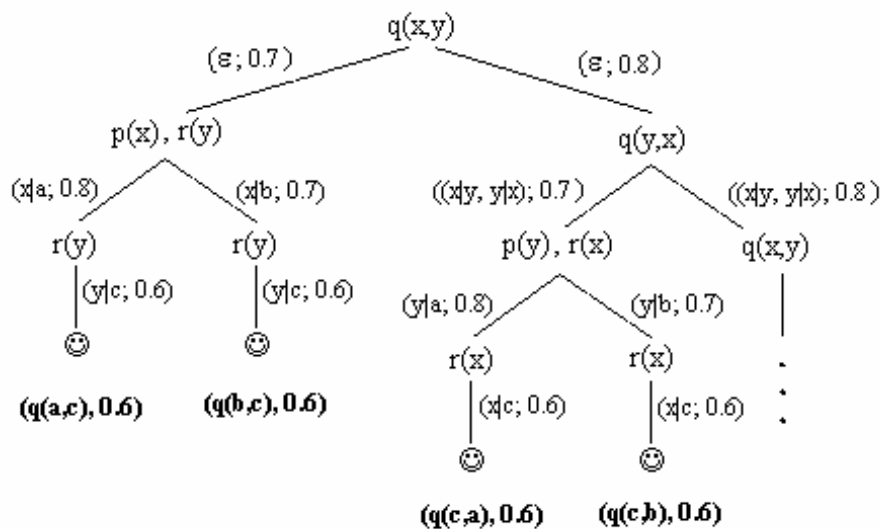
ahol θ a \odot szimbólumhoz vezető úton alkalmazott helyettesítések kompozíciója, α_g az út éleihez tartozó bizonytalansági szintek minimuma, és $\alpha_{\text{cél}}$ az összes ilyen, vagyis θ helyettesítést eredményező út α_g értékének maximuma.

4.2. Példa: Tekintsük a 4.1. példa szabályait úgy, hogy mindegyik implikációs operátorát I_1 -re változtatjuk:

$$\begin{aligned} & (p(a); 0.8) . \\ & (p(b); 0.7) . \\ & (r(c); 0.6) . \\ & q(x, y) \leftarrow p(x), r(y); 0.7; I_1 . \\ & q(x, y) \leftarrow q(y, x); 0.8; I_1 . \\ & s(x) \leftarrow q(x, y); 0.9; I_1 . \end{aligned}$$

Célunk továbbra is $(q(x,y), \alpha)$ meghatározása.

A kiértékelési gráf:



A gráfról leolvasható a feladat megoldása:

$$\{(q(a,c), 0.6); (q(b,c), 0.6); (q(c,a), 0.6); (q(c,b), 0.6)\}$$

Mivel a kapott eredmény bizonytalansági szintje minden esetben kisebb 0.8-nál, ezért a jobboldali befejezetlen ág nem ad új megoldást.

Ebben a speciális esetben is megadunk egy általános algoritmust, amely az adott program és az ismert célhalmaz (célok) esetén megadja a válaszalmazt. A megoldást a gráf csúcsainak valamely sorrendben való kiterjesztésével határozzuk meg. Azokat a csúcsokat lehet kiterjeszteni, melyeknek valamelyik atomja illeszthető legalább egy szabályfejjel. Egy csúcs kiterjesztése során meghatározzuk az összes közvetlen rákövetkezőjét. A kiterjeszthető csúcsok halmazát nyílt halmaznak nevezzük, s egy kiválasztási függvény segítségével határozzuk meg a halmaz éppen kiterjesztendő elemét. Ugyancsak egy kiválasztási függvény határozza meg az illeszthető szabályok sorrendjét, és azt is, hogy egy-egy csúcsot melyik atom szerint terjesszünk ki. A speciális szimbólumokat (☺, ☹) nem vesszük be a nyílt halmazba, hiszen ezek már nem terjeszthetőek tovább. Ugyanúgy, ahogy az általános esetben, a ☹ szimbólum esetén nem volt illeszthető szabály, míg a ☺ szimbólum esetén az illesztés után azonnal meghatározhatjuk a választ.

4.5. Algoritmus:

Kiértékelés:

```
begin
  cél := eleme(célhalmaz)
  válasz := ∅
  while nem_üres (célhalmaz) do
    kiértékel(célatom, válasz)
    célhalmaz := célhalmaz - {cél}
  endwhile
end
```

Procedure kiértékel(célatom, válasz)

```
  célváltozó := {a célatom változóinak halmaza}
  nyílt := {(célatom, ε, 1)}
  while nem_üres (nyílt) do
    (csúcs, illesztés, α) := csúcsválasztás(nyílt)
    /* a nyílt halmazból kiválasztunk egy kiterjesztendő csúcsot */
    nyílt := nyílt - {(csúcs, illesztés, α)}
    /* a vizsgált csúcs kikerül a nyílt halmazból */
    atom := atomválasztás(csúcs)
    /* a kiterjesztendő csúcsból kiválasztunk egy tetszőleges
    atomot */
    kiterjeszt(atom, nyílt, csúcs, illesztés, α, válasz)
  endwhile
endprocedure
```

Procedure kiterjeszt(atom, nyílt, csúcs, illesztés, α, válasz)

```
  R := {(r; β; I) | szabályfej(r) illeszthető az atommal }
  if R = ∅ then return
    /* ha nincs illeszthető szabály, akkor megáll */
  while nem_üres (R) do
    (r; β; I) := szabályválasztás (R)
    R := R - {( r; β; I)}
    Q := szabálytörzs(r)
    új_csúcs := csúcs - atom
    /* a kiterjesztendő csúcsból elhagyjuk az éppen vizsgált
    atomot */
```

```

for all változó  $\in$  r do
  if változó  $\in$  helyettesítőterm(illesztés) or változó  $\in$  új_csúcs
    then változó := újnév(változó)
    /* esetleges változóátnevezés */
  endfor
 $\theta :=$  lge(atom, szabályfej(r))
  /* az atom és a kiválasztott szabályfej legáltalánosabb
  egységesítője */
új_csúcs :=  $Q\theta \cup$  új_csúcs $\theta$ 
  /* alkalmazzuk a helyettesítést a csúcs maradék részére, és
  hozzáfűzzük a szabálytörzs megfelelően illesztett atomjait */
szabályillesztés := vetület(illesztés $\cdot\theta$ , célváltozó)
 $\alpha := \min(\alpha, \beta)$ 
if új_csúcs $=\emptyset$  then válasz:= válasz $\cup$  {(célatom $\cdot$ szabályillesztés, $\alpha$ )}
  /* ha eljutottunk egy kiértékelési út végére, akkor a célatomra
  alkalmazva az illesztést, egy célválaszt kapunk */
else nyílt := nyílt  $\cup$  (új_csúcs, szabályillesztés,  $\alpha$ )
  /* különben az új csúcs a megfelelő információkkal bekerül a
  nyílt csúcsok halmazába */
endwhile
endprocedure

```

4.2. Tétel: Adott célhalmaz és véges kiértékelési gráf esetén a fenti kiértékelési algoritmus ugyanazt a megoldást adja, mint a fixpontkeresés.

Bizonyítás:

Az adatvezérelt és a célvezérelt kiértékelés ekvivalenciáját ebben az esetben is a kiértékelési gráf mélységére vonatkozó indukcióval bizonyítjuk.

Először tegyük fel, hogy a kiértékelési gráf mélysége egy, vagyis a gyökér összes leszármazottja a ☺ vagy a ☹ szimbólum! Ez úgy lehetséges, ha a célpredikátumhoz vagy egyáltalán nem illeszthető szabályfej, vagy csak tényállítás illeszthető hozzá. Az ilyen gráf levele akkor és csak akkor lehet a ☺ szimbólum, ha a tényhalmazból adatvezérelt levezetéssel, a megfelelő illesztéssel előállítható a cél.

Indukciós feltevésként tegyük fel, hogy minden, legfőljebb n ($n \in \mathbb{N}$) hosszúságú utat tartalmazó kiértékelési gráf esetén igaz az állítás.

Tekintsünk olyan kiértékelési gráfot, amelynek leghosszabb útvonala $n + 1$ hosszúságú. Ha ennek az útnak a végpontja a \ominus szimbólum, akkor a tényekből nem vezet ide út.

Ha a végszimbólum \ominus , akkor az előző szinten tényatom van. Ehhez akkor és csakis akkor találunk megfelelő illesztést, ha az illesztéssel kapott alapatom szerepel a tények között. Ehhez a tényatomhoz vezető él helyett vezessünk be a kiértékelési gráfba k darab \ominus -ban végződő élet, ha a tények közül k darab illeszthető ehhez a ténypredikátumhoz.

Ha a tényatomhoz vezető él címkéje $(\theta; \beta)$ volt, és az innen továbbvezető él címkéje $(\sigma; \beta')$, akkor az új él címkéje legyen $(\theta\sigma; \min(\beta, \beta'))!$

Ez az átalakítás csak akkor lehetséges, ha a megfelelő tényekből adatvezérelt következtetéssel származtathatjuk az n -edik szinten lévő ténypredikátumot. Mivel az ilyen módon átalakított gráf mélysége n lesz, ezért az indukciós feltevés szerint igaz az állítás tetszőleges véges mélységű kiértékelési gráfra. \square

Az előzőekben megadott algoritmus az általános esethez hasonlóan, termináló implikációs operátorok esetén sem minden esetben terminál. Alkalmas mélységi korlát bevezetésével azonban ebben az esetben is megoldható ez a probléma.

Az általános algoritmushoz megkonstruált mélységi korlát most nem alkalmazható, mert ez a gráf általában mélyebb és keskenyebb a hipergráfnál. Ez abból adódik, hogy az általános esetben a szabálytörzs kiértékelendő rész céljai ugyanazon a szinten voltak, míg a speciális esetben egymás alatti szintre kerültek.

A szabálytörzsben szereplő predikátumokhoz az általános esethez hasonlóan itt is hozzárendelhető az 4.4. állításban megadotthoz hasonló mélységi korlát annyi különbséggel, hogy esetünkben a speciális kiértékelési gráf tulajdonságaiból adódóan nem kell kétszeres úthosszakkal számolnunk, vagyis az i -edik rész célhoz a

$$k_i = h_i (r_i - 1) + t_i + 1$$

korlát rendelhető, s az

$$R \leftarrow R_1, \dots, R_m; \beta; I_1.$$

alakú szabály rekurzív fejpredikátumához rendelt korlát :

$$k = 1 + \sum_{i=1}^m k_i .$$

Az 4.4. állításhoz hasonlóan belátható, hogy :

4.5. Állítás: Mélységi korlát bevezetésével a speciális top-down eljárás terminál, és megadja a célnak megfelelő összes megoldást.

4.6. Célvezérelt kiértékelési heurisztikák

A top-down algoritmus során az általános és a speciális esetben is többször alkalmaztunk kiválasztási függvényeket. Bizonyos helyzetekben érdektelen a választási sorrend, más esetekben viszont ügyes választással lényegesen hatékonyabbá tehető az eljárás. A továbbiakban külön-külön tárgyaljuk a két algoritmus kiértékelési heurisztikáit.

Tekintsük először az általános algoritmust! Az algoritmus végrehajtása során lényegtelen, hogy milyen sorrendben értékeljük ki a célhalmaz elemeit, s az sem befolyásolja az eredményt, hogy az egy-egy rész cél kiértékelésekor kapott válasz halmaz elemeivel milyen sorrendben számolunk tovább. Lényeges lehet azonban az illeszthető szabályok és a kiértékelendő rész célok sorrendje. Ezt kihangsúlyozandó, az algoritmusban az első két esetben a kiválasztást az „eleme” függvény segítségével, míg a másik két esetben a „szabályválasztás”, illetve az „atomválasztás” segítségével oldottuk meg.

Érdeemes a következő kiértékelési heurisztikákat alkalmazni:

1. *sorrendi heurisztika:*

- A szabályok közül célszerű előbb a tényekre vonatkozóakat alkalmazni, illetve azokat, melyek fejpredikátuma nem szerepel a függőségi gráf egyetlen kört tartalmazó útján sem, vagyis amely szabályok nem játszanak szerepet egyetlen rekurzív programrészletben sem.
- A rész célok közül pedig célszerű előre venni azokat, amelyek argumentumában konstans szerepel.

2. vágó heurisztika:

Az eljárás során előfordulhat, hogy többször is ki kell értékelnünk ugyanazt a részcélt. Nem törölhetjük automatikusan az egyiket, hiszen a részcélnak megfelelő alapatom más úton esetleg más szinten kerül be a megoldásba. Ha azonban ugyanazon az ágon fordul elő ismétlődés, akkor a mélyebben fekvő atom esetleg levágható, ha ezen az ágon már nem kaphatunk új eredményt. Ennek feltétele a részcélhoz vezető él címkéjében lévő implikációs operátor megfelelő tulajdonsága: $f(I, \alpha, \beta) \leq \alpha$.

3. mélységi heurisztika:

A programban szereplő predikátumokra vonatkozó konkrét ismeretek alapján egyes becslés adható a mélységi korlátra. Az algoritmus módosítható iteratíván mélyülő korlát bevezetésével is, azonban ezzel az elképzeléssel most nem foglalkozom.

A speciális algoritmus esetén szintén lényegtelen a kiértékelendő célok kiválasztási sorrendje. Mivel egy csúcs kiterjesztése során meg kell határoznunk az összes leszármazottat, vagyis illesztenünk kell a megfelelő atomot az összes lehetséges szabályfejjel, ezért ebben az esetben a szabályok kiválasztási sorrendje is lényegtelen. A kiterjesztendő csúcs kiválasztási sorrendje és a kiterjesztendő csúcs esetén a kiterjesztendő atom kiválasztásának sorrendje azonban befolyásolható, és egyes választás esetén a kiértékelési eljárás egyszerűsíthető.

Az eljárás során előfordulhat, hogy a nyílt halmazba többször is bekerül ugyanaz a formula. Nem törölhetjük automatikusan az egyiket, hiszen a formula más úton esetleg más szinten kerül be a halmazba. Azt azonban megtehetjük, hogy mindkét formulához kiszámítjuk az aktuális szintértéket (az adott formulát tényként kezelve a szintszámításnál kiinduló értéként egyet választva), és a kisebb szintértékűt kihagyjuk a nyílt halmazból. Evvel a lépéssel nem csorbítjuk a megoldást, hiszen azt a válaszok uniójaként kapjuk, s az unióképzés szabályai szerint azonos atomok esetén a nagyobb szintértéket vesszük figyelembe. Ilyen módon azonban esetleges végtelen ágakat vághatunk le.

Az adott csúcsot a formula valamelyik atomja szerint terjesztjük ki. Az atomok sorrendjének egyes megválasztásával itt is egyszerűsíthető a kiértékelés. Ha van, akkor célszerű először ténypredikátumokat választani az atomok közül.

Ezek esetén ugyanis hamar eldönthető, hogy található-e hozzájuk a célnak megfelelő illesztés. Ugyancsak célszerű előre venni azokat az atomokat, amelyek argumentumában minél több konstans szerepel (azaz amelyekben kevés a változó). Így ugyanis szűkül a szóba jöhető illesztések köre.

4.7. Az általános célvezérelt kiértékelés rétegzett fDATALOG esetén

Mint láttuk, negáció- és függvénymentes fDATALOG bottom-up és top-down kiértékelési stratégiája adott célhalmaz esetén ugyanazt a megoldást adja. A bottom-up technikát kiterjesztettük rétegzett fDATALOG esetre is. Most a top-down stratégiát értelmezzük, és megadjuk a kiértékelési algoritmus módosított változatát.

Nehézséget a negált predikátumok kiértékelése jelent, hiszen ilyen esetben nem képezhetjük a közvetlen utódokat, pontosabban a közvetlen leszármazottakat más módon kell meghatározni. Bottom-up kiértékelés során a negált predikátumok kezelésére a rétegzés nyújtott segítséget, hiszen itt a rétegzés sorrendjében kiértékelve a szabályokat, már biztos ismeretünk volt a negált predikátumról, mire az valamelyik szabály törzsében sorra jutott.

Rétegzett fDATALOG esetén a szabályfej predikátuma legalább olyan magas rétegen van, mint a törzspredikátumok. Ez azt jelenti, hogy ha a top-down kiértékeléskor a magasabb rétegek felől közelítünk az alacsonyabbak felé, akkor a kiértékelési gráfban a szülőcsúcs rétegszintje magasabb vagy egyenlő az utódok rétegszintjénél. Ebből adódik, hogy a kiértékelés során a bizonytalansági szint kiszámítását a legalsó rétegnél kezdjük. Ezt az észrevételt használjuk fel a negált predikátumok kezelésére. Ha egy rész cél negálva szerepel, akkor a kiértékelést ugyanúgy végezhetjük, mint negációmentes esetben, de jelöljük meg ezt a rész célt! A megjelölést a bizonytalansági szint számításakor vesszük figyelembe. Ekkor ugyanis megjelölt atomhoz érkezve, ha az eddig számolt bizonytalansági szint α volt, akkor az $1-\alpha$ értékkel számolunk tovább.

4.8. Kombinált kiértékelés

Mint láttuk, a top-down kiértékelés esetén elég nehéz a megállási feltétel vizsgálata, illetve időnként sok fölösleges kiértékelési lépést kell végrehajtunk. Fölszemes lépésekre a bottom-up kiértékelés esetén is sor kerülhet. Ezért célszerűnek látszik a két módszer ötvözése. A konkrét cél figyelembevételével átírhatjuk az eredeti programot, beleszöve a célállításban szereplő információkat, majd az átírt programot bottom-up módon értékeljük ki. Ily módon elérhetjük azt, hogy a bottom-up kiértékelés „célirányos” legyen, azaz ne számoljon ki fölöslegesen sok, a célhoz nem tartozó értéket, és ugyanakkor megszabadulhatunk a top-down kiértékelés kellemetlenségeitől is.

Közönséges Datalog esetén több ilyen eljárás is ismert, melyeket összefoglalóan átírási módszereknek neveznek. Ezek közül legismertebb az úgynevezett mágikus halmazok ([CGT], [U]), illetve a statikus filterek (Static Filtering) módszere ([CGT]).

A mágikus halmazok módszere adott illesztési mintájú célpredikátummal rendelkező programok kiértékelésére vonatkozik, és az oldalsó információátadás elvén alapszik. A program átírása során újabb predikátumokat és újabb szabályokat vezetünk be. Ezek a szabályok „megszorítják” a programot olyan értelemben, hogy a program változóinak ezen új predikátumoknak is eleget kell tenniük. Így a bottom-up kiértékelés során a változók csak bizonyos értékeket vehetnek fel, és nem az összes lehetséges szóba jöhető értéket. Ilyen módon szűkítjük, és evvel hatékonyabbá tehetjük a kiértékelést.

Az új predikátumok és új szabályok bevezetése nehézkessé teszi a módszer fDATALOG-ra való általánosítását, hiszen meg kellene vizsgálnunk, hogy milyen bizonytalansági szintek és milyen implikációs operátorok tartoznak az új predikátumokhoz, illetve szabályokhoz.

Ígéretesebbnek látszik a statikus filterek módszerének átvétele, hiszen itt – kivéve a cél szabállyá való átalakítását – nem szerepelnek új szabályok, illetve predikátumok. A módszer lényege, hogy bizonyos „filterek” – szűrő feltételek – alkalmazásával amilyen hamar csak lehet – vagyis a céltől minél messzebb, – levágja a fölösleges ágakat. Ezt a célból indulva, a filterek szabályokon való „átnyomásával” oldja meg. Ez azt jelenti, hogy a filterek módosulnak a

szabályok feltételrendszerétől függően. Ezt a módosítást addig végzi, ameddig csak lehet. A módosítások során az eredeti program egy átírt változatát kapjuk, melyet bottom-up módon megoldva az eredeti program célnak megfelelő megoldásával ekvivalens megoldást kapunk. Ezt a módszert sem részletezem, részletesen – bár nem elég precízen – tárgyalja [CGT].

5. A fDATALOG és a fuzzy relációk közötti kapcsolat

Egy közönséges Datalog programhoz hozzárendelhető egy relációs adatbázis, vagyis a program relációkkal interpretálható oly módon, hogy minden k argumentumú p predikátumhoz hozzárendelünk egy k argumentumú R relációt úgy, hogy R -be azok a sorok kerüljenek, amelyekre a p predikátum igaz, a program szabályaihoz pedig egy-egy relációs algebrai egyenlet rendelhető. Mint [CGT], [U] bizonyítja, ezekből az 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 fuzzy relációs algebrai kifejezést. A hozzárendelés azonban nem olyan egyszerű, mint a közönséges Datalog esetén, az irodalomban ugyanis többféle módon értelmezik a fuzzy relációkat [K1], [LL], [RM], [Um]. A következőkben két reláció-típussal foglalkozunk. Megmutatjuk, hogy az első típusú relációk és a rajtuk értelmezett relációs algebrai kifejezések hozzárendelhetők az fDATALOG programokhoz, míg a második típusú relációk az fDATALOG kiterjesztéséhez vezetnek.

5.1. Fuzzy relációs algebra

A Codd által bevezetett hagyományos relációs adatbázis egy vagy több táblából áll. Ezek a relációk. A reláció oszlopait nevekkkel, az úgynevezett attribútumokkal különböztetjük meg egymástól. Sokszor lényegtelen az attribútumnév, elég csak az oszlopok sorrendjére tekintettel lenni. Minden egyes attribútumhoz hozzátartozik egy értelmezési tartomány, az adott attribútumhoz tartozó oszlopba ebből a halmazból kerülhetnek értékek. Egy sor vagy beletartozik egy adott relációba, vagy nem. Fuzzy esetben azonban minden sorhoz hozzárendelhetünk egy $0, 1$ közötti értéket, amely a sor relációba való tartozásának mértékét mutatja.

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

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

Jelölése:

$$(R(A_1, \dots, A_n), \mu_R),$$

ahol A_1, \dots, A_n a reláció attribútumai.

