Lexikalizmus és számítástudomány

Balogh Kata
általános nyelvészet – magyar

Témavezető:
Alberti Gábor
egyetemi tanár

Tartalom

Bevezetés 1
1. Fejezet: Számítógépes nyelvészet 1
1.1 Formális nyelvleírás 1
1.2 Alkalmazások 3
2. Fejezet: Nem transzformációs generatív nyelvtanok 5
2.1 Kategoriális nyelvtanok (CCG, CUG) 6
2.2 LFG 9
2.3 HPSG 13
3. Fejezet: Totális lexikalizmus – GASG 14
4. Fejezet: A totális lexikalizmus és számítógépes implementációja 19
4.1 A Prolog nyelv 19
4.2 DRT – alapok röviden 22
4.3 Implementáció szavak szintjén 23
4.4 Implementáció morfémák szintjén 30
4.4.1 Morfofonológia 30
4.4.2 Szintaxis 33
4.4.3 Szemantika 36
4.4.4 Példák programfutásokra 38
Összegzés 41
Irodalom 43
Melléklet 45

Bevezetés

Dolgozatomban általános nyelvészet szakos hallgatóként a számítógépes nyelvészet szakterületén végzett kutatásról, egy totálisan lexikalista nyelvelmélet implementációs lehetőségeiről számolok be. A kutatási eredmények bemutatása előtt szó lesz a tudományos háttérről, magáról a lexikalista nyelvtanról valamint a lexikalizmusról mint filozófiai háttérről.

A kutatás a Ge(neratív)Lex(ikon)-projekt keretei közt zajlik, melynek feladata a GASG (Generatív Argumentum Struktúra Nyelvtan) elképzelései alapján egy Prolog nyelven írt program kidolgozása, ami egy adott nyelv (jelenleg a magyar) nyelvtanát reprezentálja és egy bemeneti szósorról egyértelműen eldönti, hogy grammatikus vagy agrammatikus, végül a grammatikusnak elfogadott mondatokhoz morfoszintaktikai és diskurzus-szemantikai elemzést tud társítani. A kutatás legvégső célja – több nyelv modellálásával – a számítógépes fordítás, amely a jelenlegi elképzelés szerint a jelentés-ábrázoláson keresztül valósítható meg.

1. Számítógépes nyelvészet

1.1 Formális nyelvleírás

A természetes nyelvek számítógépes feldolgozásának ötlete a gépi fordítás lehetőségével[1] van szoros kapcsolatban. Mivel a számítógép jelrendszerek elemzésére képes, az alapötlet az volt, hogy a természetes nyelvi jelek rendszerének elemzésére is képesnek kell lenni. Ehhez azonban szükség van a természetes nyelvek formális eszközökkel való leírására (Partee – Meulen – Wall 1990). Így megadva egy adott természetes nyelv lexikonát és grammatikáját (nyelvtani szabályok összességét), a számítógép képes az adott nyelvet értelmezni és használni.

A generatív nyelvelméletek és a formális nyelvleírás elindítója Noam Chomsky volt, aki 1957-es könyvében megfogalmazta a generatív grammatika alapjait, miszerint minden természetes nyelv egy mondatokból álló végtelen halmaz, amit véges szabályokkal tudunk meghatározni. Ez a véges szabályrendszer az adott nyelv grammatikája, ami Chomskynál a nyelvi kompetencia fogalmával azonos.

Egy nyelv tehát – formális meghatározás szerint – egy adott szimbólumhalmazból (a nyelv szavai) meghatározott szabályok alapján (a nyelv grammatikája) összeállítható füzérek halmaza. Minden olyan mondat grammatikus egy adott nyelven, amely a megadott szabályok alapján a szimbólumhalmazból

[1] A lehetőséget amerikai tudosók vetették fel az 1950-es években.

levezethető. Ez a halmaz azonban nem csak egy szabályrendszer alapján vezethető le. A formális nyelvleírás célja olyan nyelvtanok megadása, amit a számítógép (vagy a felhasználó) kezelni tud, tehát arra kell törekedni, hogy minél egyszerűbb szabályrendszerrel generáljunk egy nyelvet. Ehhez bizonyos megszorításokat kell alkalmaznunk a formális nyelvek szabályrendszereire. Ezeket a megszorításokat a Chomsky-féle nyelvtanhierarchia tartalmazza, amely négy csoportba sorolja a formális nyelveket:

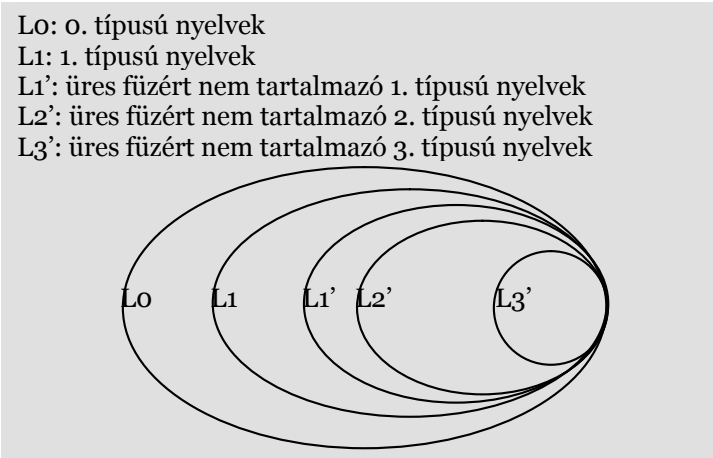
a.) *0. típus*, ahol a nyelvtan definíciójához képest nincs speciális megszorítás.

b.) *1. típus*: a környezetfüggő nyelvtanok, ahol a megszorítás a következő: $\alpha A\beta \rightarrow \alpha\mu\beta$, ahol $\mu \neq e$, vagyis egy konkrét környezet mellett (α, β) egy nem-terminális szimbólumot kicserélhetünk egy tetszőleges füzérre.

c.) *2. típus*: környezetfüggetlen nyelvtan, ahol a megszorítás a következő: $A \rightarrow \mu$, miszerint szintén egy nem-terminális szimbólumot cserélhetünk ki egy tetszőleges füzérre, de nem hivatkozhatunk környezeti feltételre.

d.) *3. típus*: jobblineáris nyelvtanok, ahol a megszorítás: $A \rightarrow xB$ vagy $A \rightarrow x$, ami már a szabály kimenetére is megkötéseket alkalmaz.

Ezek a nyelvtípusok hierarchiát alkotnak, mivel minden típusban van olyan nyelv, ami egy magasabb típusú grammatikával már nem generálható.



1.példa: a nyelvosztályok hierarchiája

A formális nyelvek elméletével a cél a természetes nyelvek rendszerének leírása, megtalálni azt a bonyolultsági fokot, amivel a természetes nyelvek megragadhatóak. A 0. és az 1. típusú formális grammatikák túl általánosnak bizonyultak ahhoz, hogy a természetes nyelveket modellálják. A környezetfüggetlen leírhatóság ellen pedig a svájci német keresztező függőség a példa. Ezek szerint tehát a természetes nyelvek

valahol az 1. és 2. típus között helyezkednek el, a szakirodalomban *enyhén környezetfüggőeknek* nevezik őket. A formális nyelvek közül idetartoznak az indexelt nyelvtanok (Gazdar–Klein–Pullum–Sag 1985), a faépítő nyelvtan (Joshi–Levy–Takahashi 1975) és a kategoriális nyelvtanok (CCG, CUG; Zeevat 1991, Karttunen 1986, König 1996).

A formális nyelvleírás elméleti eredményeire épül a számítógépes nyelvészet[2], amely mint kutatási terület már az 1960-as évek közepétől elfogadott volt. A kutatások elindítója a korábban már említett számítógépes fordító rendszerek megalkotása volt. Hosszú ideig ez jelentette a számítógépes nyelvészet történetét.

A fordító rendszerek eszközeik és technikáik alapján három nagy csoportba sorolhatók:

1. teljesen automatizált gépi fordítás,
2. ember által támogatott gépi fordítás,
3. gép által támogatott emberi fordítás.

Az első típusról már a kutatások elején megállapították, hogy megvalósíthatatlan a nyelvtípusok közötti jelentős különbségek miatt[3]. A két másik lehetőség közül az ember által támogatott gépi fordítás a kedveltebb, mivel a kettő közül ez bizonyul olcsóbb megoldásnak (ugyanis kevesebb emberi munkát igényel) valamint jó minőségű, többnyelvű és gyorsabb. Azonban mindez csak bizonyos megszorításokkal érvényes, hiszen ezt a módszert szinte csak műszaki szakszövegek fordítására használják, valamint az emberi beavatkozás csupán a többértelműségek, bizonytalanságok és fordítási hibák kiszűrésére szolgál.