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

5.1. Példa: A $(B(X,Y), \mu_B)$ első típusú fuzzy reláció, (pl. barátság):

B:	X	Y	μ_B
	Jani	Pali	0.1
	Márta	Éva	0.6

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

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

Mielőtt részletesebben tárgyalnánk ezt a hozzárendelést, definiálnunk kell néhány relációs algebrai fogalmat.

Mivel az első típusú fuzzy reláció a közönséges reláció soraihoz rendel egy-egy beletartozási szintet, ezért a relációs algebrai műveletek definiálásakor a bizonytalansági szintek kiszámításában térünk el a hagyományos relációs algebrai műveletektől.

A következő műveleteket értelmezzük:

1. Unió:

Az (R, μ_R) és (S, μ_S) reláció uniója azon sorok halmaza, melyek vagy R-ben vagy S-ben szerepelnek, s a közös sorok beletartozási szintje a beletartozási szintek maximuma:

$$(Q, \mu_Q) = (R, \mu_R) \cup (S, \mu_S) = \{(\underline{t}, \mu_Q(\underline{t})) \mid \underline{t} \in R \cup S, \mu_Q(\underline{t}) = \max(\mu_R(\underline{t}), \mu_S(\underline{t}))\}$$

2. Különbség:

Az (R, μ_R) és (S, μ_S) reláció különbsége azon sorok halmaza, melyek R-ben szerepelnek, de S-ben nem, s a közös sorok beletartozási szintje a beletartozási szintek különbsége:

$$(Q, \mu_Q) = (R, \mu_R) - (S, \mu_S) = \\ \{(t, \mu_Q(t)) \mid t \in R, \mu_Q(t) = \max(0, \mu_R(t) - \mu_S(t))\}$$

Az előző két művelet megkövetelte, hogy a relációk azonos argumentum-számúak legyenek. A következő művelet esetén nincs szükségünk erre a feltételre.

3. Descartes-szorzat:

A k argumentumú (R, μ_R) és az m argumentumú (S, μ_S) reláció Descartes szorzata azon $k + m$ elemű sorok halmaza, melyek első k komponense R-ben, utolsó m komponense S-ben szerepel, s a sorok beletartozási szintje a beletartozási szintek minimuma:

$$(Q, \mu_Q) = (R, \mu_R) \times (S, \mu_S) = \\ \{(t, \mu_Q(t)) \mid t = \underline{u}\underline{v}, \underline{u} \in R, \underline{v} \in S, \mu_Q(t) = \min(\mu_R(\underline{u}), \mu_S(\underline{v}))\}$$

4. Projekció:

Legyen U a k argumentumú (R, μ_R) reláció attribútumainak halmaza, és $X \subseteq U$! Az (R, μ_R) X halmazra vonatkozó projekciója az R sorainak X halmazra való megszorítása, s a közös sorok beletartozási szintje a beletartozási szintek maximuma:

$$(Q, \mu_Q) = \Pi_X((R, \mu_R)) = \\ \{(t, \mu_Q(t)) \mid t = \underline{u}[X], \underline{u} \in R, \mu_Q(t) = \max(\mu_R(\underline{u}) \mid t = \underline{u}[X])\},$$

ahol $t = \underline{u}[X]$ az \underline{u} X halmazra való megszorítását jelöli.

5. Szelekció:

Tegyük fel, hogy adott egy $F: D_1 \times \dots \times D_n \rightarrow [0,1]$ formula. Ekkor az (R, μ_R) reláció F formula szerinti szelekciója R azon sorainak halmaza, amelyek F szinten „teljesülnek”, azaz amelyek beletartozási szintje az eredeti szint és a formula megfelelő értékének minimuma:

$$(Q, \mu_Q) = \sigma_F(R, \mu_R) = \{(\underline{t}, \mu_Q(\underline{t})) \mid \underline{t} \in R, \mu_Q(\underline{t}) = \min(\mu_R(\underline{t}), F(\underline{t}))\}$$

6. Összekapcsolás (feltételes join):

Legyen A az (R, μ_R) , B az (S, μ_S) egy-egy attribútuma, θ pedig egy rajtuk értelmezett fuzzy predikátum, melynek igazságértékét a $\mu_\theta: D_A \times D_B \rightarrow [0,1]$ függvény jellemzi. Ekkor az (R, μ_R) és (S, μ_S) relációk A, B attribútum és θ predikátum szerinti összekapcsolása a relációk Descartes-szorzatának azon sorai, melyek megfelelő attribútumaira az adott predikátum μ_θ szinten „teljesül”, vagyis amelyek beletartozási szintje az eredeti beletartozási szintek és μ_θ megfelelő értékének minimuma:

$$(Q, \mu_Q) = (R, \mu_R) \underset{\underline{r}(A) \theta \underline{s}(B)}{\circ} (S, \mu_S) =$$

$$\{(\underline{t}, \mu_Q(\underline{t})) \mid \underline{t} = \underline{rs}, \underline{r} \in R, \underline{s} \in S, \mu_Q(\underline{t}) = \min(\mu_R(\underline{r}), \mu_S(\underline{s}), \mu_\theta(\underline{r}(A), \underline{s}(B)))\},$$

ahol $\underline{r}(A)$, $\underline{s}(B)$ az R illetve S reláció \underline{r} , \underline{s} sorának A illetve B attribútumra vonatkozó értékét jelöli.

A join kifejezhető relációs algebrai műveletek segítségével is:

$$(R, \mu_R) \underset{\underline{r}(A) \theta \underline{s}(B)}{\circ} (S, \mu_S) = \sigma_F((R, \mu_R) \times (S, \mu_S)), \text{ ahol } F = \underline{r}(A) \theta \underline{s}(B).$$

7. Természetes összekapcsolás (join):

Az (R, μ_R) és (S, μ_S) relációk $(R, \mu_R) \circ (S, \mu_S)$ összekapcsolása abban az esetben értelmezhető, ha a relációknak vannak közös attribútumai. Ekkor a két reláció join-ját a következő módon értelmezzük:

1. Kiszámítjuk a két reláció Descartes-szorzatát beletartozási szintek nélkül.
2. Minden egyes olyan A attribútumra, amely R-nek is és S-nek is attribútuma, kiválasztjuk $R \times S$ azon sorait, melyekre $\underline{r}(A) = \underline{s}(A)$.
3. Végül az A attribútumhoz tartozó két megegyező oszlopot egyesítjük.

Az így kapott sorok beletartozási szintje az eredeti beletartozási értékek minimuma.

A közönséges relációs algebrát ki szokták egészíteni bizonyos képességekkel, (pl. átlagszámítás, számlálás, stb.). Ezeket nevezik aggregációnak vagy

aggregációs függvényeknek. A fuzzy relációs algebra is kibővíthető véges sok $W: [0,1] \rightarrow [0,1]$ függvénnyel. Ily módon tudjuk értelmezni a nyolcadik műveletet:

8. Aggregáció:

Az (R, μ_R) relációra alkalmazva a W aggregációs függvényt, a $W(R, \mu_R)$ sorainak beletartozási értéke az lesz, amit a W rendel az eredeti beletartozási értékhez:

$$\mu_{W(R)} = W(\mu_R)$$

Ilyen aggregáció például a negáció, λ -levágás, de ilyen módon tudjuk alkalmazni a fuzzy módosítókat is.

Ezek után vizsgáljuk meg, mi módon rendelhető egy fDATALOG program predikátumaihoz egy-egy első típusú fuzzy reláció, illetve a program miképpen transzformálható át relációs algebrai egyenletrendszerre!

5.2. A fDATALOG relációs algebrai kiértékelése

A ténypredikátumokhoz konstans relációt rendelhetünk, a többihez pedig változó relációt, melyek egy relációs algebrai egyenletrendszer megoldása során kapnak értéket.

A ténypredikátumhoz tartozó relációt a következőképpen határozhatjuk meg: Legyen $p(x_1, \dots, x_n)$ ténypredikátum – azaz tegyük fel, hogy a programban létezik legalább egy olyan tény, melynek predikátuma p . A p predikátumhoz hozzárendeljük azt a $(P(A_1, \dots, A_n), \mu_P)$ relációt, melyre teljesül a következő feltétel: minden $(p(a_1, \dots, a_n); \alpha)$ esetén az $(a_1, \dots, a_n, \alpha)$ sor eleme a $(P(A_1, \dots, A_n), \mu_P)$ relációnak.

A továbbiakban minden egyes – ténytől különböző – szabály törzséhez megkonstruálunk egy, a törzsben szereplő atomokat felhasználó relációs algebrai kifejezést. Legyen a szóban forgó szabály

$$p(a_1, \dots, a_n) \leftarrow q_1(b_1, \dots, b_k), \dots, q_m(b_r, \dots, b_s); \beta; I.$$

alakú!

Tegyük föl, hogy a q_1, \dots, q_m predikátumokhoz rendre a (Q_i, μ_{Q_i}) $1 \leq i \leq m$ relációk tartoznak! A szabálytörzshöz tartozó kifejezés meghatározásához ezeket a relációkat kell összekapcsolnunk. Mivel azonban a törzspredikátumok argumenumai között előfordulhat ismétlődés, vagy szerepelhetnek konstans értékek, ezért a relációkat az összekapcsolás előtt szelekciónak kell alávetni.

A szelekció feltételrendszerét a konstans értékek és az azonos nevű attribútumok kezelése adja. Pontosabban:

A (Q_i, μ_{Q_i}) relációhoz tartozó F_i feltételrendszer legyen a $\$j = C$, ha $a_j = C$ és a $\$j = \k , ha $a_j = a_k$ típusú formulák konjunkciója! (Az egyenlőség speciális fuzzy predikátumnak tekinthető, melynek bizonytalansági szintje 0 vagy 1.)

A szelekciót elvégezve bizonyos oszlopok fölöslegessé válnak, emiatt elegendő, ha a relációnak a benne szereplő változók halmazára vonatkozó projekcióját tekintjük.

Végül az így kapott relációkat a természetes join-nal összekapcsoljuk. Ilyen módon megkapjuk a szabálytörzshöz tartozó relációs algebrai kifejezést.

A későbbi precíz fogalmazás kedvéért szükségünk van az értékelés fogalmára.

5.2. Definíció: Legyen X változók, C konstansok halmaza! Értékelés egy $v: X \cup C \rightarrow C$ leképezés, ahol v azonosság C -n.

Egy $\underline{t} = (a_1, \dots, a_n, \mu_R(\underline{t}))$ sor esetén

$$v(\underline{t}) = (v(a_1), \dots, v(a_n), \mu_R(v(a_1), \dots, v(a_n)))$$

5.1. Állítás: Legyen V_i a (Q_i, μ_{Q_i}) , $i = 1 \leq i \leq m$ változónak halmaza, és

$$(\mathbf{R}, \mu_{\mathbf{R}}) = \prod_{i=1}^m \Pi_{V_i}(\sigma_{F_i}(Q_i, \mu_{Q_i}))$$

az $(r = p(a_1, \dots, a_n) \leftarrow q_1(b_1, \dots, b_k), \dots, q_m(b_r, \dots, b_s); \beta; I)$ szabály törzséhez tartozó relációs algebrai kifejezés, $\underline{t} = (X_1, \dots, X_n)$ a törzs változói és v egy értékelés! Ekkor

$$\alpha_{\text{törzs}}(v(\underline{t})) = \mu_{\mathbf{R}}(v(\underline{t})),$$

ahol $\alpha_{\text{törzs}}(v(\underline{t}))$ az $(r; \beta; I)$ szabály v értékeléshez tartozó alap-előfordulásakor a szabálytörzs bizonytalansági foka.

Bizonyítás:

Mivel a bizonytalansági szint számításának lépései mindkét esetben egymáshoz rendelhetőek, ezért igaz az állítás. □

Megjegyzés: A továbbiakban az állításban szereplő (R, μ_R) relációra $(TÖRZS(r), \mu_{TÖRZS})$ néven hivatkozunk.

5.2. Példa: A

$$p(x,y) \leftarrow q_1(a, y), q_2(x, z, x); \beta; I.$$

szabály esetén, ahol „a” konstans, a törzshöz tartozó reláció:

$$(TÖRZS(r), \mu_{TÖRZS}) = \Pi_Y(\sigma_{\$1=a}(Q_1, \mu_{Q_1})) \cap \Pi_{X,Z}(\sigma_{\$1=\$3}(Q_2, \mu_{Q_2})).$$

A $(TÖRZS(r), \mu_{TÖRZS})$ reláció meghatározására megadunk egy algoritmust. Az algoritmusban megvizsgáljuk a szabálytörzs argumentumait, és elvégezzük a szükséges szelekciókat. Két speciális esetet kell szemmel tartanunk:

- i/ Az argumentumok között szerepel konstans.
- ii/ Ugyanaz a változó többször is szerepel az argumentumok között.

Az $(r; \beta; I) = (p(a_1, \dots, a_n) \leftarrow q_1(b_{1,1}, \dots, b_{1,k_1}), \dots, q_m(b_{m,1}, \dots, b_{m,k_m}); \beta; I)$ szabály esetén a következő algoritmus határozza meg a szabálytörzshöz tartozó relációs algebrai kifejezést:

5.1. Algoritmus:

Procedure törzsreláció $((TÖRZS, \mu_{TÖRZS}))$

$(TÖRZS, \mu_{TÖRZS}) := \emptyset$

for $i := 1$ to m do

$V_i := \{b_{i,1}, \dots, b_{i,k_i}\}$

$F_i := \text{true}$

for $j := 1$ to k_i do

if konstans $(b_{i,j})$ then

$F_i := F_i \wedge \$j = b_{i,j}$

$V_i := V_i - \{b_{i,j}\}$

endif

if $\exists s < j : b_{i,s} = b_{i,j}$ then

$F_i := F_i \wedge \$s = \j

```

        Vi := Vi - {bi,j}
    endif
endfor
(TÖRZS, μTÖRZS) := (TÖRZS, μTÖRZS) ∞ ΠVi(σFi(Qi, μQi))
endfor
endprocedure

```

A szabálytörzshöz tartozó reláció ismeretében meghatározhatjuk a szabályfej predikátumához tartozó relációt. Előfordulhat azonban, hogy egy programon belül több olyan szabály is van, amelynek ugyanaz a fejpredikátuma. Ezért a predikátumhoz tartozó reláció meghatározásakor célunk az, hogy a közös fejpredikátumú szabályok esetén képezzük a szabálytörzsekhez tartozó relációknak a fejben szereplő változók halmazára vonatkozó projekcióját, majd tekintsük az így kapott relációk unióját. Ehhez azonban „össze kell hangolnunk” a szabályokat. A szabályfejekben ugyanis különböző néven szerepelhetnek a predikátum azonos sorszámú argumentumai, lehetnek közöttük konstansok, vagy ismétlődések.

Amennyiben a fent említett problémák egyike sem fordul elő a közös fejpredikátumú szabályok között, akkor azt mondjuk, hogy ezek a szabályok rektifikáltak. Pontosabban:

5.3. Definíció: A p fejpredikátumú szabályok rektifikáltak, ha mindegyiküknek azonos a feje, és mindegyikük $p(x_1, \dots, x_n)$ alakú, ahol x_1, \dots, x_n különböző változók.

A rektifikációt egy új predikátum és a neki megfelelő reláció, az egyenlőség predikátum, illetve reláció bevezetésével oldjuk meg, melyet megfelelő argumentumokkal a szabálytörzshöz illesztünk. Jelöljük ezt a predikátumot $eq(x,y)$ -nal. A rá vonatkozó szabály: $eq(x,y) \leftarrow; 1; I$.

Az egyenlőség relációt $(EQ(X,Y), \mu_{EQ})$ - val jelöljük, és a következőképpen értelmezzük:

$$\mu_{EQ}(x, y) = \begin{cases} 1 & \text{ha } x = y \\ 0 & \text{egyébként} \end{cases}$$

Az egyenlőségtől megköveteljük a szimmetria teljesülését is.

Ezek után a rektifikáció alapötlete a következő: Az argumentumok helyére új változókat vezetünk be, s ez a változó és a régi argumentum egy egyenlőség predikátum argumentumaiként kerül a szabálytörzsbe.

5.3. Példa: A

$$q(x,a,z) \leftarrow p(x,y), r(y,z,a); \beta_1; I.$$

$$q(t,y,t) \leftarrow s(y,t); \beta_2; I.$$

szabályok rektifikálásakor mindkét szabályfejet $q(u,v,w)$ alakúra hozva a következőt kapjuk:

$$q(u,v,w) \leftarrow p(x,y), r(y,z,a), eq(u,x), eq(v,a), eq(w,z); \beta_1; I.$$

$$q(u,v,w) \leftarrow s(y,t), eq(u,t), eq(v,y), eq(w,t); \beta_2; I.$$

A rektifikálásra is megadunk egy egyszerű algoritmust.

Tekintsük az összes

$$(r; \beta; I) = (p(a_1, \dots, a_n) \leftarrow q_1(b_{1,1}, \dots, b_{1,k_1}), \dots, q_m(b_{m,1}, \dots, b_{m,k_m}); \beta; I)$$

alakú szabályt, és tegyük fel, hogy egyik szabály argumentumai között sem szerepelnek az u_1, \dots, u_n változók. Egy ilyen szabály a következő eljárás segítségével rektifikálható:

5.2. Algoritmus:

Procedure rektifikálás(r)

 for i :=1 to n do

 cserél(a_i, u_i)

 szabálytörzs(r) := szabálytörzs(r), eq(a_i, u_i)

 endfor

endprocedure

[U] bizonyítja, hogy közöséges Datalog szabályok esetén egy tetszőleges r szabály ekvivalens a rektifikáció útján kapott r' szabállyal abban az értelemben, hogy ugyanazok az alap-előfordulások tartoznak mindkét szabályhoz. Mivel a rektifikáció nem befolyásolja a bizonytalansági szintet, ezért az ekvivalencia fDATALOG programok esetén is fennáll.

Ezek után kiszámíthatjuk a szabályfejhez tartozó relációt. Ehhez felhasználjuk aggregációs függvényként a $W_{1,\beta}(\alpha) = f(I, \alpha, \beta)$ bizonytalansági-szint függvényt.

Tegyük fel, hogy a fDATALOG programban az x_1, \dots, x_n argumentumú p predikátum az $(r_1; \beta_1; I^1), \dots, (r_k; \beta_k; I^k)$ rektifikált szabályok fejpredikátuma! Ekkor a p -hez kiszámolt relációs algebrai kifejezés:

$$(\text{SZÁMOLT}(p), \mu_{\text{SZÁMOLT}}) = \bigcup_{i=1}^k W_{I^i, \beta_i} (\Pi_{X_1, \dots, X_n}(\text{TÖRZS}(r_i), \mu_{\text{TÖRZS}}))$$

$W_{I, \beta}(\alpha)$ definíciójából adódóan az így meghatározott reláció része a p -hez tartozó (P, μ_P) relációnak, vagyis igaz a következő állítás:

5.2. Állítás: Legyen a p predikátumhoz tartozó reláció (P, μ_P) . Ekkor

$$(\text{SZÁMOLT}(p), \mu_{\text{SZÁMOLT}}) \subseteq (P, \mu_P).$$

Mivel legkisebb (vagy minimális) modell megtalálására törekszünk, ezért a p predikátumhoz tartozó reláció meghatározására tekintsük a

$$(P, \mu_P) = (\text{SZÁMOLT}(p), \mu_{\text{SZÁMOLT}})$$

egyenletet.

Ugyanígyen egyenlet írható fel a programban szereplő többi predikátumra is.

Az előző fejezetben ismertetett fixpontkeresési eljárásnak megfeleltethető egy relációs algebrai fixpontkereső eljárás. Ez egy relációs algebrai egyenletrendszer megoldását jelenti, ahol az egyenletrendszer egyenletei

$$(P, \mu_P) = \bigcup_{i=1}^k W_{I^i, \beta_i} (\Pi_{X_1, \dots, X_n} (\bigcap_{i=1}^m \Pi_{V_i}(\sigma_{F_i}(Q_i, \mu_{Q_i}))))$$

alakúak.

5.3. Állítás: Negációmentes 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.

Bizonyítás:

Mivel bármely $(r; \beta; I)$ szabály tetszőleges v értékeléshez tartozó alap-előfordulása esetén $\alpha_{\text{TÖRZS}}(v(t)) = \mu_{\text{TÖRZS}}(v(t))$, továbbá $W_{I, \beta}(\alpha)$, NT_P (illetve DT_P) definíciója alapján $\alpha_{\text{fej}}(v(t)) = W_{I, \beta}(\mu_{\text{TÖRZS}}(v(t)))$, ezért igaz az állítás. \square

Valamely ismert egyenletrendszer-megoldó algoritmussal próbálkozva esetenként fölöslegesen sokat kell számolni. Emiatt javasolja közönséges Datalog

esetén [CGT], [U] a szemi-naiv kiértékelést. Ezt a kiértékelést általánosította fuzzy esetre [K1]. Ő egy EVAL kifejezést értelméz, amely megfelel az itt definiált (SZÁMOLT(p), $\mu_{\text{SZÁMOLT}}$) relációnak. Pontosabban:

Legyen D a program predikátumaihoz tartozó $(P_1, \mu_{P_1}), \dots, (P_n, \mu_{P_n})$ relációk halmaza! Ekkor

$$\text{EVAL}(p, D) = \bigcup_{i=1}^k W_{i, \beta_i} (\prod_{X_1, \dots, X_n} (\prod_{i=1}^m \Pi_{V_i}(\sigma_{F_i}(Q_i, \mu_{Q_i})))$$

[K1] a következő kiértékelési algoritmust adja meg:

Legyen p_1, \dots, p_n a program predikátumainak sorozata úgy, hogy $p_1, \dots, p_k, k < n$ csak tényekben szerepel.

Legyen $D_0 = ((P_1, \mu_{P_1}), \dots, (P_k, \mu_{P_k}), \emptyset, \dots, \emptyset)$, ahol (P_i, μ_{P_i}) $1 \leq i \leq k$ a p_1, \dots, p_k predikátumokhoz tartozó relációk.