1.2 Számítógépes nyelvi alkalmazások, gépi fordítás

Az első gépi fordító rendszert 1957-ben mutatták be az amerikai Georgetown Egyetemen. Ez az oroszról angolra fordító minimális rendszer illusztrációként szolgált arra, hogy bizonyítsák a gépi fordítás lehetőségét. Ebben az időben azonban még csak a közvetlen fordítást alkalmazták, ahol egyedül a szótárt és a célnyelvi szórendet tartották relevánsnak. Szemantika egyáltalán nem volt, csak néhány szemantikai jegy mint például a [± ÉLŐ] jegy. Ezek a kísérletek nem voltak

[2] A történetről például: Prószéky (1989), Kiefer (2002), Russel-Norvig (2000).

[3] Például egy angol-orosz fordítórendszer esetében, ahol a gép nem tudja azt kezelni, hogy az oroszban nincsenek névelők, vagyis nem tudja eldönteni azt, hogy oroszról angolra való fordítás esetében mikor kell névelőt használni, és ha kell, akkor éppen melyiket.

eredményesek, és az ALPAC 1966-os negatív jelentése[4] után az egész kutatás hanyatlott.

A közvetlen fordítás mellett azonban megjelentek más elképzelések is. A közvetítőnyelvre épülő rendszer a két nyelv közé átmenetként egy szintaktikai szerkezetet épített. Ez az elképzelés a Chomsky-féle mélyszerkezetre épül, de az elemzési nehézségek miatt nem bizonyult hatékonynak. Ez a tény a transzfer módszer kialakításához vezetett, ahol a két nyelvhez különböző mélyszerkezeti elemzés társul, a fordítás pedig három lépéses: analízis → transzfer (lexikális feltöltés → szerkezeti átalakítás) → szintézis. Az 1970-es években terjedt el a konceptuális függőségi modell, majd az 1980-as években a kutatásokba bekapcsolódtak a szemantika és a mesterséges intelligencia területén végzett kutatások. A korszak egyik legsikeresebb gépi fordítója a SYSTRAN (SYStem TRANslation) rendszer, amely az 1970-es években már a kereskedelmi forgalomban is megjelent.

Napjainkban számos fordító rendszer létezik a egyszerű tükörfordítástól a komoly nyelvtani reprezentációkig. Ezek a rendszerek azonban leginkább csak indoeurópai nyelvek között (főleg nyugat-európai nyelvek) fordítanak, és csak megközelítően jó eredménnyel. Az emberi utómunkálat minden esetben elengedhetetlen.

Ma működő rendszerek közül kiemelendő a SYSTRAN, a CYBERTANS, valamint az igen elterjedt fordítást segítő program, a TRADOS. A SYSTRAN közel negyvenéves fejlesztés során a legelterjedtebb gépi fordító rendszerré alakult. Az alkalmazásokat kezdetben az amerikai légierő, a NASA, majd 1975-től a Xerox cég is felhasználta. A rendszer két meghatározó eleme a szótár és a forrásnyelvi elemző. Az évtizedek során nagyon precíz és gazdag szótárakat alkottak, amelyekben szavak és kifejezések (idiómák) egyaránt szerepelnek, számos speciális kifejezést tartalmaznak szakterületek szerint, valamint a felhasználóspecifikus szótárak (CSD) beépítésével a felhasználó saját igényeihez alakíthatja a szótárát. A CYBERTRANS egy angol-magyar fordító, amely a szoftwerfejlesztők szerint angol és magyar nyelvtani elemzést készít: felismeri az angol igeragozást, a rendhagyó alakokat, igeidőket, valamint a magyar toldalékolást, igemódokat, igeidőket.

[4] Az Egyesült Államok Nemzeti Tudományos Akadémiájának jelentése, miszerint a gépi fordítás jövője korlátozott. A jelentés után az Egyesült Államok kormánya befejezte a GF-projekteket és finanszírozásukat.

Magyarországon a legismertebb kutatói és szoftverfejlesztői cég a Prószéky Gábor vezette MorphoLogic. Több nyelvhelyességellenőrző programot[5], szótárprogramokat[6], valamint az ezekhez kapcsolódó szintaktikai és morfológiai elemzőket fejlesztettek ki. A szintaktikai elemzés (Kis 1997) környezetfüggetlen nyelvtanokra épül, és a mondat elemzését frázisstruktúra fákkal reprezentálják.

```
[1] A nagy kutya ugatja a holdat.
[1:1]
    S -> [NN V Sfx NN Acc End]
        NN -> [Det Adj N]
            Det -> a[DET]=A
            Adj -> nagy[MN]
            N -> kutya[FN]
        V -> ugat[IGE]
        Sfx -> +ja[Te3]
        NN -> [Det N]
            Det -> +a[DET]
            N -> +hold[FN]
        Acc -> +at[ACC]
        End -> .
[/1:1]
[/1]
```

2. példa: egy lehetséges szintaktikai kimenet

Az elemző nyelvfüggetlen, minden nyelvtani tudást a formális készletből kap. A nyelvtani szerkezetekről készített reprezentáció egy-egy *metabetű*, amelyek magasabb szinten értelmezhető *metabetűvé* épülnek. Így a rendszer nem közvetlenül a formális nyelvtant használja, hanem a fenti módon megalkotott nyelvtani szerkezetekből felépülő *metaszótárakat*.

A mondatelemzést szóelemzéssel kezdi: meghatározza a szavakban szereplő morfémákat és azok nyelvtani jellemzőit, majd azonosítja a mondatban szereplő főnévi csoportokat, végül az előző két szint információi alapján megállapítja a mondat mivoltát. Amennyiben az elemző a kapott bemenetet nem ítéli mondatnak, a részelemzéseket külön megadja.

2. Nem transzformációs generatív nyelvtanok

A generatív nyelvleírás legelterjedtebb változatai a transzformációs grammatikák, melynek kiindulása Chomsky nevéhez fűződik. Az elmúlt évtizedekben azonban a nyelvleírás mindinkább a lexikon irányába tolódik (a

---

[5] ***Helyesek***: teljes nyelhelyesség ellenőrző, ***Helyes-e?***: helyesírás-ellenőrző és elválasztó, ***Helyesebb***: magyar nyelvhelyesség-ellenőrző, ***Helyette***: szinonimaszótár és ragozó tezaurusz, ***Helyesel***: elválasztó program

[6] MoBiDic, MobiMouse

minimalizmus elvével Chomsky munkásságában is, Chomsky (1995)). A transzformációs nyelvtanok összetevős szerkezeti fákat építenek, a generálás két fő mozzanatból áll: a *bázis* szabályai generálják a kiinduló (vagy mély-) szerkezetet, a *transzformációs* szabályok pedig a felszíni szerkezetet. Az 1990-es évek elejére azonban bebizonyították, hogy ezen nyelvtanok generatív kapacitása gyakorlatilag a megszorítatlan (0. típusú) nyelvtanokéval egyenértékű, míg a természetes nyelvek az enyhén környezetfüggő típusba (1. és 2. között) tartoznak. Az egyik probléma a transzformációs modellekkel tehát az, hogy leíró erejük nem felel meg a természetes nyelvek megragadására. A másik felmerült probléma pedig az volt, hogy ezekben az elméletekben nem (vagy csak igen nehézkesen) lehet megteremteni a kapcsolatot szintaxis és szemantika között. Ezek a gondolatok vezettek a transzformációt elutasító generatív grammatikákhoz, ahol már nem különböztetnek meg mély- és felszini szerkezetet, a mondatgeneráláshoz szükséges információkat a lexikonból nyerik. A legjelentősebb modellek a Kategoriális Grammatikák (CCG, UCG), a Fejvezérelt Frázisstruktúra Nyelvtan (HPSG) és a Lexikai-Funkcionális Nyelvtan (LFG). A következő alfejezetekben néhány példa segítségével ezek alapgondolatait szeretném bemutatni, tartózkodva az elméletek minden részletére kiterjedő bemutatásától.

2.1 Kategoriális nyelvtanok

A lexikalista elképzelést, miszerint a szavak kombinációjának módja a lexikonban van kódolva, elsőként a Klasszikus Kategoriális Grammatika fogalmazta meg, amivel döntő fordulatot hoztak a nyelvleírásban.