Nem rekurzív esetben a kiértékelési algoritmus predikátumonként kiszámítja az általunk is definiált relációt. Rekurzív esetben pedig az EVAL művelet egymás utáni alkalmazásával egy rekurzív sorozatot értelméz, amely véges sok lépésben terminál. Ez a sorozat a következő:

$$D_i = ((P_1, \mu_{P_1}), \dots, (P_k, \mu_{P_k}), \text{EVAL}(p_{k+1}, D_{i-1}), \dots, \text{EVAL}(p_n, D_{i-1})).$$

[K1] megmutatja, hogy

i/ $D_0 \subseteq D_1 \subseteq \dots$

ii/ Az algoritmus terminál, azaz $\exists h: D_0 \subseteq \dots \subseteq D_h = D_{h+1} = D_{h+2}$

iii/ D_h a legkisebb modell.

Megjegyzés: A fentiekben ismertett relációs algebrai kiértékelés bottom-up kiértékelés volt. Hasonló módon meg lehetne adni a top-down kiértékeléssel ekvivalens relációs algebrai eljárást is.

5.3. A rétegzett fDATALOG relációs algebrai kiértékelése

Ahhoz, hogy negációt tartalmazó fDATALOG programokat is ki tudjunk értékelni relációs algebrai módszerekkel, értelmeznünk kell a negatív literálhoz tartozó relációt. Ezt egyszerűen megtehetjük az

$$N(\alpha) = 1 - \alpha$$

aggregációs függvény segítségével.

Rétegzett fDATALOG esetén negált predikátum kiértékelésére már csak akkor kerül sor, ha kiértékeljük a predikátum összes pozitív előfordulását. Ezért feltételezhetjük, hogy egy $\neg q(x_1, \dots, x_n)$ rész cél vizsgálatakor már ismert a q -hoz tartozó (Q, μ_Q) reláció. Így alkalmazhatjuk rá az $N(\mu_Q)$ aggregációt.

Ezt a módosítást figyelembe véve a relációs algebrai fixpontoszámítási eljárás ugyanúgy alkalmazható, mint negációmentes esetben.

Mint láttuk, nem determinisztikus szemantikával meg tudtuk határozni egy tetszőleges rétegzett program – rétegzési sorrendtől függő – minimális modelljét úgy, hogy a szabályokat a rétegzés sorrendjében értékeltük ki. Ugyanez mondható a relációs algebrai kiértékelésre is.

Mivel negált atom bizonytalansági szintje $(\alpha_{\neg A} = 1 - \alpha_A)$ az aggregációs függvénnyel megegyező módon számolható, valamint a logikai és a relációs algebrai fixpontoszámítás egyéb lépései továbbra is egymáshoz rendelhetőek, ezért igaz a következő:

5.4. Állítás: Adott rétegzés mellett a fixpontoszámítást a rétegzés sorrendjében végrehajtva a logikai és algebrai módszerrel kapott fixpont megegyezik, s ez a fixpont a program minimális modellje.

6. A fDATALOG fuzzy adatokra való kiterjesztése

Az első típusú fuzzy relációk azt mutatják meg, hogy mennyire szoros kapcsolat áll fenn az attribútum-értékek között. Ezen 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.

6.1. Definíció: Legyen D_1, \dots, D_n tetszőleges halmaz és $F(D_1), \dots, F(D_n)$, $1 \leq i \leq n$ a rajtuk értelmezett fuzzy részhalmazok!

A D_1, \dots, D_n halmazon értelmezett második típusú fuzzy reláció az $F(D_1) \times \dots \times F(D_n)$ egy fuzzy részhalmaza:

$$\mu_R: F(D_1) \times \dots \times F(D_n) \rightarrow [0,1]$$

Mivel a szövegkörnyezetből kiderül, hogy első vagy második típusú relációról van-e szó, ezért továbbra is megtartjuk az $(R(A_1, \dots, A_n), \mu_R)$ jelölést.

6.1. Példa: Az

R:	Név	Kor	Fizetés	μ_R
	János	31	$\{(70000, 0.8), (75000, 0.7)\}$	0.7
	Tamás	középkorú	82000	0.8
	Anna	fiatal	$\{(60000, 0.6), (70000, 0.8)\}$	0.9

egy második típusú fuzzy reláció.

6.1. Kiterjesztett fDATALOG

Célszerű lenne úgy kiterjeszteni 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.

6.2. Definíció: Egy D tartomány fölötti $\text{prox}_D: D \times D \rightarrow [0,1]$ leképezés szomszédsági predikátum, ha reflexív és szimmetrikus, azaz, ha

$$\begin{aligned} \text{prox}_D(x,x) &= 1 \text{ minden } x \in D \text{ esetén, és} \\ \text{prox}_D(x,y) &= \text{prox}_D(y,x) \text{ minden } x,y \in D \text{ esetén.} \end{aligned}$$

Szomszédsági mátrix a szomszédsági predikátum által meghatározott mátrix, vagyis egy n elemű D halmaz esetén olyan $n \times n$ -es mátrix, amelynek elemei a halmazelemek közötti szomszédsági értékek.

Ha a szomszédság tranzitív, azaz $\text{prox}_D(x, z) \geq \min(\text{prox}_D(x, y), \text{prox}_D(y, z))$ teljesül minden x, y, z esetén, akkor hasonlóságnak nevezzük.

Megjegyzés: Legyen a prox szomszédsági predikátum mátrixa Pr ! Prox akkor és csakis akkor tranzitív, ha

$$\text{Pr} \geq \text{Pr} \cdot \text{Pr} ,$$

ahol a mátrixszorzásban az elemek szorzása a minimum-, a szorzatok összegzése a maximumképzést jelenti.

6.2. Példa: Egyszerű számolással ellenőrizhető, hogy az alábbi Pr_1 mátrix szomszédsági, Pr_2 hasonlósági relációt ír le.

$\text{Pr}_1:$	a	b	c	d	e	$\text{Pr}_2:$	a	b	c	d	e
a	1	0	0.1	0.2	0.8	a	1	0.7	0.8	0.7	0.8
b	0	1	0.9	0.1	0	b	0.7	1	0.7	0.9	0.7
c	0.1	0.9	1	0.2	0	c	0.8	0.7	1	0.7	0.8
d	0.2	0.1	0.2	1	0.1	d	0.7	0.9	0.7	1	0.7
e	0.8	0	0	0.1	1	e	0.8	0.7	0.8	0.7	1

Mivel egy program változói különböző értelmezési tartományokból vehetnek fel értékeket, ezért a szomszédsági predikátumokat is és a hozzájuk tartozó

szomszédsági mátrixokat is értelmezési tartományonként definiálhatjuk. Ezen szomszédsági mátrixok segítségével tudjuk kiterjeszteni a fDATALOG-ot:

6.3. Definíció: Kiterjesztett fDATALOG programnak nevezzük a szomszédsági mátrixokkal kibővített, fuzzy adatokon értelmezett fDATALOG programot.

Ahhoz, hogy ki tudjunk értékelni egy kiterjesztett fDATALOG programot, át kell alakítanunk azt közönséges fDATALOG programmá. Ezt a szomszédsági predikátumok bevezetésével oldhatjuk meg, vagyis úgy, hogy a szabályokba beépítjük a szomszédság-kezelését. Most a tényeket is az eredeti definíciónak megfelelően – üres törzsű – szabály formájában tároljuk. Az átalakítást a következő lépésekben tehetjük meg:

1. Nevezzük át a program szabályaiban lévő változókat! Az új változónevek legyenek kettős indexűek: a változókat az első index alapján különböztetjük meg. Kiinduláskor mindegyik változó 0-t kap második indexként.

2. A szabályokban szereplő konstansok helyére írjunk indexelt változót, és egészítsük ki a szabály törzsét egy, a változó és a konstans szomszédságára utaló predikátummal. Precízebben:

Legyen „c” a szóban forgó konstans, és kerüljön helyére az az $x_{k,0}$ változó, amely nem szerepel az eddigi változónevek között. Ekkor a törzs a $\text{prox}(c, x_{k,0})$ predikátummal bővül.

3. Szabályonként folytassuk a változóátnevezést. Ha balról kezdve az átnevezési eljárást, a szabályban másodszorra is előfordul az $x_{i,0}$ változó, akkor azt nevezzük át $x_{i,1}$ -nek, majd folytassuk az eljárást. Ha ismét előfordul az $x_{i,0}$ változó, akkor nevezzük át $x_{i,2}$ -nek, stb., majd bővítsük a szabálytörzset az összes megfelelő szomszédsági állítással! Azaz: ha az átnevezés végén az $x_{i,0}$, $x_{i,1}, \dots, x_{i,k}$ változók szerepelnek a szabályban, akkor bővítsük a törzset a

$$\text{prox}_{Dx_i}(x_{i,0}, x_{i,1}), \text{prox}_{Dx_i}(x_{i,0}, x_{i,2}), \dots, \text{prox}_{Dx_i}(x_{i,0}, x_{i,k}), \\ \text{prox}_{Dx_i}(x_{i,1}, x_{i,2}), \dots, \text{prox}_{Dx_i}(x_{i,1}, x_{i,k}), \dots, \text{prox}_{Dx_i}(x_{i,k-1}, x_{i,k})$$

predikátumsorozattal. A prox predikátumok szimmetriája és a szabálytörzs jelentése miatt a predikátumok sorrendje közömbös.

6.1. Állítás: Az összes szabály megfelelő átírása után közönséges fDATALOG programot kapunk.

Ezek után a program a determinisztikus vagy a nemdeterminisztikus szemantika alapján kiértékelhető.

6.2. Állítás: Ha $\text{prox}_{D_{X_i}}$ tranzitív, akkor a

$$\text{prox}_{D_{X_i}}(x_{i,0}, x_{i,1}), \text{prox}_{D_{X_i}}(x_{i,0}, x_{i,2}), \dots, \text{prox}_{D_{X_i}}(x_{i,0}, x_{i,k}), \\ \text{prox}_{D_{X_i}}(x_{i,1}, x_{i,2}), \dots, \text{prox}_{D_{X_i}}(x_{i,1}, x_{i,k}), \dots, \text{prox}_{D_{X_i}}(x_{i,k-1}, x_{i,k})$$

kifejezés a

$$\text{prox}_{D_{X_i}}(x_{i,0}, x_{i,1}), \text{prox}_{D_{X_i}}(x_{i,0}, x_{i,2}), \dots, \text{prox}_{D_{X_i}}(x_{i,0}, x_{i,k})$$

alakra egyszerűsíthető.

Bizonyítás:

Belátjuk, hogy $\text{prox}_{D_{X_i}}(x_{i,1}, x_{i,2})$ elhagyható. Hasonlóan bizonyítható a többi tényező kihagyhatósága is.

A tranzitivitás miatt

$$\text{prox}_{D_{X_i}}(x_{i,1}, x_{i,2}) \geq \min(\text{prox}_{D_{X_i}}(x_{i,0}, x_{i,1}), \text{prox}_{D_{X_i}}(x_{i,0}, x_{i,2})).$$

Mivel az $A \leftarrow A_1, \dots, A_n$ szabály esetén $\alpha_{\text{törzs}} = \min(\alpha_{A_1}, \dots, \alpha_{A_n})$, ezért $\text{prox}_{D_{X_i}}(x_{i,1}, x_{i,2})$ kihagyható. \square

6.3. Példa: A

$$p(x) \leftarrow (a), q(x); \beta; I.$$

$$s(x, x) \leftarrow n(x); \beta; .$$

$$q(x, y) \leftarrow p(x, z), q(z, y); \beta; .$$

szabályok átalakított változata:

$$p(x_{1,0}) \leftarrow r(x_{2,0}), q(x_{1,1}), \text{prox}(x_{2,0}, a), \\ \text{prox}(x_{1,0}, x_{1,1}); \beta; I.$$

$$s(x_{1,0}, x_{1,1}) \leftarrow n(x_{1,2}), \text{prox}(x_{1,0}, x_{1,1}), \\ \text{prox}(x_{1,0}, x_{1,2}), \text{prox}(x_{1,1}, x_{1,2}); \beta; I.$$

$$q(x_{1,0}, x_{2,0}) \leftarrow p(x_{1,1}, x_{3,0}), q(x_{3,1}, x_{2,1}), \\ \text{prox}(x_{1,0}, x_{1,1}), \text{prox}(x_{2,0}, x_{2,1}), \\ \text{prox}(x_{3,0}, x_{3,1}); \beta; I.$$

A kiterjesztett fDATALOG P program szabályainak átírására megadunk egy algoritmust.

6.1. Algoritmus:

Procedure átírás

$R := \{A \text{ P program szabályainak halmaza}\}$

while nem_üres (R) do

szabály := kiválasztás(R)

változólista := {a szabály változóinak listája}

konstanslista := {a szabály konstansainak halmaza}

törzs := szabálytörzs

i := 1

while nem_üres(változólista) do

v := elsőelem(változólista)

változólista := változólista - v

cserél (v, $x_{i,0}$)

*/*átírjuk az összes változót */*

i := i+1

endwhile

while nem_üres(konstanslista) do

c := elsőelem(változólista)

konstanslista := konstanslista - c

cserél (c, $x_{i,0}$)

/ átírjuk az összeskonstansot */*

törzs := törzs, $\text{prox}_{D_{x_i}}(c, x_{i,0})$

i := i+1

endwhile

for k := 1 to i-1 do

m := max(n, $x_{k,n}$)

/ meghatározzuk a k-adik változó legnagyobb
második indexét */*

if m > 0 then

for d := 1 to m do

törzs := törzs, $\text{prox}_{D_{x_k}}(x_{k,0}, x_{k,d})$

*/*bővítjük a törzset a megfelelő szomszédsági
kapcsolattal*/*

```

if (d>1 and nem_tranzitív (proxDxk)) then
    /* ha a reláció nem tranzitív, akkor bővítjük a törzset a
    többi szükséges szomszédsággal is */
    for j := 1 to d - 1 do
        törzs := törzs, proxDxk(xk,j, xk,d)
    endfor
endif
endfor
endif
endwhile
R := R - szabály
endprocedure

```

Az elmondottak megvilágítására nézzünk meg két példát! (A könnyebb olvashatóság kedvéért a kettős indexek helyett különböző változóneveket használunk.)

6.4. Példa:

```

(p(a,b), 0.9).
(p(c,d), 0.8).
q(x,y) ← p(x,y); 0.7; I1.
q(x,y) ← p(x,z), q(z,y); 0.8; I1.

```

prox	a	b	c	d	e
a	1	0	0.1	0.2	0.8
b	0	1	0.9	0.1	0
c	0.1	0.9	1	0.2	0
d	0.2	0.1	0.2	1	0.1
e	0.8	0	0	0.1	1

Az átírt szabályok (a szomszédsági mátrix nélkül):

```

p(x,y) ← prox(x,a), prox(y,b); 0.9; I1.
p(x,y) ← prox(x,c), prox(y,d); 0.8; I1.
q(x,y) ← p(x1,y1), prox(x1,x), prox(y1,y); 0.7; I1.
q(x,y) ← p(x1,z1), q(z2,y1), prox(x1,x), prox(y1,y),
    prox(z1,z2); 0.8; I1.

```

A fixpont :

p:	a	b	c	d	e	q:	a	b	c	d	e
a	0.1	0.9	0.9	0.1	0.1	a	0.2	0.7	0.7	0.7	0.2
b	0.2	0.1	0.2	0.8	0.1	b	0.2	0.2	0.2	0.7	0.2
c	0.2	0.1	0.2	0.8	0.1	c	0.2	0.2	0.2	0.7	0.2
d	0.2	0.2	0.2	0.2	0.1	c	0.2	0.2	0.2	0.2	0.2
e	0	0.8	0.8	0.1	0	e	0.2	0.7	0.7	0.7	0.2

és az eredeti szomszédsági mátrix.

6.5. Példa: Egy bűntény felderítéséhez fel akarjuk használni a szemtanúknak az elkövető termetére és korára vonatkozó megfigyeléseit. Ez alapján valaki „gyanúsnak” tekinthető, ha termete és kora „jól illeszkedik” a szemtanúk személyleírásához. Jelentse a p, t, k, g predikátum a következőket:

p(x,y,z): az x nevű személy termete y, kora z
 t(x): a szemtanúk x termetűnek látták az elkövetőt
 k(x): a szemtanúk x korúnak vélték az elkövetőt
 g(x): az x nevű személy gyanúsítható

Ekkor a gyanús személy meghatározására vonatkozó fDATALOG program:

```
(p(A, sovány, 35) ; 0.9) .
(p(B, átlagos, 40) ; 0.8) .
(t(sovány) ; 0.6) .
(k(38) ; 0.7) .
(k(40) ; 0.8) .
g(x) ← p(x, y, z) , t(y) , k(z) ; 0.8 ; I2 .
```

Mivel három különböző értelmezési tartományt adhatunk meg, ezért három különböző szomszédsági predikátumot definiálhatunk:

prox₁(x, y): Az x és y nevű személy hasonlóságát mutató szomszédsági reláció:

$$\text{prox}_1(x, y) = \text{eq}(x, y)$$

$\text{prox}_2(x, y)$: Az x és y termet szomszédsága:

prox_2	sovány	átlagos	kövér
sovány	1	0.7	0.3
átlagos	0.7	1	0.6
kövér	0.3	0.6	1

$\text{prox}_3(x, y)$: Az x és y kor szomszédsága:

$$\text{prox}_3(x, y) = 1 - |x-y| / \min(x,y)$$

A programban szereplő konstansokra $\text{prox}_3(x, y)$ mátrix alakban is megadható:

prox_3	35	38	40
35	1	0.91	0.86
38	0.91	1	0.95
40	0.86	0.95	1

Az átírt szabályok:

$(\text{prox}_1(A, A); 1)$.

...

$(\text{prox}_2(\text{sovány}, \text{sovány}); 1)$.

...

$(\text{prox}_3(35, 35); 1)$.

...

$p(x, y, z) \leftarrow \text{prox}_1(x, A), \text{prox}_2(y, \text{sovány}),$
 $\text{prox}_3(z, 35); 0.9; I_1.$

$p(x, y, z) \leftarrow \text{prox}_1(x, B), \text{prox}_2(y, \text{átlagos}),$
 $\text{prox}_3(z, 40); 0.8; I_1.$

$t(x) \leftarrow \text{prox}_2(x, \text{sovány}); 0.6; I_1.$

$k(x) \leftarrow \text{prox}_3(x, 38); 0.7; I_1.$

$k(x) \leftarrow \text{prox}_3(x, 40); 0.8; I_1.$

$g(x) \leftarrow p(x_1, y, z), t(y_1), k(z_1), \text{prox}_1(x, x_1),$
 $\text{prox}_2(y, y_1), \text{prox}_3(z, z_1); 0.8; I_2.$

Ily módon közöséges fDATALOG programot kaptunk, amely a tárgyalt kiértékelési stratégiák valamelyikével kiértékelhető. Bottom up kiértékelés esetén a kapott fixpont a szomszédsági mátrixokat is tartalmazza.

Megjegyzés: Abban az esetben, ha a szomszédsági predikátum az egyenlőségi predikátum, a megfelelő változók átírásával és az egyenlőségi predikátum kihagyásával a szabálymegadás egyszerűsödhet.

6.2. Top-down kiértékelés kiterjesztett fDATALOG programok esetén

A szomszédsági mátrix használata még kis méretű adatbázisok esetén is nagyon megnöveli a kiértékelés számolásigényét, hiszen ha a programhoz k db szomszédsági predikátum tartozik, és a lehetséges konstansok száma n_i , $1 \leq i \leq k$, akkor a szomszédsági predikátum bevezetésével a program $\sum n_i^2$ ténnyel bővül és a kiértékelendő szabályok is jóval bonyolultabbakká válnak.

Mivel a bottom-up kiértékelés figyelembe veszi az összes szabályt, és alulról építkezve határozza meg a program outputjának számító legkisebb fixpontot, ezért ez a kiértékelés sajnos nem egyszerűsíthető. A top-down módszernél azonban jelentős egyszerűsítés érhető el, ha ismerjük a cél-predikátum bizonytalansági szintjét, β -t. Ekkor ugyanis a szomszédsági mátrixból és a szabályok közül elhagyható az összes β -nál kisebb elem, illetve a β -nál kisebb bizonytalansági szintű szabály, így adott esetben lényegesen egyszerűsödhet a számítás.

7. Fuzzy tudásbázis

Egy fuzzy Datalog program önmagában is leír valamilyen tudást, de sokszor mégsem ad választ a kérdéseinkre. Még akkor sem, ha az előző fejezetben értelmezett kiterjesztéssel próbálkozunk. Tény, hogy ekkor valamivel pontatlanabb információk esetén is tudunk következtetéseket levonni, de gyakran előfordulhat, hogy még ennél több információra, valamilyen háttértudásra is szükségünk van a válasz megtalálásához. A következőkben a hatodik fejezetben tárgyalt elképzelést fejlesztjük tovább.

7.1. Háttértudás

Először a háttértudás egy lehetséges modelljét építjük fel, melyet a 6.2. definícióban értelmezett szomszédság fogalmára épülő szomszédsági halmazok segítségével definiálunk.

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

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

ahol $d_i \in D$ és $\text{prox}_D(d, d_i) = \lambda_i$ minden $i = 1, \dots, n$ esetén.

A szomszédság fogalmával definiálhatjuk, hogy tetszőleges alaptermek és tetszőleges predikátumszimbólumok mennyire vannak közel egymáshoz, mennyire tekinthetők egymás „szinonimáinak”, ily módon megalkothatjuk az úgynevezett háttértudást.

7.2. Definíció: 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:

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

7.2. A 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 háttér-tudás 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 szomszédsági értékéből számíthatjuk ki. Erre szolgálnak a dekódoló függvények, amelyekről elvárhatjuk, hogy azonosság esetén ne változtassák meg a kapott szintet, egyébként pedig legfőljebb akkora értéket adjanak, mint az eredeti bizonytalansági szint vagy a szomszédsági értékek. Azt is megköveteljük, hogy a dekódoló függvények minden változójukban monoton növekvők legyenek. Ezek alapján a dekódoló függvény fogalmát a következő módon definiálhatjuk:

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

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

valós függvényt értünk, melyre

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

$$\varphi(\alpha, 1, 1, \dots, 1) = \alpha \quad \text{és}$$

$$\varphi(\alpha, \lambda, \lambda_1, \dots, \lambda_n) \text{ minden argumentumában monoton növekvő.}$$

7.1. Példa:

$$\varphi_1(\alpha, \lambda, \lambda_1, \dots, \lambda_n) = \min(\alpha, \lambda, \lambda_1, \dots, \lambda_n);$$

$$\varphi_2(\alpha, \lambda, \lambda_1, \dots, \lambda_n) = \min(\alpha, \lambda, (\lambda_1 \cdot \dots \cdot \lambda_n));$$

$$\varphi_3(\alpha, \lambda, \lambda_1, \dots, \lambda_n) = \alpha \cdot \lambda \cdot \lambda_1 \cdot \dots \cdot \lambda_n$$

dekódoló függvények.

Érdemes megemlíteni, hogy bármely trianguláris norma eleget tesz a dekódoló függvény feltételeinek, pl. a fenti „min” és szorzás operátorok t-normák.

A program összes predikátumához – de legalábbis a fej-predikátumokhoz – dekódoló függvényt kell rendelnünk. Ezek halmaza alkotja a program dekódoló halmazát. A definícióhoz szükségünk lesz a *funktor* fogalmára, amely egy atom predikátumszimbólumának és argumentumszámának együttesét, azaz pl. $q(t_1, t_2, \dots, t_n)$ esetén q/n -t jelenti.

7.4. Definíció: Legyen P egy fuzzy Datalog program, F_P a program funktorainak halmaza! P dekódoló halmazán a

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

halmazt értjük.

Tisztáztuk a szükséges fogalmakat, értelmezzük tehát a fuzzy tudásbázist!

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

7.6. Definíció: Legyen (P, B_k, Φ_P, mA) egy fuzzy tudásbázis. A tudásbázis következményén a módosítási algoritmus szerint meghatározott mP program legkisebb fixpontját, $lfp(mP)$ -t értjük. Jelölése: $C(P, B_k, \Phi_P, mA)$.

7.3. Módosítási algoritmusok

Különböző módosítási algoritmusokat definiálhatunk, a továbbiakban ezeket foglaljuk össze.

7.3.1. Egyszerű módosítás (mA1 algoritmus)

Legyen P egy fuzzy Datalog program, és B_k 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 szomszédsági halmazokkal, SR_p -vel, illetve SC_t -vel, és minden x változót az $X=\{x\}$ halmazzal!

A módosítás algoritmus:

7.1. Algoritmus:

Procedure módosítás(P, mP)

$mP := \emptyset$

while nem(üres(P)) do

$(r;\beta;I) :=$ a P első szabálya

$(R;\beta;I) :=$ (helyettesített(r); β ;I)

$mP := mP \cup (R;\beta;I)$

$P := P - \{(r;\beta;I)\}$

endwhile

endprocedure

```

function helyettesített(r)
  Predr := az r predikátumainak halmaza
  Termr := az r alap-termjeinek halmaza
  Varr := az r változóinak halmaza

  while nem(üres(Predr)) do
    q := Predr első predikátumszimbóluma
    Q := SRq
    Predr := Predr - {q}
  endwhile

  while nem(üres(Termr)) do
    t := Termr első alap-termje
    T := SCt
    Termr := Termr - {t}
  endwhile

  while nem(üres(Varr)) do
    x := Varr első változója
    X := {x}
    Varr := Varr - {x}
  endwhile

  return helyettesített(r)
endfunction

```

Az így kapott mP programot ugyanúgy értékeljük ki, mint egy közöségi fDATALOG programot. A kapott fixpont azonban még a szomszédsági halmazokat tartalmazza, ezekből meg kell határoznunk a végeredményt jelentő alaptermeket és bizonytalansági fokukat. Ezek alkotják a módosított program fixpontját. Mivel $\text{lfp}(\text{NT}_{\text{mP}})$ legkisebb fixpont, ezért a tagjainak „kibontásával” kapott halmaz is legkisebb fixpont lesz.

7.7. Definíció: A módosított mP program legkisebb fixpontja:

$$\text{lfp}(\text{mP}) = \bigcup \{ (q(t_1, t_2, \dots, t_n); \varphi_q(\alpha_q, \lambda_q, \lambda_{t_1}, \dots, \lambda_{t_n})) \mid \forall (Q(T_1, T_2, \dots, T_n); \alpha) \in \text{lfp}(\text{NT}_{\text{mP}}), (q, \lambda_q) \in Q, (t_i, \lambda_{t_i}) \in T_i, 1 \leq i \leq n \}$$

Mivel $(q(t_1, t_2, \dots, t_n); \alpha) \in \text{Ifp}(\text{NT}_p)$ esetén
 $(Q(T_1, T_2, \dots, T_n); \alpha) = (\text{SR}_q(\text{SC}_{t_1}, \text{SC}_{t_2}, \dots, \text{SC}_{t_n}); \alpha) \in \text{Ifp}(\text{NT}_{mP})$, és
 $\varphi_q(\alpha, 1, 1, \dots, 1) = \alpha$, ezért igaz a következő állítás:

7.1. Állítás:

$$\text{Ifp}(\text{NT}_p) \subseteq C(P, B_k, \Phi_p, mA1).$$

7.2. Példa: Tekintsük a 3.3. példa fDATALOG programját, és egészítsük ki az alábbi háttértudással, dekódoló halmazzal!

$$\begin{aligned} & (r(a), 0.8). \\ p(x) & \leftarrow r(x), \neg q(x); 0.6; I_1. \\ q(x) & \leftarrow r(x); 0.5; I_1. \\ p(x) & \leftarrow q(x); 0.8; I_1. \end{aligned}$$

$$\begin{aligned} \varphi_p(\alpha, \lambda_p, \lambda) &= \varphi_q(\alpha, \lambda_p, \lambda) = \min(\alpha, \lambda_p, \lambda); \\ \varphi_r(\alpha, \lambda_r, \lambda) &= \alpha \cdot \lambda_r \cdot \lambda \end{aligned}$$

	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	0.8
t			0.7	0.8	1

Az 3.3. példa eredménye szerint

$$\text{Ifp}(\text{NT}_{mP}) = \{(R(A), 0.8); (P(A), 0.5); (Q(A), 0.5)\}$$

$$\begin{aligned} \text{Mivel } (R(A), 0.8) &= (\{(r, 1), (s, 0.6), (t, 0.7)\}(\{(a, 1), (b, 0.8)\}), 0.8); \\ (P(A), 0.5) &= (\{(p, 1), (q, 0.4)\}(\{(a, 1), (b, 0.8)\}), 0.5); \\ (Q(A), 0.5) &= (\{(q, 1), (p, 0.4)\}(\{(a, 1), (b, 0.8)\}), 0.5)\}, \text{ ezért} \end{aligned}$$

$$\begin{aligned} \text{Ifp}(mP) &= \\ & \{(r(a), 0.8), (r(b), 0.8 \cdot 0.8 = 0.64), (s(a), 0.6 \cdot 0.8 = 0.48), (s(b), 0.6 \cdot 0.8 \cdot 0.8 = 0.384), \\ & (t(a), 0.56), (t(b), 0.448), (p(a), 0.5), (p(b), 0.5), (q(a), 0.5), (q(b), 0.5)\}. \end{aligned}$$

7.3.2. 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 részalmazai fölött értelmeztük. A transzformáció módosításához a Herbrand bázis kiterjesztésére van szükségünk. Egészítsük ki a program Herbrand univerzumát a háttértudásban szereplő összes alap-term-mel! 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 $P \cup B_k$ halmaznak, argumentumai pedig a módosított Herbrand univerzum elemei. A transzformáció segítségével a szabályfejekben lévő atomokra és a hozzájuk hasonló atomokra is tudunk következtetni. Pontosabban:

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

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

ahol

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

($|A_i|$ - 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.

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

7.9. Definíció: A módosított mP program legkisebb fixpontja:

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

A fenti konstrukció alapján nyilvánvaló, hogy az mA2 algoritmus esetén is fennáll a 7.1. állításban megfogalmazotthoz hasonló tartalmazási reláció:

7.2. Állítás:

$$\text{lfp}(NT_P) \subseteq C(P, B_k, \Phi_P, mA2).$$

7.3. Példa: Tekintsük ismét a 7.2. 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 7.2. példában, vagyis 1., 3., 2., 4. szabály.

A kiindulási halmaz a szomszédság 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 szomszédságot, 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:

$$\text{lfp}(mP) = \{(r(a), 0.8), (r(b), 0.64), (s(a), 0.48), (s(b), 0.384), (t(a), 0.56), (t(b), 0.448), (q(a), 0.5), (q(b), 0.5), (p(a), 0.5), (p(b), 0.5)\}$$

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:

7.3. Állítás:

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

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

7.4. Példa: Tekintsük a következő tudásbázist:

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

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

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

	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

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

$C(P, Bk, \Phi_P, mA2) = \{(k(V),0.9); (k(B),0.81); (k(H),0.54); (jo(V),0.675);$
 $(jo(B),0.6075); (jo(H),0.405); (zk(M),0.8); (mu(M),0.6); (jo(BB),0.9);$
 $(jo(K),0.8); (k(BB),0.75); (k(K),0.75); (mu(P),0.8); (zk(P),0.6); (ko(B),1);$
 $(ko(K),1); (sz(M,V),0.6); (sz(M,B),0.6); (sz(M,H),0.405); (sz(M,BB),0.6);$

(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 muzsikuskok (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 muzsikusk 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.

7.4. Kiértékelési stratégiák

Egy fuzzy tudásbázis következményét fixpont-szemantikával értelmeztük, vagyis a tényekből kiindulva a szabályok és a szomszédság alkalmazásával adatvezérelt módon következtettünk az összes előállítható atomra. Azonban hasonlóan, mint ahogy a fuzzy Datalog tárgyalásánál láttuk, ebben az esetben is gondolkozhatunk célvezérelt módon is.

A stratégia alkalmazhatóságához ki kell egészítenünk a tudásbázist a meghatározandó céllal (kérdéssel), vagyis egy $(q(t_1, t_2, \dots, t_n), \alpha)$ párral, ahol $q(t_1, t_2, \dots, t_n)$ 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 célvezérelt kiértékelését tárgyaljuk.

7.4.1. Az mA1 algoritmussal összekapcsolt tudásbázis kiértékelése

A 4. fejezetben tárgyaltuk a fuzzy Datalog célvezérelt kiértékelését. 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 szomszédsági halmazok felhasználásával alakítsuk át az adott $(q(x_1, x_2, \dots, x_n); \alpha)$ célt a módosított $(Q(X_1, X_2, \dots, X_n); \alpha)$ céllá! Az így keletkező kérdésre a 4. fejezetben leírt módon kapunk választ. Innen a dekódoló függvényeken alapuló eljárás segítségével meghatározhatunk egy válasz-halmazt, amely tartalmazza az eredeti kérdésre nyert választ is.

7.2. Algoritmus:

Procedure dekódolás (Q, P, SR, SC, Φ_P , Válaszok)

VQ := a (Q; α) kérdésre adott válaszok halmaza

Válaszok := \emptyset

while nem(üres(VQ)) do

$(Q(T_1, T_2, \dots, T_n); \alpha)$:= VQ első eleme

 Válaszok := Válaszok \cup dekódolt($(Q(T_1, T_2, \dots, T_n); \alpha)$)

 /*az összes válasz dekódolása */

 VQ := VQ - $\{(Q(T_1, T_2, \dots, T_n); \alpha)\}$

endwhile

endprocedure

function dekódolt($(Q(T_1, T_2, \dots, T_n); \alpha)$)

 Dekódolt_halmaz := \emptyset

 while nem(üres(Q)) do

(q, λ_q) := Q első eleme

 q_set := \emptyset

 for all possible $((t_1, \lambda_{t_1}), (t_2, \lambda_{t_2}), \dots, (t_n, \lambda_{t_n})) \in T_1 \times T_2 \times \dots \times T_n$ do

 q_set := q_set \cup $\{(q(t_1, t_2, \dots, t_n); \phi_q(\alpha, \lambda_q, \lambda_{t_1}, \dots, \lambda_{t_n}))\}$

 endfor

 Dekódolt_halmaz := Dekódolt_halmaz \cup q_set

 /*a $(Q(T_1, T_2, \dots, T_n); \alpha)$ válasz összes „hasonmása” */

 Q := Q - $\{(q, \lambda_q)\}$

 endwhile

 return Dekódolt_halmaz

endfunction

Az előbbi algoritmus alapján könnyen belátható a következő állítás:

7.4. Állítás: Legyen Válaszok a 7.2. algoritmus alapján meghatározott válasz-halmaz. Ekkor

$$\text{Válaszok} \subseteq C(P, B_k, \Phi_P, mA1).$$

7.4.2. 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 célvezérelt 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 4. fejezetben tárgyalt helyettesítés és egységesítés fogalmára, illetve speciális helyettesítésekre is: időnként helyettesíteni kell egy p predikátumot vagy egy t term-et a szomszédsági halmazával, S_p -vel, illetve S_t -vel, és időnként helyettesíteni kell egy szomszédsági halmazt az elemeivel. Az egyszerűbb fogalmazás kedvéért ezen túl 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 szülő-csúcsai a keresett kiindulási tények. Ez a keresési fa a szomszédságon és a szabályokon alapuló egységesítés váltakozásával épül fel.

A szabályokon alapuló helyettesítés a részcejt a vele illeszthető szabályfejekkel egységesíti, és a kiértékelést a szabálytörzsszel folytatja. Az egységesítés során alkalmazott helyettesítés speciális abban az értelemben, hogy egy konstansot a

szomszédsági halmazával helyettesítünk, illetve az argumentumokban már szereplő szomszédsági halmazok úgy viselkednek, mint a közöséges egységesítés konstansai.

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

Az előzőek figyelembevételével a keresési fa a következőképpen épül fel: Ha a cél szintjét 0 mélységűnek tekintjük, akkor minden $3k+2$ ($k=0, 1, 2, \dots$) 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 $3k+1$ ($k=0, 1, 2, \dots$) 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 -l, 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 közöséges 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 tárgyalás mód rovására menne.

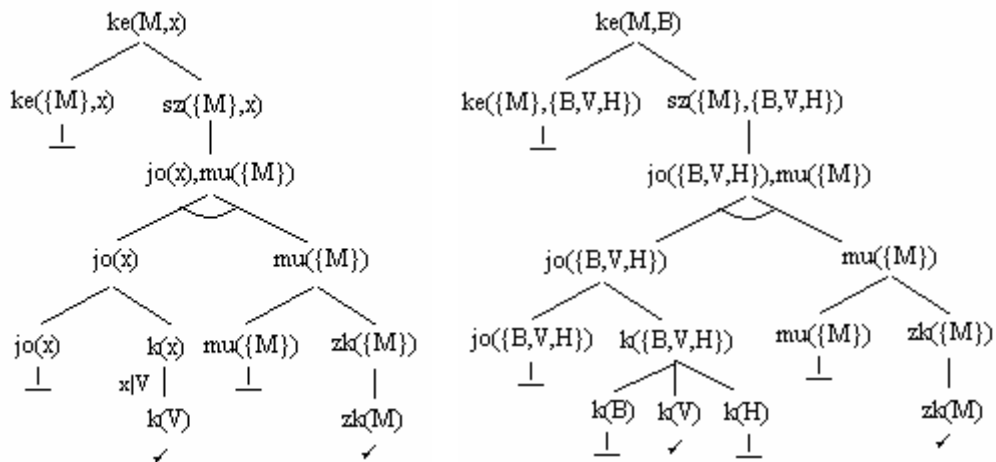
A második esetben a rákövetkezők az IGAZ/HAMIS szimbólumok valamelyike, attól függően, hogy az egységesített tény szerepel-e a program alapatomjai között vagy sem. A HAMIS címkéjű csúcspontnak nincs utóda.

A keresési gráf konstrukciója alapján nyilvánvaló:

7.5. 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.

7.5. Példa: Egészítsük ki a 7.4. 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 a 7.4. feladatban, de ez a halmaz tartalmazza a kérdésekre adható választ.

A fentiekben tárgyalt konstrukció a következő algoritmusban foglalható össze:

7.3. Algoritmus:

Procedure kiértékelés($g(t_1, t_2, \dots, t_n)$, Eredmények)

/ $g(t_1, t_2, \dots, t_n)$ a megoldandó cél */*

Fejek := {a programban szereplő szabályfejek}

Tények := {a programban szereplő tények}

Eredmények := \emptyset

/ az eredményül kapott kiindulási tények */*

for $i := 1$ to n do

 if változó(t_i) then $s_i := t_i$

 else $s_i := St_i$

/ St_i a t_i szomszédsági halmaza */*

 end_if

end_for

Csúcsok := { $g(s_1, s_2, \dots, s_n)$ }

/ Csúcsok a kiértékelendő részcélok halmaza, elemeire a szomszédsági-kiértékelést alkalmazzuk */*

Új_csúcsok := \emptyset

/ Csúcsok rákövetkezői, elemeire a szabály-kiértékelést alkalmazzuk */*

while nem_üres(Csúcsok) do

$r(\underline{t}) := \text{eleme}(\text{Csúcsok})$

 Csúcsok := Csúcsok – { $r(\underline{t})$ }

 Srcsúcsok := \emptyset

/ $r(\underline{t})$ rákövetkezői */*

 szomszédsági_kiértékelés($r(\underline{t})$, Srcsúcsok)

/ meghatározzuk az r predikátummal szomszédosakból képzett csúcsokat */*

 Új_csúcsok := Új_csúcsok \cup Srcsúcsok

while nem_üres(Új_csúcsok) do

$p(\underline{t}) := \text{eleme}(\text{Új_csúcsok})$

 Új_csúcsok := Új_csúcsok – { $p(\underline{t})$ }

 Spcsúcsok := \emptyset

/ $p(\underline{t})$ rákövetkezői */*

 szabály_kiértékelés($p(\underline{t})$, Spcsúcsok)

 Csúcsok := Csúcsok \cup Spcsúcsok

```

    end_while
  end_while
end_procedure

```

```

procedure szomszédási_kiértékelés(r(t1,t2,...,tn),Srcsúcsok)
  for all q ∈ Sp do
    /* Sr az r szomszédási halmaza */
    Srcsúcsok := Srcsúcsok ∪ {q(t1,t2,...,tn)}
  end_for
end_procedure

```

```

procedure szabály_kiértékelés(p(t1,t2,...,tn), Spcsúcsok)
  for all p(v1,v2,...,vn) ∈ Fejek do
    if egységesíthető(p(t1,t2,...,tn),p(v1,v2,...,vn)) then
      Spcsúcsok := Spcsúcsok ∪
        {a p(t1θ,t2θ,...,tnθ) fejű szabály törzsének illesztett
        predikátumai}
      /* θ a megfelelő egységesítés */
    end_if
  end_for

```

```

  for all p(v1,v2,...,vn) ∈ Tények do
    illeszhető := true
    for i := 1 to n do
      if változó(ti) then
        ti := vi /* a változót a konstanssal illesztjük */
      else if szomszédási_halmaz(ti) then
        if vi ∉ ti then illeszhető := false
        end if
      end_if
    end_for
    if illeszhető then
      Eredmények := Eredmények ∪ {p(v1,v2,...,vn)}
      /* az illeszhető tényt hozzávesszük az eredményhalmazhoz */
    end_if
  end_for
end_procedure

```

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

8. Kitekintés

Miután az előző fejezetekben felépítettünk egy lehetséges bizonytalanságkezelési modellt, szeretnék még egy kis kitekintést nyújtani. Témám nagy részével korábban kezdtem el foglalkozni. Különböző hatások – elsősorban szakmai jellegű olvasmányaim – indítottak el ebben az irányban. A fuzzy Datalog fogalmával először Kiss Attila cikkében találkoztam [K1]. A programnyelv elnevezése és szintaktikájának definiálása ebből a cikkből származik, szemantikájának megadása, kiértékelési stratégiáinak kidolgozása, illetve a különböző kiterjesztések értelmezése azonban saját munkám eredménye.

Amikor elkezdtem kutatni a témát, akkor a nemzetközi szakirodalomban sok cikket lehetett olvasni a Datalog-szerű nyelvekről, de ismereteim szerint azok fuzzy kiterjesztésével csak Kiss Attila cikke, illetve néhány, fuzzy Prologgal kapcsolatos írás foglalkozott ([LL], [PVM]). Így eredményeim újdonságnak számíthattak. Közben kimaradt néhány év, és amikor újra kezdtem olvasgatni a témáról, már jóval szélesebb körű szakirodalmat találtam. Ez természetesen nem jelenti azt, hogy a kezdeti időkből rajtunk kívül senki más sem foglalkozott a témával – sőt, ma már látom, hogy több eredmény is keletkezett abban az időben – pusztán annyit, hogy most, az Internet segítségével jóval könnyebb megtalálni ezeket a cikkeket. De azt is jelenti, hogy a kimaradt időben sokan foglalkoztak ilyen és hasonló jellegű kérdésekkel, sok érdekes eredmény született.

Régebbi cikkeim a fuzzy Datalog nyelv értelmezésével, fixpont-szemantikájának meghatározásával, különböző kiértékelési stratégiákkal, illetve a fuzzy Datalog fuzzy adatokra való kiterjesztésével foglalkoztak. Az utóbbi években kezdett izgatni a kérdés: nem lehetne-e még tovább bővíteni a fuzzy Datalog hatáskörét. Igyekeztem felhasználni korábbi eredményeimet, s azt tapasztaltam, hogy ezen az úton is eljuthatok egy bizonytalanságkezelő modellhez, egyfajta fuzzy tudásbázishoz. Erről próbáltam számot adni az előző fejezetekben.