A szavak kombinációjának tulajdonságai a szavak lexikai kategóriáiba vannak beépítve. Kétféle kategóriát különböztetnek meg:

1.) *alap* kategória: a címkék elemi szimbólumok,

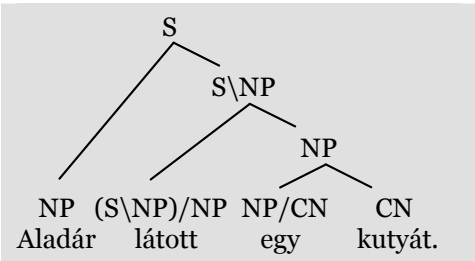
2.) *kiterjesztett* kategória (=funktor): a címkék összetett kifejezések.

```
alapkategóriák:
    CN (common noun = köznév)
    NP (noun phrase = főnévi kifejezés)
    S (Sentence = mondat)
kiterjesztett kategóriák:
    NP/CN (determináns)
    S\NP (intranzitív ige)
    (S\NP)/NP (tranzitív ige)
```

3. példa: CCG kategóriák

A kiterjesztett kategóriákban szerepelnek a szintaktikai szabályok, amelyek szerepe megegyezik a faépítő nyelvtanok újraíró szabályaival. A fenti példában

látszik, hogy ezek a szabályok rendezett hármasok, melynek tagjai: <eredmény, irány, argumentum>. Vagyis egy X/Y címke azt mondja, hogy ha jobbról kap egy Y címkéjű elemet, akkor X kategóriává egyesülnek. Ugyanez X\Y esetében balról érkező Y esetén áll fenn. A szavakhoz – vagyis minden terminális szimbólumhoz – egy függvény rendeli hozzá a kategóriacímkét. A szintaktikai szerkezet megalkotását pedig a függvény-alkalmazás művelete végzi: X/Y Y ⇒ X vagy Y X\Y ⇒ X. Szemléltetésképp nézzünk egy egyszerű példát a fenti eszköztárral:



4. példa: CCG elemzés

A kategoriális nyelvtanok újdonsága a transzformációs nyelvtanokkal szemben, hogy már képesek szintaxis és szemantika együttes kezelésére, a függvény-alkalmazás műveletét használva a szemantikai reprezentációhoz is, ahol X/Y Y ⇒ X szabályt alkalmazza a második kifejezésben szereplő Y típusú dologra, és X kategóriájú összevont kifejezés számára egy X típusú dolgot nyerünk. A generatív kapacitás tekintetében azonban még mindig gyengének bizonyul, nem képes kezelni például a szórendi változatokat, illetve a távolsági kapcsolatokat. Erre az egyik megoldás az volt, hogy a függvény-alkalmazás mellé más összevonási szabályokat is bevezettek:

függvény-kompozíció	X/Y Y/Z ⇒ X/Z
típusemelés	X ⇒ Y/(Y\X)
kommutatív szabály	(X/Y)\Z ⇒ (X\Z)/Y
Geach-szabály	X/Y ⇒ (X/Z)/(Y/Z)

A másik megoldás az unifikáció alapú formalizmus bevezetése. A CG eszköztárának az unifikációval való kibővítése eredményeként született meg a UCG (Unification Categorial Grammar; Zeevat-Klein-Calder 1988, Zeevat 1989) és a CUG (Categorial Unification Grammar; Uszkoreit 1986, Karttunen 1986).

Az UCG formalizmusban a CG eszközeit jegyekkel egészítik ki, amelyek a reprezentáció három szintjét (fonológia, szintaxis, szemantika) tartalmazó összetett kifejezések. Ehhez a CG-ben bemutatott kategóriák halmazát a következőképp kell módosítani:

1. elemi kategóriák a következők: N, NP, S;
2. bármely elemi kategória kategória;
3. ha X egy kategória és E egy jegy, akkor X/E is kategória.

Az összetett kategóriák tartalmazhatnak változókat, ez képezi az aktív részt. Továbbá a változók megjelenhetnek a fonológia és a szemantika szintjén is. A változók bevezetésével oldja meg Zeevat például a keresztező függőséget. A címkékben a kisbetűk jelentik a konstansokat, a nagybetűk pedig a változókat.

(1) Jan Marie de kinderen bier zag laten drinken.

n	n	n	n	s\n\n\(s\n)	X\n\(s\n)\(X\(s\n))	Y\n\(Y\(s\n))
Jani	Mari	a gyerekek	sör	látta	hagyni	inni

„Jani látta, hogy Mari hagyja, hogy a gyerekek sört igyanak."

A példa lényege, hogy elvileg bármilyen hosszúságú infinitívusz-lánc alkotható, ami miatt a *drinken* típusú kifejezéseknek végtelen számú kategóriacímkét kellene adni. Mivel ez nem lehetséges, egy változót vezet be a kiváltására.