A továbbiakban néhány egyéb, a témához kapcsolódó kutatási irányt és eredményt szeretnék bemutatni. Bár manapság annyira széles a bizonytalanságkezeléssel foglalkozó tudományágak, illetve valós alkalmazások köre, hogy önálló

tudományterületként (Soft Computing) kezdik emlegetni, megmaradok a logikai programozás területén belül, és innen mutatok be néhány elképzelést.

Mint már az Előzmények című fejezetben is említettem, vannak szerzők, akik a bizonytalanságot nem egy, hanem több számértékkel jellemzik. Többféle módon közelítik meg a témát, ezek közül most csak az intervallum értékű fuzzy halmazokról (pl. [BM2], [Na], [DP4]) és az ilyen irányú megközelítésről ejtek néhány szót.

Dubois és Prade [DP4] intervallum értékű fuzzy halmazokat tárgyalnak. Ennek kapcsán két tagsági értéket definiálnak, az egyik a beletartozás, a másik a „bele nem tartozás” értéke, vagyis egy univerzum minden u eleméhez egy $(F^+(u), F^-(u))$ párost rendelnek, ahol $F^+(u)$ a halmazba tartozás szintje, $F^-(u)$ pedig azt mutatja, hogy u mennyire nem tartozik a halmazba. (F^+ az igazság bizonyossága, azaz a pozitív evidencia foka, F^- pedig a hamisság bizonyossága, azaz a negatív evidencia foka.) Megkövetelik, hogy $F^+(u) + F^-(u) \leq 1$ legyen. Az így értelmezett intervallumokkal különböző műveleteket lehet végezni, és így módon használhatóak bizonytalan információk kezelésére.

A szerzőpár ezt az elképzelést kapcsolatba hozza a lehetőségi logika fogalomrendszerével. Egy lehetőségi eloszlást definiálnak, amelynek segítségével meghatározható egy lehetőségi-, egy szükségességi- és egy garantált lehetőségi mérték. A lehetőségi fokok annak mértékét határozzák meg, amellyel egy esemény elfogadható (elhihető), vagyis konzisztens a világ egy lehetséges állapotával, a szükségességi mérték az események bizonyosságát fejezi ki, a garantált lehetőségi mérték pedig azt mutatja meg, mennyire fogadható el a világ összes olyan állapota, amely tartalmazza a vizsgált eseményt.

A hagyományos értelemben vett fuzzy interpretációk egyébként a lehetőségi interpretációk olyan speciális esetének tekinthetőek, amelyek nem rendelnek bizonyossági értékeket a hamissághoz, csak az igazsághoz. (Pontosabban: a hamissághoz nullát rendelnek.)

Gerd Wagner [W2] kritizálja Dubois-ék elképzelését, amiért csak egyfajta negációval írják le a lehetőségi logika szükségességi és lehetetlenségi mértékeit. Szerinte kétfajta negációra lenne szükség. Az erős (vagyis a „hagyományos”) negáció ugyanis egyszerűen csak megcseréli a két bizonytalansági értéket, vagyis az eredeti (F^+, F^-) lehetőségi értékű atom negáltjához (F^-, F^+) tartozik.

Kiterjeszti a bizonytalansági mértékek szokásos $[0,1]$ skáláját a $[-1,1]$ intervallumra. A 0 a teljes bizonytalanságot jelenti (nincs se pozitív, se negatív evidencia), a -1 a teljesen biztosan hamis, vagyis lehetetlen esetet. Hasonló általánosítási utat követve, mint ahogy a közönséges Datalog esetén a háromértékű logika segítségével eljutottunk a megalapozott szemantikáig, ő a lehetőségi logika ilyen értelmezésére építve jut el egy lehetőségi megalapozott (stabil) modellig.

Érdekes megemlíteni, hogy Dubois és társai pszichológiai jellegű vizsgálatok kapcsán jutottak arra az ötletre, hogy két értékkel definiálják a bizonytalanságot. Azt tapasztalták ugyanis, hogy az emberi tudás, előítélet nem ugyanolyan természetű, mint egy mérési adat, mert az emberek bizonyos eseteket eleve elutasítanak.

Mivel az a kérdés indított a fuzzy tudásbázis egy lehetséges felépítésének megfogalmazására, hogy vajon hogyan lehetne modellezni egy szinonimákat is tartalmazó emberi beszélgetést, illetve hogyan lehet ezt úgy modellezni, hogy esetleg mesterséges ágensek (pl. internetes kereskedő ágensek) is alkalmazhassák, ezért érdemes lenne megvizsgálni, hogyan lehetne ötvözni az általam felépített modellt az intervallum-értékű bizonytalanságkezeléssel. Persze, az emberi gondolkodás modellezése, illetve annak ágensekre való leszűkítése nagyon sok egyéb irányhoz, például különböző logikák (hiedelem-, lehetőségi-, modális-, stb. [BPV], [BGR], [DHP], stb.) alkalmazásának kérdéséhez is vezethet, de nem valószínű, hogy evvel a témakörrel valaha is behatóbban foglalkoznák, hiszen véges az élet. A továbbiakban csak a témámhoz szorosabban kapcsolódó kérdésekről ejtek szót.

Bár Wagner érdekesen tágítja a negáció fogalmát, és érdekesen jut el a lehetőségi megalapozott modellig, alapjában szűkebben értelmezi a fuzzy logikai program fogalmát, mint ahogy dolgozatomban a fuzzy Datalog-ot értelmeztem. Szerinte ugyanis egy fuzzy logikai program egy (X,R) pár, ahol X fuzzy adatbázis (1. típusú, vagyis sorhoz rendel beletartozási értéket), és R közönséges logikai program. Vagyis nem engedi meg, hogy a szabályok is tükrözzenek bizonytalanságot.

Ugyancsak viszonylag egyszerűen oldja meg a bizonytalanság kezelését Rafee Ebrahim [E]. Gyakorlatilag közönséges logikai programot definiál, vagyis

„éles” szabályokat. A „fuzzyságot” úgy oldja meg, hogy minden egyes predikátum argumentumszámát kibővíti eggyel, és ez a plusz argumentum tartalmazza majd a $[0,1]$ közötti beletartozási értéket. Ily módon a predikátumok ugyancsak első típusú fuzzy relációkkal hozhatók kapcsolatba. Az így kibővített logikai program közönséges „éles” illesztéssel értékelhető ki. Véleményem szerint az én megoldásom erénye az, hogy sokkal inkább támaszkodik a fuzzy logikára, fuzzy adatokra, [E] javaslatának viszont az az előnye, hogy jóval könnyebben implementálható.

Dolgozatomban végig „éles” illesztéssel oldottam meg a célvezérelt kiértékelési algoritmusokat, és utólag határoztam meg a cél bizonytalansági értékét. Kérdés – és ez egy esetleges további kutatás célja lehet –, hogy megoldható-e a kiértékelés olyan fuzzy egységesítéssel, amely már az illesztés során számol bizonytalansági értékeket. Többek között ez a kérdés inspirál arra, hogy bemutassak néhány, a fuzzy egységesítést tárgyaló munkát. Ezek mindegyike hasonlóságon – vagyis reflexív, szimmetrikus és tranzitív reláción – alapul, eltérően az én szomszédság alapú elképzeléseimmel.

A fuzzy egységesítés témakörével több cikk is foglalkozik. Közülük talán a legelső Virtanené [V1], amely körülbelül abban az időben született, amikor az én korai cikkeim. Nyelvi változókkal értelmezi a fuzzy term fogalmát, és ezek között definiálja az egységesítési algoritmust. A nyelvi változót egy ötelemű $LV=(x, T(x), U, G, M)$ kifejezésnek tekinti, ahol U az LV univerzuma; x a nyelvi változó neve; $T(x)$ az x -hez tartozó term-halmaz, vagyis az x lehetséges nyelvi értékei (azaz nevek, kiegészítve az U -n értelmezett fuzzy számokkal); G az x értékeinek generálására szolgáló szintaktikus szabály; M pedig egy szemantikus szabály, amely minden értékhez hozzárendeli a jelentését. Az egységesítéshez szüksége van a fuzzy ekvivalencia-reláció fogalmára, ami nála egy Lukasziewicz implikáción és konjunkción alapuló hasonlósági reláció. Bár érdekes a felvetés, de véleményem szerint túl absztrakt a cikk, inkább csak elvi szinten beszél az egységesítésről, de azt a kérdést, hogy vajon hogyan lehet illeszteni pl. a magasságra vonatkozó „170” értéket az „átlagos” fuzzy termmel, illetve, hogy miként lehet eldönteni azt, hogy ez a két term ugyanahhoz az univerzumhoz tartozik, ezt nem tárgyalja, sőt, áthárítja egy esetleges programnyelv szintjére. Ugyanakkor nagyon jó kiindulási alapul szolgál, és nagyon jó keretet biztosít több konkrét elképzeléshez.

Az egyik ilyen konkretizálást maga Virtanen végzi [V2]. Szintaktikát és szemantikát definiál a nyelvi változók fuzzy konstanssal való értelmezésére, illetve egy hasonlóságon alapuló kiértékeléséhez, s erre alapozva ad meg egy egységesítési algoritmust. Wagnerhez hasonlóan ő sem rendel bizonytalanságot a szabályokhoz, de a nyelvi változók használata miatt egészen más jellegű programokat tárgyal, mint Wagner. Virtanen egységesítési eljárása is „kétlépéses”, ugyanis előbb elvégzi a lehetséges és szükséges illesztéseket, majd csak ezután számolja a bizonytalansági értékeket.

Alsinet és Godo cikke [AlGo1] Virtanen eredeti elképzelését más módon konkretizálja, de – legalábbis számomra – sokkal világosabban és érthetőbben. Nyelvi termként csak fuzzy konstansokat fogad el, sőt, azok közül is csak a normalizált trapezoid fuzzy halmazokat. Értelmezi köztük a hasonlósági relációt, és megadja a hasonlóság kiszámítási módját. Ezekre építve oldja meg az illesztést. Ez az egységesítés is „kétlépéses”, ennek ellenére – ismét egy esetleges későbbi kutatásra utalok – érdemes lenne megvizsgálni, hogy ez az egységesítési eljárás nem ötvözhető-e az általam megadott modellel, annál is inkább, mert a trapezoid fuzzy halmazokat viszonylag könnyen lehet implementálni.

Gondolkodásmódomhoz Formato, Godo, Sessa és társai észjárása és témaközelítése áll legközelebb ([AFG], [AlGo1], [S], stb.). Bár egymástól függetlenül dolgoztunk, mégis van valamennyi hasonlóság az elért eredményekben. Nagyjából azonos időben kezdtünk dolgozni a témán, de egymástól teljesen függetlenül. (Igaz, korábbi cikkeim megjelenése után Formato e-mail-ben elkérte az írásaimat, ő pedig elküldte Likelog néven fejlesztett logikai programnyelvének egy változatát [AF1], de sajnos nem alakult ki köztünk szakmai kapcsolat.) Munkásságukkal főleg mostanában, a fuzzy tudásbázisra vonatkozó vizsgálódásaim kapcsán ismerkedtem meg.

Formatóék úgynevezett felhőkkel oldják meg az egységesítést [AFG]. Egy univerzumon értelmezett hasonlósági reláció ekvivalencia osztályozást létesít az univerzum elemei között. Azt, hogy milyen tág vagy szűk egy-egy ilyen osztály, a hasonlóság foka határozza meg, hiszen az adott mértékben hasonló elemek kerülnek egy osztályba. Ezeket az osztályokat nevezik felhőknek, és ezek segítségével definiálják a kiterjesztett termék fogalmát, illetve megadnak egy algoritmust két azonos argumentumszámú kiterjesztett term egységesí-

tésére. Kicsit leegyszerűsítve: a felhők közötti egységesítést oldják meg, a szükséges bizonytalansági szintet pedig – az egységesítési algoritmus végén, vagyis szintén a „második” lépésben – a felhők képzéséhez felhasznált hasonlósági szintek alapján számolják. Bár úgy tűnik, kapcsolatba hozható az általam definiált szomszédsági halmaz (7.1. definíció) a felhők fogalmával, de míg a szomszédsági halmazba egy adott elemmel valamilyen szinten szomszédos elemek kerülnek, addig felhőt az egy adott szinten hasonló elemek alkotnak. Ráadásul a szomszédsági halmaz alapja a szomszédsági reláció, míg a felhőké a hasonlósági. Ennek ellenére nem kizárt, hogy kapcsolatba hozható a két tárgyalásmód, de ez is egy esetleges későbbi kutatás célja lehet.

Sessa [S] továbbviszi Formatoék elképzelését, és az általuk definiált egységesítési algoritmus alapján tárgyalja a hasonlóság-alapú SLD rezolúciót. Közönséges (vagyis a klasszikus elsőrendű logikára alapuló) logikai programot tekint, amelyet kibővít egy, a predikátum- és függvényszimbólumokra is vonatkozó hasonlósági relációval. Az ily módon kibővített programra alkalmazza a hasonlóság-alapú rezolúciót. A rezolúciós bizonyítás elvégzése után a fennálló hasonlóságok alapján tud bizonytalansági értéket rendelni a kapott eredményhez. Ez az elképzelés és a megadott algoritmus érdekes bizonytalanságkezelési modellt épít fel, amely azonban több ponton eltér az általam megalkotott modelltől: az enyém a fuzzy logikára épül, ez nem, én szomszédsági relációt használok, ő hasonlóságit.

Az előzőektől eltérő ötletre épít Schroeder és Schweimeier [SS1], [SS2], [SS3]. A szerzők az esetleges elírások kezelésére dolgoznak ki egy számítási módszert, az elírásból adódó hibák számával definiálják a termék és predikátumok távolságát, és ezeknek a távolságoknak a felhasználásával értelmezik a fuzzy egységesítést. A bioinformatikában a genomsorozatok összehasonlítására is használt szerkezeti távolság (edit distance) azt mutatja meg, hogy két karaktersorozat hány helyen tér el egymástól. Ennek a távolságfogalomnak a segítségével értelmezni tudják a termék távolságát is, és így meg tudják határozni, hogy két termék milyen szinten egységesíthető. Ez ugyan „egylépéses” egységesítés, vagyis az egységesítés során azonnal számol bizonytalansági értéket is, viszont csak szűk területen, szintaktikai és nem tartalmi különbségek, vagyis nem szinonimák esetén használható.

A fuzzy egységesítéssel foglalkozó témaköröknek búcsút mondva szeretnék még megemlíteni két további cikket:

Izgalommal olvastam Yager és társainak új fuzzy implikációs osztályról szóló írását [TKY], hiszen érdekelt, vajon kibővíthető-e a fuzzy Datalog kapcsán alkalmazott implikációs operátorok köre. A cikk ugyanis egy újabb – a legtöbb elvárt feltételnek eleget tévő – implikációs operátor, az

$$I(\alpha, \beta) = \begin{cases} \beta^\alpha & \alpha > 0, \beta > 0 \\ 1 & \alpha = \beta = 0 \end{cases}$$

felfedezéséről számol be.

Természetesen érdeklődéssel vizsgáltam meg, hogy az új operátor ismeretében módosul-e valamennyire a fuzzy Datalogról kialakult kép, de rövidesen kiderült, hogy legföljebb csak elméleti szinten, hiszen ez az új operátor nem termináló, vagyis használata esetén nem garantált a program megállása. (Pl. a 3.1. példa ilyen operátor esetén sem terminál.)

A másik cikk Hsuan-Shih Lee munkája [Lee]. Egy hatékony eljárást ad arra vonatkozóan, hogyan lehet meghatározni egy szomszédsági mátrix tranzitív lezártját. (A cikk címében szereplő hasonlóságon – mint egy-két korábbi cikkemben magam is – egy reflexív és szimmetrikus relációt, vagyis szomszédsági relációt ért.) A tranzitív lezárt kérdése azért lehet érdekes, mert – mint a 6.2. állításban speciális esetre magam is bizonyítottam, illetve a fent felsorolt munkák mindegyike alátámasztja – a tranzitivitás, vagyis szomszédság helyett hasonlóság szerepeltetése, sok előnnyel járhat. Ugyanakkor maga a tranzitív lezárt fogalma komoly problémákat okozott számomra, ugyanis nagyon nagy eltérés van az eredeti szomszédsági mátrix és tranzitív lezártjának elemei között. A kérdés lényege a következő: az elemek közötti „hasonlóság” meghatározásakor nem feltétlenül kell megkövetelni a tranzitivitást (pl. egy gyerek hasonlíthat mindkét szülőjére, de attól még a szülőknek nem kell hasonlítaniuk egymásra), ugyanakkor – könnyebb kezelhetőség, hatékonyság, stb. miatt – jó lenne tranzitív relációval foglalkozni. A tranzitív lezárt viszont nagyon megváltoztathatja az eredeti hasonlósági értéket, vagyis alkalmazásával irreális eredményekre számíthatunk. A kérdést e-mail-ben Lee-nek is felvettem. Válaszul egy nagyon jónak tűnő ötletet kaptam: nem a tranzitív lezárttal, hanem a szomszédsági mátrixhoz „legközelebbi” hasonlósági mátrixsal kellene foglalkoznom. Kis méretű konkrét mátrixok esetén sikerült találnom ilyen megoldást,

általános esetben és behatóan azonban még nem tudtam foglalkozni a kérdéssel. Terveim között szerepel, hogy alaposabban elmélyüljek ebben a problémában, hacsak Lee meg nem oldotta azóta.

Végül még egy élményemet szeretném leírni néhány szóban. Volt szerencsém meghallgatni a nagyon kedves és barátságos Lofti Zadeh barcelonai előadását [Z3], ahol – Vörösmarty „ment-e a könyvek által a világ elébb” (Gondolatok a könyvtárban) kérdésfelvetéséhez hasonlóan – feltette a kérdést: „ment-e a fuzzy logika által a világ elébb?”, vagyis közelebb jutottunk-e a természetes nyelven megfogalmazott tudás gépi intelligenciává alakításához? Rögtön előadása elején leszögezte, hogy nem. Ennek ellenére az egész előadás alatt lelkesen beszélt arról, hogyan lehet a fuzzy logikával, illetve a ráépült technológiákkal, elméletekkel, leíró nyelvekkel mégis használható módon átalakítani, számszerűsíteni, a gépi intelligencia számára kezelhetővé tenni az emberi tudást, illetve annak egy, sőt, egyre növekvő részét.

Zárszó

Dolgozatomban a fuzzy logikára épülő bizonytalanságkezelési modellek bemutatásával foglalkoztam. Természetesen több hasonló modell lehet, és van is, hiszen a bizonytalanságot sohasem lehet teljesen biztosan leírni, csak megközeleltetni. Az általam felépített modell a Datalog fuzzy kiterjesztésére épül. A közönséges Datalog program szabályait kiegészítettük egy operátorhalmazból származó implikációs operátorral és egy bizonytalansági szinttel, amely azt mutatja, milyen mértékben tekinthető igaznak egy predikátum vagy egy szabály. Ezek segítségével tudtunk következtetni a szabályfej bizonytalansági szintjére. Az így kapott nyelvet neveztük fDATALOG-nak.

A nyelvhez egy determinisztikus és egy nem determinisztikus szemantikát is rendeltünk, melyek negációmentes esetben megegyeznek egymással, negációt is tartalmazó esetben azonban csak a nem determinisztikus szemantika alkalmazható. Rétegezhető programok esetén meghatározható a program minimális modellje, de egy esetleges további vizsgálatot igényelne a nem rétegezhető programok szemantikájának kérdése, és ugyancsak érdemes lenne megvizsgálni a függvényszimbólumokat is tartalmazó programok körét.

A fDATALOG szabályok kiértékeléséhez kétféle stratégiát, egy adatvezérelt és egy célvezérelt megoldási lehetőséget tárgyaltam. Érdemes lenne esetleges egyéb, hatékonyabb stratégiákon is gondolkozni, illetve alaposabban megvizsgálni, hogy a közönséges Datalog esetén ismert „kevert” módszerek hogyan alkalmazhatóak fuzzy esetre.

Szó esett a fDATALOG és a fuzzy relációk kapcsolatáról is. Ez vetette fel a kérdést, hogyan lehetne kiterjeszteni a fDATALOG hatáskörét fuzzy adatok kezelésére is. A kérdésre megfogalmaztam egy, a szomszédsági transzformáción alapuló lehetséges választ, érdemes lenne azonban egyéb lehetőségek után is kutatni.

További általánosításként eljutottunk a fuzzy tudásbázis fogalmához. Ezt egy négyelemű halmazként definiáltuk, amelynek egyik eleme egy fuzzy következtetési rendszer, a fuzzy Datalog; másik eleme a háttértudás, amelyet a termek és

predikátumok közötti szomszédság segítségével adunk meg; harmadik eleme egy összekapcsolási algoritmus, amely összeköti a háttértudást és a következtési mechanizmust; negyedik eleme pedig a program dekódoló függvényeinek halmaza, amelyek segítségével meg tudjuk határozni az eredmény bizonytalansági szintjét. Dolgozatomban kétféle összekapcsolási algoritmust mutattam be, mindkettőhöz kiértékelési stratégiát is értelmeztem. Megvizsgálandó kérdés maradt azonban az, hogy vajon lehet-e hatékonyabb kiértékelést találni a meglévő összekapcsolási módszerekhez, illetve, hogy lehet-e más összekapcsolásokat értelmezni.

Dolgozatom írása közben számtalan további, megoldásra váró kérdés fogalmazódott meg bennem. Érdemes lenne megvizsgálni, hogyan lehetne ötvözni az általam felépített modellt az intervallum-értékű bizonytalanságkezeléssel, hiszen sok esetben nem csak az a kérdés, mennyire lehet igaz egy állítás, de az is, hogy mennyire tekinthető hamisnak. Az is izgat, hogyan lehetne megkeresni egy adott szomszédsági mátrixhoz legközelebbi hasonlósági mátrixot, hiszen a hasonlóságra alapozva sok probléma, többek között a fuzzy egységesítés kérdése is sokkal egyszerűbben és hatékonyabban kezelhető. Az is érdekelne, és azt gondolom, az már csak kis lépés innen, hogyan lehetne megvalósítani két, saját fuzzy tudásbázissal rendelkező mesterséges ágens „párbeszédét”, alkudozását (pl. internetes kereskedő ágensek). Végül, a sok kisebb kérdést kihagyva, az is motoszkál bennem, hogy vajon miként lehetne valamilyen struktúrába rendezni a háttértudást – ahogy az emberi gondolkodás is teszi –, és a struktúra szerkezetéből származó előnyöket kihasználni a tudásbázis kiértékelésekor. Ezek megválaszolása azonban további kutatómunka eredménye lehet.

Summary

The basic question of this dissertation is: how we can give a possible model for handling uncertain information. This model is worked out in the framework of Datalog.

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 ([CGT], [U]) 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. The second chapter of this dissertation summarize the main concepts of deductive databases, according to [AV1], [AV2], [AV3], [AHV], [CGT],[GM], [LLS], [Pr1], [Pr2], [U], etc.

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. About the concept of deductive databases and fuzzy logic can be read in the classical works, for example in [CGT], [DP], [L], [N], [U].

A few years ago in [AK1], [A1] and [AK3] there was given a possible combination of Datalog-like languages and fuzzy logic. In these works – which are the bases of third and fourth chapter of this dissertation – there was introduced the concept of fuzzy Datalog by completing the Datalog-rules and facts by an uncertainty level and an implication operator. In [AK2] – which is the base of sixth chapter – there was given an extension of fuzzy Datalog to fuzzy relational databases.

In the seventh chapter, continuing the former concept, there is given a possible model for 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. A possible evaluation strategy is also given.

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. They consider the similarity as reflexive, symmetric and transitive relation. Some of them take any classification according to similarities and use this equivalence classes to make unification and fuzzy resolution, for example [AFG], [S]. In [V1], [V2] the author deals with linguistic variables and their linguistic values, and give definition of the fuzzy unification algorithm by means of these values. In [SS1] the authors discuss a special kind of fuzzy unification. They 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. The eighth chapter deals with these concepts.

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

The 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 $r;\beta;I$, where r is a formula of the form

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

A is an atom (the head of the rule), A_1, \dots, 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.

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 $DTP: F(BP) \rightarrow F(BP)$ and $NTP: F(BP) \rightarrow F(BP)$ are defined as

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

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

respectively, where

$$(A \leftarrow A_1, \dots, A_n; I; \beta) \in \text{ground}(P), (|A_i|, \alpha_{A_i}) \in X, 1 \leq i \leq n,$$

$$\alpha_A = \max(0, \min\{\gamma \mid I(\alpha_{\text{body}}, \gamma) \geq \beta\}).$$

$|A_i|$ 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).

In the third chapter it was proved, that starting from the set of facts, both DT_P and NT_P have fixpoints, which are the least fixpoints in the case of positive (negation-free) P . These fixpoints are denoted by $\text{lfp}(DT_P)$ and $\text{lfp}(NT_P)$. It was also proved, that $\text{lfp}(DT_P)$ and $\text{lfp}(NT_P)$ are models of P , so we could define $\text{lfp}(DT_P)$ as the deterministic semantics and $\text{lfp}(NT_P)$ 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 $\text{lfp}(DT_P)$ is not

always a minimal model, but the nondeterministic semantics – $\text{lfp}(\text{NT}_p)$ – is minimal under certain conditions. This condition is the stratification. Stratification gives an evaluating sequence in which the negative literals are evaluated first. (Detailed in the third chapter.)

The fourth chapter deals with the evaluation strategies of fuzzy Datalog. 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.

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 this chapter.

Extension of fDATALOG for Fuzzy Data

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 . [CGT], [U].

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 [K1], [LL], [RM], [Um]. In the fifth chapter, relying on [K1], 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, \dots, D_n is characterised by the membership function: $\mu_R : D_1 \times \dots \times D_n \rightarrow [0,1]$

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

Example 1.

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 aim of this chapter 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 $\text{prox}_D: D \times D \rightarrow [0,1]$ of $D \times D$ such that the following properties hold:

- $\text{prox}_D(x,x) = 1$ for any $x \in D$ (reflexivity)
- $\text{prox}_D(x,y) = \text{prox}_D(y,x)$ for any $x,y \in D$ (symmetry).

In this part of dissertation there is given an algorithm for rewriting the rules according to proximity, now I show only an example:

Example 2.

For the data of example 1, let us correspond to the relation R the predicate "person" with arguments of R's tuples. Consider the next rule:

$$\text{head}(x) \leftarrow \text{person}(x, \text{about_40}, \text{about_80}); 0.8; I.$$

The rewritten rule is

$$\text{head}(x) \leftarrow \text{person}(x_1, y_1, z_1), \text{prox}_1(x, x_1), \\ \text{prox}_2(\text{about_40}, y_1), \text{prox}_3(\text{about_80}, z_1); 0.8; I.$$

Of course it is necessary to define the above proximity predicates, for example $\text{prox}_2(\text{about_40}, \text{middle aged}) = 0.9$, etc.

Fuzzy knowledgebase

In the seventh chapter, continuing the concept of the previous part, there is a further extension of fDATALOG to a possible fuzzy knowledgebase. 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:

Definition 7. Let $d \in D$ any element of domain D . The proximity set of d is a fuzzy subset over D : $SD_d = \{(d_1, \lambda_1), (d_2, \lambda_2), \dots, (d_n, \lambda_n)\}$, where $d_i \in D$ and $\text{prox}D(d, d_i) = \lambda_i$ for $i = 1, \dots, n$.

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 their 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 :

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

so that

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

$$\varphi(\alpha, 1, 1, \dots, 1) = \alpha, \text{ and}$$

$$\varphi(\alpha, \lambda, \lambda_1, \dots, \lambda_n) \text{ is monotone increasing in all arguments.}$$

It is worth mentioning that any triangular norm is suitable for decoding function, for example the 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.

In this chapter there was defined two kind of modifying algorithm. According the first one we modify the program and use the original consequence transformation for evaluation; in the second case a modified consequence transformation is applied for the original program.

Simple modification (algorithm mA1):

Let P be a fuzzy Datalog program, and let B_k be any background knowledge. By the proximities of the background knowledge we can define the modified fDATALOG program, mP : Let us replace each predicate symbol p of the program P by SR_p , each ground term of P by SC_t and each variable x by $X=\{x\}$. mP is evaluable as an ordinary fDATALOG program. 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.

Transformation modification (algorithm mA2):

In this case a modified consequence transformation is applied for the original program. To define the modified transformation's domain, let us extend P 's Herbrand universe with all possible ground terms occurring in background knowledge – so we get the modified Herbrand universe, mH_p . Let the modified Herbrand base, mB_p be the set of all possible ground atoms whose predicate symbols occur in $P \cup B_k$ and whose arguments are elements of mH_p . The modified consequence transformation is defined over this modified Herbrand base, so we can deduce to the atoms being in the rule-heads and the atoms being similar to them.

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

In this chapter there was given top-down evaluation algorithms for both case of modifying.

Irodalomjegyzék

- [A1] Ágnes Achs: Evaluation Strategies of Fuzzy Datalog
Acta Cybernetica, 1997., Szeged (85-102.)
- [A2] Ágnes Achs: Fuzzy Datalog with Background Knowledge,
Teaching Mathematics and Computere Science, 3(2005)2,
Debrecen, (1-25.)
- [A3] Ágnes Achs: Fuzzy knowledge-base with fuzzy Datalog
IEEE 4th International Conference on Intelligent System Design and
Application, Budapest, 2004. aug. 26-28, (55-60.)
- [A4] Ágnes Achs: Computed answer based on fuzzy knowledgebase
EUSFLAT-LFA 2005 – Fourth Conference of the European Society
of Fuzzy Logic and Technology, September. 7-9, Barcelona, (94-99.)
- [A5] Achs Ágnes: Mennyire biztos a bizonytalan? - Gondolatok egy
fuzzy tudásbázisról
(előadás + konferenciakiadvány), Informatika a felsőoktatásban 2005,
Konferencia, Debrecen, 2005. augusztus 24-26. (6 oldal)
- [A6] Achs Ágnes: Datalog alapú fuzzy tudásbázis
megjelenőben: Alkalmazott Matematikai Lapok
- [A7] Ágnes Achs: Computed answer from uncertain knowledge
– a model for handling uncertain information –
elküldve: European Journal of Operational Research
- [A8] Ágnes Achs: Creating and evaluating of fuzzy knowledgebase
elküldve: Journal of Universal Computer Science
- [AK1] Ágnes Achs, Attila Kiss: Fixpoint query in fuzzy Datalog programs
Annales Universitatis Scientiarum Budapestiensis, Sectio
Computatorica, 1995., Budapest (223-231.)

- [AK2] Ágnes Achs, Attila Kiss: Fuzzy Extension of Datalog
Acta Cybernetica, 12 (1995), Szeged (153-166.)
- [AK3] Achs Ágnes-Kiss Attila: A Datalog fuzzy kiterjesztése
Alkalmazott Matematika Lapok, 1998., Budapest (111-138.)
- [AF1] F.Arcelli and F.Formato, "LIKELOG: a logic programming
language for flexible data retrieval", Proc. of Symposium
on Applied Computing, SAC99, ACM Press, San
Antonio, Texas, March 1999.
- [AF2] Francesca Arcelli Fontana, Ferrante Formato: Unification as
Negotiation: a Context-based Approach
IFSA/NAFIPS Conference 2001 July 25-28, Vancouver (5 oldal)
- [AFG] F.Arcelli, F.Formato and G.Gerla: Fuzzy Unification as
Foundations of Fuzzy Logic programming
in Logic Programming and Soft Computing, Ed. RSP-Wiley, England,
1998.
- [AG] S.Abiteboul, S.Grumbach: A Rule- Based Language with
Functions and Sets
ACM Transactions on Database Systems, Vol. 16, No.1, March,
1991. (1-30.)
- [AHV] S. Abiteboul, R. Hull, V. Vianu: Foundations of Databases
Addison-Wesley Publishing Company, 1995.
- [AlGo1] Teresa Alsinet, Lluís Godo: Fuzzy unification degree
Logic Programming and Soft Computing - Theory and Applications,
A Post-conference Workshop of JICSLP'98, 1998.
Joint International Conference and Symposium on Logic
Programming, 15-19 June 1998. Manchester, UK.
- [AlGo2] Teresa Alsinet, Lluís Godo: Adding similarity-based reasoning
capabilities to a Horn fragment of possibilistic logic with fuzzy
constants

Fuzzy Sets and Systems 144 (2004) (43-65.)

- [ASV] Serge Abiteboul, Eric Simon, Victor Vianu: Non Deterministic Languages to Express Deterministic Transformations
Symposium on Principles of Database Systems 1990, Nashville, Tennessee (218-229.)
- [AV1] S.Abiteboul, V.Vianu: Datalog Extensions for Database Queries and Updates
INRIA, 1988. (1-65.)
- [AV2] S.Abiteboul, V.Vianu: Fixpoint Extension of First-Order Logic and Datalog- like Languages
Proc. Fourth Annual Symposium on Logic in Computer Science, Asilomar, California, 1989. (71-79.)
- [AV3] S.Abiteboul, V.Vianu: Procedural Languages for Database Queries and Updates
Journal of Computer and System Sciences 41, 1990. (181-219.)
- [AV4] S.Abiteboul, V.Vianu : Expressive Power of Query Languages in J. Ullman, editor, Theoretical Studies in Computer Science. Academic Press, 1992. Festschrift in Honour of Seymour Ginsburg's 64th Birthday.
- [B] J.F.Baldwin: An Uncertainty Calculus for Expert Systems
Approximate Reasoning in Intelligent Systems, Decision and Control, ed. by E.Sanchez, L.A.Zadeh, Pergamon Press, Oxford, England, 1987. (33-54.)
- [BEGS] B.Bouchon-Meunier, F.Esteva, L.Godo, M.Rifqi, S. Sandri:
A principled approach to fuzzy rule base interpolation using similarity relations
EUSFLAT-LFA 2005 – Fourth Conference of the European Society of Fuzzy Logic and Technology, September.7-9, Barcelona, (757-763.)

- [BGR] Luciano Blandi, Lluís Godo, Ricardo O. Rodríguez: A connection between Similarity Logic Programming and Gödel Modal Logic
EUSFLAT-LFA 2005 – Fourth Conference of the European Society of Fuzzy Logic and Technology, September.7-9, Barcelona, (775-780.)
- [BM1] J. F. Baldwin, T. P. Martin: An abstract mechanism for handling uncertainty
in: Uncertainty in Knowledge Bases, ed. by B.Bouchon-Meunier, R.R.Yager, L.A. Zadek, 3rd International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, IPMU'90 Paris, France, July 2-6, 1990, Springer-Verlag (126-135.)
- [BM2] J. F. Baldwin, T. P. Martin: Towards Inductive Support Logic Programming
IFSA/NAFIPS Conference 2001 July 25-28, Vancouver (6 oldal)
- [Bo] Bernadette Bouchon-Meunier: Questionnaires and fuzziness
in: An Introduction to fuzzy logic applications in intelligent systems, ed.by R.R. Yager, L.A. Zadeh, Kulwer Academic Publishers, 1992. (221-231.)
- [BPS] B.P.Buckles, F.E.Petry, H.S.Sachar: A Domain Calculus for Fuzzy Relational Databases
Fuzzy Sets and Systems 29 (1989) (327-340.)
- [BPV] A.D.C. Bennett, J.B. Paris, A. Vencovská: A New Criterion for Comparing Fuzzy Logics for Uncertain Reasoning
Journal of Logic, Language, and Information 9: 2000, Kluwer Academic Publishers (31-63.)
- [C] Daniel E. Cohen: Computability and Logic
Ellis Horwood Limited Publishers, Chicester, 1987.
- [CGT] S.Ceri, G.Gottlob, L.Tanca: Logic Programming and Databases
Springer-Verlag Berlin, 1990.

- [CL] Ching-Liang Chang, Richard Char, Tung Lee: Symbolic Logic and Mechanical Theorem Proving
Academic Press, 1973, New York
- [D] Subrata Kumar Das: Deductive Databases and Logic Programming
Addison-Wesley Publishing Company. 1992.
- [DDP] Demetrovics János, Jordan Denev, Radiszlav Pavlov:
A számítástudomány matematikai alapjai (2. kiadás)
Tankönyvkiadó, Budapest, 1989.
- [DHP] Didier Dubois, Petr Hajek, Henri Prade: Knowledge Driven versus data-driven logic
Journal of Logic, Language and Information, 9 (2000), Kluwer Academic Publishers, (65-89.)
- [DLP] Didier Dubois, Jerome Lang, Henri Prade: Fuzzy sets in approximate reasoning, Part 2: Logical approaches
Fuzzy Sets and Systems 40 (1991) (203-244.)
- [DP1] Didier Dubois, Henri Prade: Fuzzy sets in approximate reasoning, Part1: Inference with possibility distributions
Fuzzy Sets and Systems 40 (1991) (143-202.)
- [DP2] Didier Dubois, Henri Prade: Fuzzy rules in knowledge-based systems – Modelling gradedness, uncertainty and preference
in: An Introduction to fuzzy logic applications in intelligent systems, ed.by R.R. Yager, L.A. Zadeh, Kulwer Academic Publishers, 1992. (45-68.)
- [DP3] Didier Dubois, Henri Prade: Fuzzy sets in approximate reasoning: A Personal View
Fuzzy logic: implementation and applications, John Wiley & Sons, Inc. New York, 1996. (3-35.)
<http://citiseer.ist.psu.edu/57432.html>

- [DP4] Didier Dubois, Henri Prade: Interval-valued Fuzzy Sets, Possibility Theory and Imprecise Probability
EUSFLAT-LFA 2005 – Fourth Conference of the European Society of Fuzzy Logic and Technology, September. 7-9, Barcelona, (314-319.)
- [E] Rafee Ebrahim: Fuzzy logic programming
Fuzzy Sets and Systems 117, 2001. (215-230)
- [G] Siegfried Gottwald: Fuzzy Sets and Fuzzy Logic
Verlag Vieweg, Wiesbaden, 1993.
- [Ge] Van Gelder: The Alternating Fixpoint of Logic Programs with Negation
Proceedings of the Eight ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, March 29-31, 1989. Philadelphia, Pennsylvania (1-10.)
- [GL] Michael Gelfond, Vladimir Lifschitz: Classical Negation in Logic Programs and Disjunctive Databases
New Generation Computing, 9, 1991. OHMSHA LTD. (365-385)
- [GM] John Grant, Jack Minker: The impact of Logic Programming on Databases
Communications of the ACM, 1992. Vol 35. No. 3. (67 – 81.)
- [GRS] Van Gelder, A. Ross, Schlipf: The well-founded semantics for general logic programs
J. ACM 38, (1991), (620-650.)
- [GS] Yuri Gurevich, Saharon Shelah: Fixed-point extensions of firstorder logic
IEEE Symp. on FOCS (1985), (346-353.)
- [HU] John E.Hopcroft, Jeffrey D.Ullman: Formal Languages and their Relation to Automata
Addison - Wesley Publishing Company, 1969.

- [K1] Attila Kiss: On the least models of fuzzy Datalog programs
International Conference on Information Processing and
Management of Uncertainty in Knowledge-based system,
Mallorca, 1993. (465-471.)
- [K2] Kiss Attila: Relációs adatbázisok függőségeinek vizsgálata
Kandidátusi értekezés, Bp, 1990.
- [KLV] Stanislav Krajci, Ratislav Lencses, Peter Vojtas: A data model for
annotated programs
Electronic Notes in Theoretical Computer Science, Vol. 66., 5.,
2002.
- [Ko] Robert Kowalski : Logic for Problem Solving
Elsevier Science Publishing Co., 1987.
- [L] J.W.Lloyd : Foundations of Logic Programming
Springer-Verlag, Berlin, 1987.
- [Le] Richard C.T.Lee: Fuzzy Logic and the Resolution Principle
Journal of the Association for Computing Machinery, Vol 19,
No.1., January 1972. (109-119.)
- [Lee] Hsuan-Shih Lee: An optimal algorithm for computing the max-min
transitive closure of a fuzzy similarity matrix
Fuzzy Sets and Systems 123 (2001) (129-136.)
- [Li] Vladimir Lifschitz: On the Declarative Semantics of Logic Programs
with Negation
in: Foundations of Deductive Databases and Logic Programming
ed.by Minker, Morgan Kaufmann Publishers, Inc. 1988.
- [LL] Deyi Li, Dongbo Liu: A Fuzzy PROLOG Database System
Research Studies Press LTD., Taunton,Somerset, England, 1990.
- [LLS] D.Laurent, V.Phan Luong, N.Spyratos: Updating Intensional
Predicates in Deductive Databases

9th International Conference on Data Engineering, 1993. Wien

- [M] Zohar Manna: Programozáselmélet
Műszaki Könyvkiadó, Budapest, 1981.
- [MPV] J.M.Medina, O.Pons, M.A.Vila: Gefred: A Generalized Model for
Fuzzy Relational Databases
Information Sciences 76., 1994. (87-109.)
- [MS] A. Melton, S. Sheno: Fuzzy relations and fuzzy relational databases
Computers Math. Applic, Vol. 21, No. 11/12, 1991. (129-138)
- [Mu] V.Murali: Fuzzy equivalence relations
Fuzzy Sets and Systems 30, 1989. (155-163)
- [N1] Vilém Novák: Fuzzy sets and their applications
Adam Hilger Bristol and Philadelphia, 1989.
- [N2] Vilém Novák: Fuzzy sets in natural language processing
in: An Introduction to fuzzy logic applications in intelligent systems,
ed.by R.R. Yager, L.A. Zadeh
Kulwer Academic Publishers, 1992. (185-199.)
- [Na] Benedek Nagy: A General Fuzzy Logic Using Intervals
Proceedings of the 6th International Symposium of Hungarian
Researchers on Computational Intelligence, 2005. nov.18-
19.,Budapest (613-624)
- [O1] Sergei Ovchinnikov: On modelling fuzzy preference relations
in: Uncertainty in Knowledge Bases, ed. by B.Bouchon-Meunier,
R.R.Yager, L.A. Zadek, 3rd International Conference on Information
Processing and Management of Uncertainty in Knowledge-Based
Systems, IPMU'90 Paris, France, July 2-6, 1990, Springer-Verlag
(154-164.)
- [O2] Sergei Ovchinnikov: Similarity relations, fuzzy partitions, and fuzzy
ordering

Fuzzy Sets and Systems 40 (1991), (107-126.)

- [P1] Pásztorné Varga Katalin: A matematikai logika és alkalmazásai
Tankönyvkiadó, Budapest, 1986.
- [P2] Pásztorné Varga Katalin, Várterész Magda: A matematikai logika
alkalmazásszemléletű tárgylása
Panem, Budapest, 2003.
- [Pr1] Teodor Przymusinski: Every Logic Program Has a Natural
Stratification And an Iterated Least Fixed Point Model
Proceedings of the Eight ACM SIGACT-SIGMOD-SIGART
Symposium on Principles of Database Systems, March 29-31, 1989.
Philadelphia, Pennsylvania (11-21.)
- [Pr2] Halina Przymusinska, Teodor Przymusinski: Semantic Issues in
Deductive Databases and Logic Programs
Formal Techniques in Artificial Intelligence A Sourcebook ed.:
R.B. Banerji., Elsevier Science Publishers B.V. (North Holland),
1990. (321-367.)
- [Pr3] Teodor Przymusinski: Stable Semantics for Disjunctive Programs
New Generation Computing, 9, 1991. OHMSHA LTD. (401-424)
- [PVM] O.Pons, M.A.Vila, J.M.Medina: Interface between Fuzzy
Relational Databases and Prolog
Technical Report, University of Granada
- [R] E.H.Ruspini : On the Semantics of Fuzzy Logic
International Journal of Approximate Reasoning 1991;5 (45-88.)
- [RM] Raju K.V.S.V.V., Majumdar A.K.: The study of Joins in Fuzzy
Relational Databases
Fuzzy Sets and Systems 21 (1987) (19-34.)
- [RS] L.G.Rios-Filho, S.A. Sandri: Contextual fuzzy unification

Proceedings 5th IFSA'95 Conference, Sao Paulo, Brazil, 1995.
(81-84.)

- [S] Maria I. Sessa: Approximate reasoning by similarity-based SLD resolution
Theoretical Computer Science 275(2002) (389-426.)
- [SMF] S.Shenoi, A.Melton, L.T.Fan: An equivalence classes model of fuzzy relational databases
Fuzzy Sets and Systems 38 (1990) (153-170.)
- [SS1] Michael Schroeder, Ralf Schweimeier: Fuzzy Argumentation an Extended Logic Programming
Proceedings of ECSQARU Workshop Adventures in Argumentation, Toulouse, Sept. 2001.
URL: <http://citeseer.ist.psu.edu/schroeder01fuzzy.html>
- [SS2] Michael Schroeder, Ralf Schweimeier: Fuzzy Unification and Argumentation for Well-founded Semantic
Annals of Mathematics and Artificial Intelligence (Oct. 2002.)
URL: www.soi.city.ac.uk/~msch/abstracts/jamai02.html (40 oldal)
- [SS3] 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.
- [Su] V.S. Subrahmanian: Paraconsistent disjunctive deductive databases
Theoretical Computer Science 93, 1992., (115-141)
- [SZ] Domenico Saccà, Carlo Zaniolo: Stable Models and Non-Determinism in Logic Programs with Negation
Proceedings of the Ninth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, April 2-4, 1990. Nashville Tennessee (205-217.)

- [TKY] I.B.Türksen, V.Kreinovich, R.R.Yager: A new class of fuzzy implications. Axioms of fuzzy implication revisited
Fuzzy Sets and Systems 100, 1998. (267-272)
- [U] J.D.Ullman : Principles of database and knowledge-base systems
Computer Science Press, Rockville, 1988.
- [Um] M. Umamo: Retrieval from Fuzzy Database by Fuzzy Relational Algebra
Knowledge Information Systems, Medical Applications, IFAC
Fuzzzy Information, Marseille, France, 1983. (1-6)
- [V1] Harry E. Virtanen: Fuzzy unification
5th International Conference on Information Processing and
Management of Uncertainty in Knowledge-Based Systems, 1994. July
4-8, Paris
- [V2] Harry E. Virtanen: Vague Domains, S-Unification and Logic
Programming
Electronic Notes in Theoretical Computer Science 66, No.5, 2002.
(18 old.)
URL: <http://www.elsevier.nl/locate/entcs/volume66.html>
- [VCMP]M.A.Vila, J.C.Cubero, J.M.Medina, O. Pons: Logic and Fuzzy
Relational Databases: a new Language and a new Definition
Fuzzy Sets and Possibility Theory in Database Management Systems,
ed. by P. Bosc and J. Kacprzyk, Physica Verlag, 1993.
- [VCMP]M.A.Vila, J.C.Cuberto, J.M.Medina, O.Pons: On the Use of Logical
Definition on Fuzzy Reletional Database
2nd IEEE International Conference on Fuzzy Systems,1993.(489-495.)
- [VMPC]M.A.Vila, J.M.Medina, O.Pons, J.C.Cuberto, M.Prados, J.Diaz:
A Knowledge Representation Model for Fuzy Databases
ACM Trans. on Database Systems, 1993.

- [Vo] Peter Vojtas: Fuzzy logic programming
Fuzzy Sets and Systems 124, 2001. (361-370)
- [VSCCK]L.Vanderberghe, A. Van Schooten, R. De Caluwe, E.E. Kerre:
Some Practical Aspects of Fuzzy Database Techniques: an Example
Information Systems Vol. 14. No. 6., 1989. (465-472)
- [W1] Gerd Wagner: Vivid Logic – Knowledge-Based Reasoning with Two
Kind of Negation
Springer-Verlag Berlin Heidelberg 1994.
- [W2] Gerd Wagner: Negation in Fuzzy and Possibilistic Logic Programs
Logic Programming and Soft Computing Research Studies Press, 1998.
URL: <http://citeseer.ist.psu.edu/359516.html>
- [Z1] Lofti. A. Zadeh : Fuzzy sets
Inf. Control 8. (1965) (338-353.)
- [Z2] Lofti A. Zadeh: Knowledge representation in fuzzy logic
in: An Introduction to fuzzy logic applications in intelligent systems,
ed.by R.R. Yager, L.A. Zadeh, Kulwer Academic Publishers, 1992.
(1-25.)
- [Z3] Lofti A. Zadeh: Fuzzy Logic as a Basis for a Theory of Precisiation of
Meaning (TPM)—The Concept of Cointensive Precisiation
prezentáció, elhangzott: EUSFLAT-LFA 2005 – Fourth Conference of
the European Society of Fuzzy Logic and Technology, September 8,
Barcelona

Melléklet

Mivel a deduktív adatbázisok speciális logikai programok, ezért világának felépítéséhez szükségünk van a logikai alapfogalmak tárgyalására.

Elsőrendű logika

A logika két alkotóelemre bomlik: a nyelvre, melyen az ismeretek kifejezhetők, és a következtetési rendszerre, („gondolkodási szabályokra”), melyek segítségével a konkrétan megadott ismeretekből bizonyos „többlet-tudást” származtathatunk.

A nyelv valós objektumok leírására szolgál, s segítségével ezekről az objektumokról fogalmazunk meg formális állításokat, melyek egy-egy interpretáció révén válnak konkrétá. A következtetési rendszer a nyelv által megfogalmazott formulákkal manipulál, s egy formula univerzálisan igaz voltát vizsgálja. Az, hogy egy nyelv milyen tág világról tud beszélni, attól függ, hogy mennyire használunk benne változókat. Ha egyáltalán nem, akkor nulladrendű nyelvhez jutunk. Elsőrendű nyelvet kapunk, ha megengedünk elemváltozókat, másodrendűt, ha függvények és predikátumok is szerepelhetnek változóként.

A következőkben tekintsük át az elsőrendű logika felépítését!

Alapfogalmak

M.1. Definíció: Az elsőrendű L nyelv szimbólumai:

- változók: x_1, x_2, \dots
- konstansok: c_1, c_2, \dots
- n változós függvényeszimbólumok ($n = 0, 1, 2, \dots$)
- n változós predikátumeszimbólumok ($n = 1, 2, \dots$)
- logikai összekötőjelek: $\wedge, \vee, \neg, \rightarrow$
- kvantorok: \forall, \exists
- zárójelek: $(,)$

M.2. Definíció: Termnek nevezünk

- egy változót,
- egy konstansot,
- egy $f(t_1, \dots, t_n)$ alakú kifejezést, ahol f n -változós függvényszimbólum és t_1, \dots, t_n termek.

M.3. Definíció: Ha p egy n -változós predikátumszimbólum és t_1, \dots, t_n termek, akkor $p(t_1, \dots, t_n)$ atomi formula vagy röviden: atom.

M.4. Definíció: Jól formált formula:

- egy atom,
- ha φ és ψ formulák, akkor $(\neg\varphi)$, $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \rightarrow \psi)$,
- ha φ formula és x változó, akkor $(\forall x\varphi)$ és $(\exists x\varphi)$.

M.5. Definíció: Egy szimbólumhalmazhoz tartozó elsőrendű nyelv a szimbólumokból készíthető összes jól formált formula.

M.6. Definíció: A $\forall x$ (illetve $\exists x$) hatásköre a $(\forall x\varphi)$ (ill. $(\exists x\varphi)$) formulában a φ formula. $(\forall x\varphi)$, $(\exists x\varphi)$ esetén x kötött változó φ -ben. Zárt formulának nevezzük az olyan formulát, amelynek minden változója kötött. A nem kötött változó a szabad változó.

Megjegyzés: A továbbiakban csak zárt formulákkal foglalkozunk.

M.7. Definíció: Egy elsőrendű formula interpretációja egy nem üres D halmaz (univerzum) és a rajta értelmezett következő hozzárendelés:

- A formulában szereplő összes konstanshoz hozzárendelünk egy-egy D -beli elemet.
- A formula minden n -változós függvényszimbólumához hozzárendelünk egy $D^n \rightarrow D$ leképezést ($D^n = D \times \dots \times D$).
- A formula minden n -változós predikátumszimbólumához hozzárendelünk egy $D^n \rightarrow \{\text{igaz, hamis}\}$ leképezést.

M.8. Definíció: Egy φ formula kielégíthető, ha van olyan I interpretációja, amelyben φ igaz. I -t a φ modelljének nevezzük. Formulák egy Φ halmaza kielégíthető, ha van olyan I interpretáció, amelyben minden $\varphi \in \Phi$ igaz. I -t a Φ modelljének nevezzük. Jelölése: $I \models \Phi$.

M.9. Definíció: Ha egy Φ formulahalmazt kielégítő összes interpretáció kielégíti a ψ formulát, akkor azt mondjuk, hogy ψ a Φ logikai következménye. Jelölése: $\Phi \models \psi$.

M.10. Definíció: Ha a φ és ψ formulákra $\varphi \models \psi$ és $\psi \models \varphi$, akkor a két formula logikailag ekvivalens.

M.11. Definíció: Egy φ formula (Φ formulahalmaz) kielégíthetetlen (inkonzisztens), ha nincs olyan interpretáció, amely kielégítené φ -t (Φ -t).

Formulák klóz alakja

A továbbiakban speciális alakú formulák vizsgálatára szorítkozunk.

M.12. Definíció: Literálnak nevezünk egy atomi formulát vagy annak negáltját. Literálok diszjunkciója a klóz.

Az atomi formulát szokás pozitív literálnak is nevezni, míg az atom negáltját negatív literálnak.

M.13. Definíció: A $Q_1x_1 \dots Q_kx_k \varphi$ formulát prenex formulának hívjuk, ha Q_i a \forall vagy a \exists kvantor, és φ -ben már nincs kvantor. Ha az x_i - k különbözőek, és mindegyik szerepel φ -ben, akkor prenex normálformáról beszélünk.

M.14. Definíció: Az olyan prenex normálformát, amelyben minden Q_i univerzális kvantor és φ klózik konjunkciója, Skolem standard formának vagy klózformának nevezzük.

Mint többek között [CL], [M], [P1], [P2] is bizonyítja, van olyan algoritmus, amely segítségével bármely zárt formula átalakítható Skolem standard formává. Ez az átalakítás azért fontos, mert a következő tétel alapján egy formula kielégíthetősége helyett elég a Skolem standard formájának kielégíthetetlenségét bizonyítani.

M.1. Tétel: Legyen φ formula és φ' a φ Skolem standard formája. A φ akkor és csak akkor kielégíthetetlen, ha φ' kielégíthetetlen.

A logikai programozás elméletében kitüntetett szerepet játszik egy speciális klóz, az úgynevezett Horn-klóz.

M.15. Definíció: Horn-klóznak nevezzük az olyan klózt, amelyben legfeljebb egy nem negált atom szerepel. Horn-formula az olyan Skolem standard forma, amelyben szereplő klózek Horn-klózek.

Megjegyzés: Az $A_1 \vee \dots \vee A_k \vee \neg B_1 \vee \dots \vee \neg B_n$ klóz átírható $B_1 \wedge \dots \wedge B_n \rightarrow A_1 \vee \dots \vee A_k$ alakba. Így speciálisan egy Horn-klóz $A \vee \neg B_1 \vee \dots \vee \neg B_n \equiv B_1 \wedge \dots \wedge B_n \rightarrow A$ alakú.

Herbrand tétel

Egy formula definíció szerint akkor inkonzisztens, ha egyetlen interpretációban sem elégíthető ki. Ez azt jelenti, hogy az inkonzisztencia eldöntéséhez az összes interpretációt meg kellene vizsgálni, ami gyakorlatilag lehetetlen. Erre a problémára adott megoldást 1930-ban Herbrand, melynek lényege, hogy lecsökkenti a modellek osztályát egy speciális alaphalmazra, a Herbrand univerzumon vett modellekre.

M.16. Definíció: Egy kifejezést alapkifejezésnek nevezünk, ha nem szerepel benne változó. Ilyen módon használjuk az alapterm, alapatom, alapliterál és alapklóz elnevezéseket.

M.17. Definíció: Egy Φ formulahalmaz H_Φ Herbrand univerzuma a Φ -ben előforduló konstansokból és függvényszimbólumokból képzett alaptermek halmaza. (Ha Φ -ben nincs konstans, akkor tetszőlegesen fölveszünk egyet.)

M.18. Definíció: Egy Φ formulahalmaz B_Φ Herbrand bázisa a Φ -ben előforduló predikátumszimbólumokból és a H_Φ -ben szereplő alaptermekből képzett alapatomok halmaza.

M.19. Definíció: Herbrand interpretáció a Herbrand univerzumon vett olyan interpretáció, amely a konstansoknak önmagukat felelteti meg, egy n -változós f függvényszimbólumnak pedig olyan leképezést, amely a $t_1, \dots, t_n \in H$ termekhez az $f(t_1, \dots, t_n) \in H$ termet rendeli.

Legyen $B = \{A_1, A_2, \dots, A_n, \dots\}$ a formulahalmaz Herbrand bázisa. Egy I Herbrand interpretációt kényelmes módon egy $I = \{m_1, m_2, \dots, m_n, \dots\}$ halmazzal reprezentálhatunk, ahol m_i vagy A_i vagy $\neg A_i$ ($i = 1, 2, \dots$). Ez azt jelenti, hogy ha $m_i = A_i$, akkor A_i - t igaznak tekintjük, ellenkező esetben hamisnak.

M.20. Definíció: A Φ formula Herbrand modelljén Φ egy olyan Herbrand interpretációját értjük, amely a formula modellje.

Többek között [CL] és [L] bizonyítja a következő fontos állítást:

M.2. Tétel: (Herbrand) Egy Φ klózforma akkor és csak akkor kielégíthetetlen, ha hamis a hozzá tartozó összes Herbrand interpretáción.

Hálóelméleti fixponttétel

Az elsőrendű logika formalizálásával nem kezelhető a rekurzió. Ez teszi szükségessé, hogy hálóelméleti tárgyalásmódot hívjunk segítségül.

M.21. Definíció: Legyen H egy halmaz. A H halmazon értelmezett parciális rendezés egy olyan R reláció, amely eleget tesz a következőknek:

- reflexív : $\forall x \in H : xRx$
- antiszimmetrikus : $\forall x, y \in H : ((xRy) \wedge (yRx)) \rightarrow x = y$
- tranzitív : $\forall x, y, z \in H : ((xRy) \wedge (yRz)) \rightarrow xRz$

M.1. Példa: Legyen H egy halmaz és 2^H a H összes részhalmazának halmaza. Ekkor a tartalmazási reláció parciális rendezést értelmez 2^H -n.

Megjegyzés: Az egyszerűség kedvéért a továbbiakban R helyett „ \leq ”-t írok.

M.22. Definíció: Legyen a H halmazon értelmezve a „ \leq ” parciális rendezés. Az $X \subseteq H$ felső korlátja az az $a \in H$, melyre $\forall x \in X$ esetén $x \leq a$. Az X alsó korlátja az az $b \in H$, melyre $\forall x \in X$ esetén $b \leq x$. Ha az $a \in H$ az X felső korlátja, és bármely más a' felső korlát esetén $a \leq a'$, akkor „ a ” az X legkisebb felső korlátja. Hasonlóan értelmezhető a legnagyobb alsó korlát is. Jelölés: $\sup(X)$, illetve $\inf(X)$.

M.23. Definíció: Egy parciálisan rendezett H halmaz háló, ha minden $a, b \in H$ esetén $\inf(\{a, b\})$ is és $\sup(\{a, b\})$ is létezik. H teljes háló, ha a H minden X részhalmazára $\inf(X)$ is és $\sup(X)$ is létezik. A H teljes háló legnagyobb elemét ($\sup(H)$ -t) τ -val, legkisebb elemét ($\inf(H)$ -t) α -val jelöljük.

M.2. Példa: Az előző példában 2^H teljes háló, melynek legkisebb eleme az üres halmaz, legnagyobb eleme H .

A továbbiakban a teljes hálókön értelmezett transzformációk tulajdonságait vizsgáljuk.

M.24. Definíció: A H teljes hálón értelmezett $T: H \rightarrow H$ leképezés monoton, ha minden $x, y \in H$, $x \leq y$ esetén $T(x) \leq T(y)$.

M.25. Definíció: A H teljes hálón értelmezett $T: H \rightarrow H$ leképezés folytonos, ha $T(\sup(X)) = \sup(T(X))$ teljesül H minden direkt részalmazán. (Egy halmaz direkt, ha minden véges részalmazának van felső korlátja.)

M.26. Definíció: Legyen H véges háló és $T: H \rightarrow H$ leképezés. Az $a \in H$ a T legkisebb fixpontja, ha $T(a) = a$, és minden olyan b -re, melyre $T(b) = b$, teljesül, hogy $a \leq b$. Hasonlóan értelmezhetjük a legnagyobb fixpontot.

[CGT] illetve [L] bizonyítja a deduktív adatbázisok elméletében fontos szerepet játszó tételt:

M.3. Tétel: A H teljes hálón értelmezett $T: H \rightarrow H$ monoton leképezésnek van legkisebb fixpontja, $\text{lfp}(T)$ és legnagyobb fixpontja, $\text{lfp}(T)$.

A T transzformáció fixpontjainak jellemzéséhez definiáljuk T hatványait:

M.27. Definíció: Legyen a H teljes hálón értelmezett $T: H \rightarrow H$ leképezés monoton. Ekkor T hatványai:

$$\begin{aligned} T \uparrow 0 &= \alpha \\ T \uparrow (i+1) &= T(T \uparrow i) \\ &\dots \\ T \uparrow \omega &= \sup(\{T \uparrow i \mid i < \omega\}); \\ \\ T \downarrow 0 &= \tau \\ T \downarrow (i+1) &= T(T \downarrow i) \\ &\dots \\ T \downarrow \omega &= \sup(\{T \downarrow i \mid i < \omega\}), \end{aligned}$$

ahol ω az első végtelen rendtípus, azaz a természetes számok nagyság szerint rendezett halmazának rendtípusa.

[L] bizonyítja a következő fontos eredményt:

M.4. Tétel: Legyen a H teljes hálón értelmezett $T: H \rightarrow H$ leképezés folytonos. Ekkor $\text{lcf}(T) = T \uparrow \omega$.

[L] arra is mutat példát, hogy a legnagyobb fixpontra nem teljesül a szimmetrikus állítás, azaz $\text{Inf}(T) \neq T \downarrow \omega$.

Fuzzy alapok

A filozófusok jó ideje egyetértenek abban, hogy a túlzott egzakttság sokszor mesterséges és erőltetett. Mint Bernard Russell írja 1923-ban „Vagueness” című cikkében (Australian J. Phil. 1. 84-92): „... minden hagyományos logika feltételezi a precíz szimbólumok használatát. Emiatt nem alkalmazhatóak erre a földi életre, csak egy képzeletbeli mennyei létre.”

A mindennapi élet fogalmainak, szituációinak leírására nagyon hatékony eszköz a természetes emberi nyelv. Azt tapasztaljuk, hogy a beszélt nyelv fogalmai sok bizonytalanságot tartalmaznak. Ha például azt mondjuk valakire, hogy okos, vagy hogy fiatal, magas, hogy nem lakik túl messzire innen, akkor nem határozzuk meg teljes precízséggel, hogy mit is jelent az okosság, magasság, mégis van elképzelésünk róla. Mit jelent az, hogy valaki fiatal? Ha egy osztályba vagy halmazba szeretnénk sorolni a fiatalokat, hamarosan komoly probléma elé kerülnénk. Ennek a halmaznak a határai elmosódottak. Nem minden emberről tudjuk egyértelműen eldönteni, hogy beletartozik-e ebbe a halmazba vagy sem. Ha a hagyományos értelemben vett halmazfogalommal akarunk dolgozni, akkor a „határon” lévőket önkényesen vagy bele vesszük a halmazba vagy sem. Épp ez az oka annak, hogy a precíz matematikai fogalmak nem mindig alkalmazhatóak a gyakorlatban. Ugyanakkor azonban sok, a gyakorlatban is használt összefüggés precíz leírásához matematikai fogalmakra van szükség. Ezt az ellentétet próbálja áthidalni a Lofti Zadeh által 1965-ben bevezetett fuzzy-elmélet [Z].

A fuzzy halmaz fogalma

Kézenfekvő megoldásnak látszik, hogy precíz igen-nem válasz helyett „elkent” válaszokat adjunk, vagyis minden egyes elemhez hozzárendeljünk egy mérőszámot, amely a halmazba tartozás mértékét jelöli. Minél kisebb ez az érték,

annál „kevésbé” tartozik bele az illető elem a halmazba. Célszerű a mérőszámok skáláját a $[0,1]$ intervallumnak választani, vagyis a legesélytelenebb értéket a 0-val, a legesélyesebbet az 1-gyel jelölni. Ha minden egyes elemhez egy $[0,1]$ közötti értéket rendelünk, akkor egy *fuzzy* (bolyhos) *halmazt* kapunk. A fuzzy halmaz gyakorlatilag a klasszikus halmaz karakterisztikus függvényének általánosítása. Precízebben:

M.28. Definíció: Legyen D egy halmaz. A D feletti fuzzy halmaz egy $F: D \rightarrow [0,1]$ függvény.

Az F függvényt szokták beletartozási függvénynek is nevezni, az $F(d) \in [0,1]$ értéket pedig a $d \in D$ elem halmazba tartozási szintjének vagy fokának. Ha $F(d) = 0$, akkor d nem tartozik F -hez.

Ha egy F fuzzy halmaz csak a véges sok d_1, \dots, d_n elemen vesz föl 0-tól különböző értéket, akkor az F fuzzy halmazra a $\{(d_1, F(d_1)), \dots, (d_n, F(d_n))\}$ halmazjelölést is alkalmazzuk.

Ha F végtelen sok elemen válik 0-tól különbözővé, akkor a $\{(d, F(d)) \mid d \in D\}$ jelölést használjuk. Ugyancsak szokásos az

$$F = \bigcup_{d \in D} (d, \alpha_d)$$

jelölés is, ahol $(d, \alpha_d) \in D \times [0,1]$.

A D összes fuzzy halmazából álló halmazt $F(D)$ - vel jelöljük.

M.3. Példa:

a/ A 10-hez közeli egészek jellemezhetőek pl. a

$$\{(7, 0.1), (8, 0.5), (9, 0.8), (10, 1), (11, 0.8), (12, 0.5), (13, 0.1)\}$$

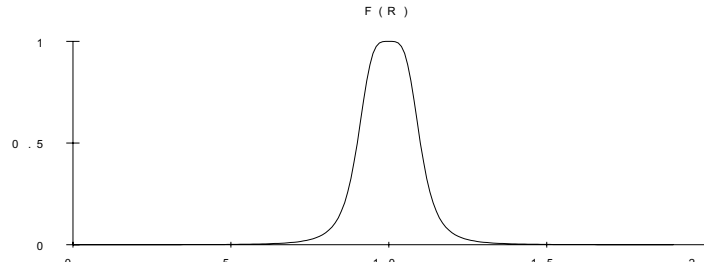
halmazzal.

b/ A 10-hez közeli valós számok pedig például az

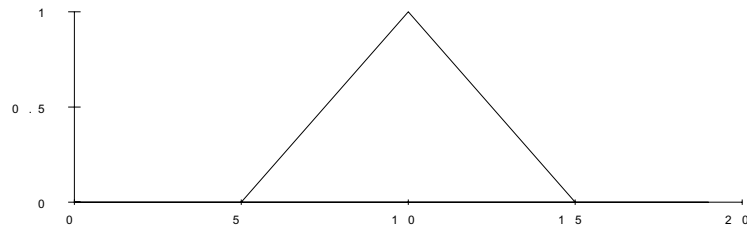
$$\{(x, F(x)) \mid x \in \mathbb{R}\}, F(x) = [1+(x-10)^4]^{-1}$$

halmazzal.

Grafikonon ábrázolva:

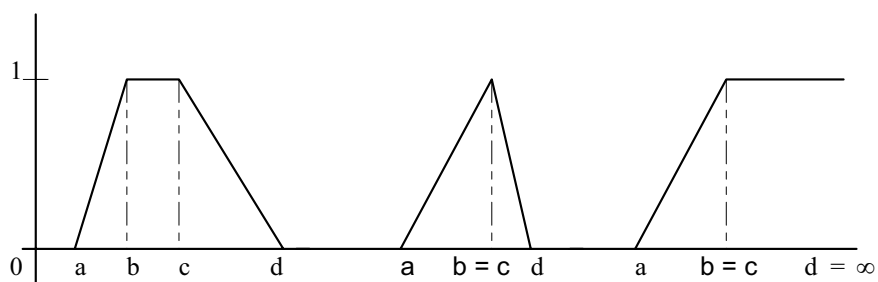


c/ A 10-hez közeli valós számok a



alakú fuzzy halmazzal is jellemezhetőek.

Megjegyzés: Ez utóbbi alakú halmaz a könnyű kezelhetőség miatt előnyt élvezhet, ha a 10-hez közeli számokat valamely adatbázisban szeretnénk felhasználni. Általában a fuzzy adatmodellek megalkotásakor tárolási szempontok és az aritmetikai, összehasonlítási műveletek egyszerűbb kiértékelhetősége miatt a lehetséges fuzzy halmazok körét leszűkítik, és csak a véges sok paraméterrel megadható speciális formájú fuzzy halmazokat használják. Folytonos tartományon csak trapéz alakú fuzzy halmazokat engednek meg, melyek speciális esetként tartalmazzák a háromszög és az S alakú fuzzy halmazokat is. A trapéz alakú fuzzy halmazokat négy számmal, a csúcspontok helyét jelző a , b , c , d értékekkel reprezentálják. Ez a halmazosztály elég bő ahhoz, hogy a gyakorlatban előforduló, folytonos halmazon adott bizonytalanságot modellezni lehessen. Ilyen megszorításokkal dolgozott ki fuzzy adatmodelleket pl. [LL], és hasonló adatmodelleket vizsgálnak a granadai egyetem informatika tanszékén ([MPV], [PVM], [VCM], [VCMP], [VMPC]). A szóban forgó halmazokat a következő ábra szemlélteti:



A fuzzy halmazokkal kapcsolatban bevezetünk még néhány fogalmat:

M.29. Definíció: Legyen F a D egy fuzzy halmaza.

a/ Az F tartója a hagyományos $\text{Supp}(F) = \{d \mid F(d) \neq 0\}$ halmaz.

b/ Adott $\lambda \in [0,1]$ esetén az F λ -levágása a közönséges $F_\lambda = \{d \mid \lambda \leq F(d)\}$ halmaz.

c/ Az F λ -szintje: $F^\lambda = \{d \mid \lambda = F(d)\}$

d/ Az F magja: $\text{Ker}(F) = \{d \mid F(d) = 1\}$

Azt mondjuk, hogy az F fuzzy halmaz normális, ha $\text{Ker}(F) \neq \emptyset$. Ellenkező esetben szubnormális. A fuzzy halmazok kezelésében szokás, hogy elhagyjuk az $F = \cup(d, \alpha_d)$ halmazból azokat a (d, α_d) párokat, melyekre $\alpha_d = 0$, illetve az ellenkezője, hogy kibővítjük a $\text{Supp}(F)$ halmazt azokkal a $(d, 0)$ párokkal, melyekre $d \in D$, de $d \notin \text{Supp}(F)$. [N1] egyszerű számítással bizonyítja, hogy a nagyobb szinten történő vágás részhalmaza az alacsonyabb értékű levágásnak, azaz bármely $\alpha, \beta \in [0,1]$, $\alpha \leq \beta$ esetén $F_\beta \subseteq F_\alpha$.

A halmazelméleti műveleteket többféle módon is általánosítják a fuzzy halmazokra. Általában a t-norma és t-conorma felel meg a metszetnek, illetve az uniónak, de gyakorlati szempontból rendszerint elég ezek speciális eseteként a minimum és a maximum operátort alkalmazni, vagyis :

$$F \cup G(d) \stackrel{\text{def}}{=} \max(F(d), G(d))$$

$$F \cap G(d) \stackrel{\text{def}}{=} \min(F(d), G(d))$$

Egy $F(\cdot)$ fuzzy halmaz komplementerének tekintjük az $1 - F(\cdot)$ fuzzy halmazt. Az üres fuzzy halmaznak az azonosan 0 függvényt, az alaphalmaznak az azonosan 1 függvényt feleltetjük meg.

Az így bevezetett műveletekre a legtöbb halmazelméleti azonosság érvényben marad, de például egy fuzzy halmaz és komplementerének metszete nem feltétlenül az üres halmaz, és uniójuk sem adja ki az egész alaphalmazt.

Egy D halmaz fuzzy halmazai között értelmezhető egy parciális rendezés a következő módon: $F \leq G \Leftrightarrow F(d) \leq G(d), \forall d \in D$.

Belátható, hogy $(\mathcal{F}(D), \leq)$ teljes háló, hiszen bármely $X \in \mathcal{F}(D)$ esetén

$$\inf(X) = \bigcap_{F \in X} F \quad \text{és} \quad \sup(X) = \bigcup_{F \in X} F$$

is létezik.

A háló legnagyobb eleme az $E: D \rightarrow [0,1], E(d)=1, \forall d \in D$. A legkisebb elem a $\emptyset: D \rightarrow [0,1], \emptyset(d)=0, \forall d \in D$.

A fuzzy halmaz fogalmához vezető általánosítás folytatható részben úgy, hogy a $[0,1]$ egységintervallumot egy hálóval helyettesítjük (pl. [N1]), részben pedig a következő módon: megengedjük, hogy a beletartozási függvény értéke maga is fuzzy halmaz legyen. Az ilyen fuzzy halmazt második típusú fuzzy halmaznak (vagy ultrafuzzy halmaznak) nevezzük.

M.4. Példa: Legyen $D = \{\text{János, Péter, Zsófi}\}$. Ekkor az „okosság”-ot jellemző első típusú fuzzy halmaz lehet pl. az

$$\text{okos} = \{(\text{János}, 0.9), (\text{Péter}, 0.3), (\text{Zsófi}, 0.6)\},$$

míg ugyanezt a fogalmat második típusú fuzzy halmazzal például a következő módon adhatnánk meg:

$$\text{okos} = \{(\text{János}, \text{nagyon}), (\text{Péter}, \text{alig}), (\text{Zsófi}, \text{közepesen})\},$$

ahol az „alig”, „közepesen”, illetve „nagyon” maguk is fuzzy halmazok, melyeket pl. a következő módon értelmezhetünk:

$$\text{alig} = \{(0.1, 0.8), (0.2, 1), (0.3, 0.8), (0.4, 0.6)\}$$

$$\text{közepesen} = \{(0.4, 0.7), (0.5, 0.8), (0.6, 1), (0.7, 0.8), (0.8, 0.7)\}$$

$$\text{nagyon} = \{(0.7, 0.5), (0.8, 0.7), (0.9, 0.9), (1, 1)\}.$$

A második és magasabb típusú fuzzy halmazok fogalmát precízen definiálja [N1]. Gyakorlati szempontból azonban a legfontosabbak az első és második típusú fuzzy halmazok. A továbbiakban csak az első típusú fuzzy halmazokról lesz szó.

A fuzzy logika alapelemei

Fuzzy logika terminológia alatt a nem klasszikus logika két ágát is értjük, mindkettő a klasszikus kétértékű logika általánosítása. Ez a két ág a következő:

a/ Többértékű logika, vagyis az a logika, amelyben az igazságértékek egy adott halmaz – rendszerint a $[0,1]$ intervallum – elemei.

b/ Lingvisztikus logika, vagyis az a logika, melynek igazságértékei a természetes beszélt nyelv szavai, pl. igaz, többé-kevésbé igaz, inkább hamis, stb. Ezeket a kifejezéseket a $[0,1]$ intervallum fuzzy halmazával modellezhetjük.

A fuzzy logika mindkét ágának precíz felépítése megtalálható [N1] könyvében, de több más publikáció is foglalkozik ezzel a témával (pl. [G], [Pv], [R]).

Ha bizonytalan információk kezelésére vonatkozó logikai programnyelvet akarunk konstruálni, akkor két irányban indulhatunk el: vagy egy, a klasszikus logikára épülő programnyelvet veszünk alapul, s ebből kiindulva próbálunk általánosítani, vagy eleve egy nem-klasszikus logikát – pl. a fuzzy logikát – vesszük kiindulásul, s erre alapozva építünk fel egy nyelvet. Mivel dolgozatomban az előbbi utat választottam, ezért nem tárgyalom teljes részletességgel a fuzzy logikát, hanem csak egy leszűkített részét, amelyből kiderül, hogy a Datalog általánosított tárgyalta általánosítása nincs ellentmondásban a fuzzy logika eredményeivel.

A fuzzy logika szintaktikája a következőkből áll:

A nyelv szimbólumai:

- változók: x_1, x_2, \dots
- konstansok: c_1, c_2, \dots
- az igazságértékek szimbólumai: $\alpha; \alpha \in [0,1]$
- n változós függvényszimbólumok: f_1, f_2, \dots
- n változós predikátumszimbólumok: p_1, p_2, \dots
- logikai összekötőjelek: $\wedge, \vee, \neg, \rightarrow$
- kvantorok: \forall, \exists
- zárójelek: $(,)$

Termnek nevezünk egy változót, egy konstansot, és egy $f(t_1, \dots, t_n)$ alakú kifejezést, ahol f egy n -változós függvényszimbólum és t_1, \dots, t_n termek.

A nyelv formulái:

- az α igazságértékszimbólum;
- ha p n -változós predikátumszimbólum és t_1, \dots, t_n termek, akkor $p(t_1, \dots, t_n)$;
- ha φ és ψ formulák, akkor $(\neg\varphi)$, $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \rightarrow \psi)$;
- ha φ formula és x változó, akkor $(\forall x\varphi)$ és $(\exists x\varphi)$.

A klasszikus elsőrendű logikához hasonlóan itt is bevezethetjük a szabad és kötött változó fogalmát. Ha t term és φ egy formula, akkor $\varphi_x(t)$ azt a formulát jelöli, amelyet úgy kapunk, hogy φ -ben az x minden szabad előfordulását t -vel helyettesítjük. Egy formula zárt, ha minden változó kötött benne.

Egy formulát a következő módon interpretálhatunk:

Egy elsőrendű formula interpretációja egy nem üres D halmaz (univerzum) és a rajta értelmezett következő hozzárendelés:

- A formulában szereplő összes $d \in D$ konstanshoz hozzárendelünk egy-egy $\alpha \in [0,1]$ fuzzy értéket. Jelölése: (d,α) .
- A formula minden n -változós f függvényszimbólumához hozzárendelünk egy n változós f fuzzy függvényt. (A fuzzy függvény fogalma többféleképpen értelmezhető, de mivel a későbbiekben nem használjuk, ezért itt nem részletezzük.)
- A formula minden n -változós predikátumszimbólumához hozzárendelünk egy $P: D^n \rightarrow [0,1]$ leképezést.

Minden egyes φ formulához hozzárendelhetjük a formula igazságértékét, $v(\varphi)$ -t:

- Ha α igazságértékszimbólum, akkor $v(\alpha) = \alpha$.
- Ha t term, akkor $v(t)$ az interpretációnak megfelelő érték.
- Ha φ és ψ formulák, akkor

$$v(\varphi \wedge \psi) = \min(v(\varphi), v(\psi)),$$

$$v(\varphi \vee \psi) = \max(v(\varphi), v(\psi)),$$

$$v(\neg\varphi) = 1 - v(\varphi),$$

$$v(\varphi \rightarrow \psi) = I(\varphi, \psi), \text{ ahol } I(x, y) \text{ implikációs operátor,}$$

$$v(\forall x\varphi) = \inf_{t \in D} v(\varphi_x(t)).$$

Megjegyzések:

1. Az interpretációban szereplő D tetszőleges halmaz, így speciálisan fuzzy halmaz is lehet.
2. Az általam definiált fuzzy logika erős megszorítása és speciális esete az [N1] által definiált logikának, ahol az igazságértékek egy reziduális háló lehetséges elemei, és az általam említett, a klasszikus logikából ismert összekötőkön és kvantorokon kívül egyéb összekötőket és kvantorokat is értelmeznek.

Egy φ formula α szinten igaz egy adott interpretációban, ha $v(\varphi) = \alpha$. A formula α -tautológia, ha $v(\varphi) = \alpha$ teljesül minden interpretációban. Ez utóbbi jelölése: $\models \varphi$.

[N1] bebizonyítja, hogy $\models \varphi \rightarrow \psi \Leftrightarrow v(\varphi) \leq v(\psi)$

A $\varphi \rightarrow \psi$ formula igazságértékét az implikációs operátor segítségével értelmeztük, de az implikációs operátorok fogalma bővebb kifejtést igényel.

Az implikációs operátorok megválasztásával és tulajdonságaival kapcsolatban sok vizsgálódás folyik. Ezekről a vizsgálatokról ad összefoglaló képet [DP1]. A különböző implikációs operátorokat a normák és conormák segítségével értelmezzük. Ezek a metszet és az unió műveletének a fuzzy halmazokra való kiterjesztése, pontosabban:

t-norma (trianguláris norma) egy $T: [0,1]^2 \rightarrow [0,1]$ kommutatív, asszociatív, mindkét változójában monoton növekvő függvény, melyre

$$T(1, \alpha) = \alpha, \forall \alpha \in [0,1].$$

t-conorma egy $S: [0,1]^2 \rightarrow [0,1]$ kommutatív, asszociatív, mindkét változójában monoton növekvő függvény, melyre

$$S(0, \alpha) = \alpha, \forall \alpha \in [0,1].$$

Három fő implikációs osztályt különböztetünk meg:

1. S - implikációk

Ez a fajta implikáció a klasszikus implikáció $\varphi \rightarrow \psi = \neg\varphi \vee \psi$ tulajdonságán alapszik, és $I(\alpha, \beta) = S(n(\alpha), \beta)$ alakú, ahol S a többértékű diszjunkció kifejezésére szolgáló t-conorma, és n erős negáció, vagyis n szigorúan csökkenő, folytonos $[0,1] \rightarrow [0,1]$ függvény, melyre $n(0) = 1$ és $n(1) = 0$.

2. R-implikációk

Ez a klasszikus implikáció azon elképzelésén alapul, miszerint az implikáció parciális rendezést indukál, vagyis $I(\alpha, \beta) = 1 \Leftrightarrow \alpha \leq \beta$. Ez a fajta implikáció a reziduális hálók tulajdonságain alapszik – innen is származik elnevezése. Az R-implikációk általános alakja: $I(\alpha, \beta) = \sup\{\gamma \in [0, 1] \mid T(\alpha, \gamma) \leq \beta\}$, ahol T trianguláris norma.

3. QL implikációk

Ez a kvantumlogikában használatos implikációs forma: $I(\alpha, \beta) = S(n(\alpha), T(\alpha, \beta))$ alakú, ahol S egy t-conorma, n egy erős negáció, és T az S n-duálisa, azaz: $T(\alpha, \beta) = n(S(n(\alpha), \beta))$.

M.1. táblázat: A következő táblázatban összefoglaljuk a leggyakoribb implikációs operátorokat:

jelölés	név	formula	implikáció-típus
I ₁	Gödel	1 ha $\alpha \leq \beta$ β egyébként	R (T = $\min(\alpha, \beta)$)
I ₂	Lukasiewicz	1 ha $\alpha \leq \beta$ $1 - \alpha + \beta$ egyébként	S (S = $\min(1, \alpha + \beta)$) R (T = $\max(0, \alpha + \beta - 1)$)
I ₃	Goguen	1 ha $\alpha \leq \beta$ β/α egyébként	R (T = $\alpha \cdot \beta$)
I ₄	Kleene-Dienes	$\max(1 - \alpha, \beta)$	S (S = $\max(\alpha, \beta)$) QL (S = $\min(1, \alpha + \beta)$)
I ₅	Reichenbach	$1 - \alpha + \alpha\beta$	S (S = $\alpha + \beta - \alpha\beta$)
I ₆	Zadeh	$\max(1 - \alpha, \min(\alpha, \beta))$	QL (S = $\max(\alpha, \beta)$)
I ₇	Gaines-Rescher	1 ha $\alpha \leq \beta$ 0 egyébként	csak formálisan R

Más tárgyalásmód szerint az implikációs operátoroktól megkövetelik, hogy eleget tegyenek egy axiómarendszer feltételeinek. A legelfogadottabb feltételek a következők:

1. Ha $\alpha \leq \alpha'$ akkor $I(\alpha, \beta) \geq I(\alpha', \beta)$ (első változójában monoton csökkenő)
2. Ha $\beta \geq \beta'$ akkor $I(\alpha, \beta) \geq I(\alpha, \beta')$ (második változójában monoton növekvő)
3. $I(0, \beta) = 1$ (a hamis bármit implikál)
4. $I(1, \beta) = \beta$ (a tautológia semmit sem indokol)
5. $I(\alpha, \beta) \geq \beta$ ($\psi \rightarrow (\varphi \rightarrow \psi)$ numerikus megfelelője)
6. $I(\alpha, \alpha) = 1$ (az identitás elve)
7. $I(\alpha, I(\beta, \gamma)) = I(\beta, I(\alpha, \gamma))$ (felcserélhetőségi elv)
8. $I(\alpha, \beta) = 1 \Leftrightarrow \alpha \leq \beta$ (az implikáció egy rendezést definiál)
9. $I(\alpha, \beta) = I(n(\beta), n(\alpha))$ valamely erős negációra (kontrapozíció elve)
10. I folytonos

A következő táblázat megmutatja, hogy az előzőekben értelmezett implikációs operátorok mely tulajdonságoknak tesznek eleget.

jelölés	tulajdonságok
I_1	1-8
I_2	1-10
I_3	1-8, 10
I_4	1-5, 7, 9-10
I_5	1-5, 7, 9-10
I_6	2, 3, 4, 10
I_7	1-3, 6-9

A táblázatokból is látható, hogy a Lukasiewicz implikáció az egyetlen olyan, amely mind a tíz feltételnek eleget tesz, és ez az egyetlen olyan, amely egyszerre R és S implikáció is.

Általában az S implikáció megsérti az identitási feltételt (6.) és nem definiál rendezést (8.), az R implikáció viszont gyakran nem tesz eleget a kontrapozíciós elvnek (9.).

BIZONYTALANSÁGKEZELÉSI MODELLEK

Értekezés a doktori (Ph.D.) fokozat megszerzése érdekében informatika tudományágban.

Írta: Achs Ágnes okleveles matematikus

Készült a Debreceni Egyetem IK Matematika- és Számítástudományok doktori iskolája (Informatika programja) keretében

Témavezető: Dr. Fazekas Gábor

A doktori szigorlati bizottság:

elnök: Dr.
tagok: Dr.
Dr.

A doktori szigorlat időpontja: 200... ..

Az értekezés bírálói:

Dr.
Dr.
Dr.

A bírálóbizottság:

elnök: Dr.
tagok: Dr.
Dr.
Dr.
Dr.

Az értekezés védésének időpontja: 200... ..