A szemantikai reprezentációhoz a Kamp-féle DRT (Discourse Represent Theory, Kamp–Reyle 1993) egy változatát, az InL-t (Indexed Language) használják, melynek két fő tulajdonsága van:

1.) minden kifejezés rendelkezik egy kitüntetett változóval – ez az index,

2.) minden változó osztályozott, így jelöli a változóval jelzett elem kategóriáját.

A három komponens ismeretében az *ugrik* igén nézzük meg, hogy milyenek az UCG lexikai egységei.

fonológiai komponens:	w ugrik
szintaktikai komponens:	S[fin]\w:NP[3, Sg, Subj]:x
szemantikai komponens:	[p][PRES(p), jump(p, x)]

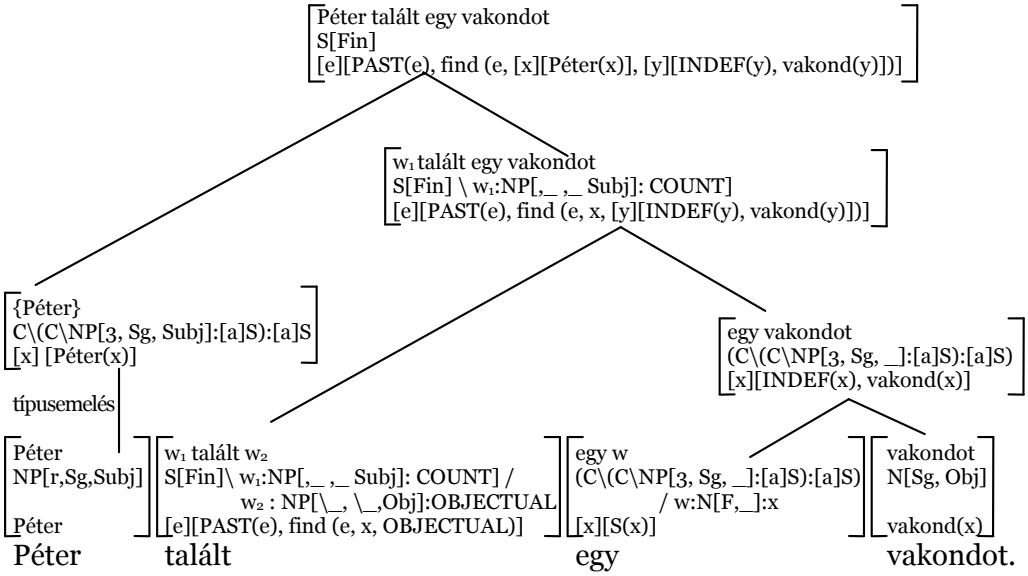
4. példa: UCG lexikai egység

A szintaktikai szinten látjuk a konstansokkal és változókkal megfogalmazott S\w:NP:x szabályt, ahol a jobb oldalon a várt alany három komponense szerepel. A 4. példában szereplő, szintaktikai jegyekkel ellátott szabály tehát azt mondja, hogy az *ugrik* ige szintaktikailag olyan, hogy ha balról kap egy NP-t, ami harmadik személyű, egyes számú és alany esetben áll, akkor ez a két elem egy mondatot alkot. A várt alanyról továbbá rögzíti, hogy van egy *w* változóval jelölt fonológiai komponense, és egy *x* változóval jelölt szemantikai komponense.

A szemantikai oldalon megad egy esemény-változót (szituációs referenst), amiről elmondja, hogy a jelenben zajlik, az esemény *'ugrás'*, és egy x referens végzi.

Végül nézzük, hogyan épül fel egy konkrét mondat az UCG formalizmusa szerint:

(2) Péter talált egy vakondot.



2.2 LFG

A talán legelterjedtebb és legjelentősebb nem transzformációs elmélet a Lexikai-Funkcionális Grammatika (LFG[7]), melynek alapelveit más elképzelések is átvették, elsősorban a hatékony számítógépes alkalmazhatósága miatt.

Az elméletet Joan Bresnan és Ronald Kaplan dolgozták ki, vagyis egy nyelvész és egy számítógépes szakember közösen, amiből egyenesen következik két alapeszme: a számítógépes implementálhatóság és konzisztencia a pszicholingvisztikai elképzelésekkel nyelvelsajátításról és megértésről. A formális eszközökben ezen elvek alapján nem szerepelnek üres elemek és transzformációk. Az LFG felépítésének alapelvei:

1. *reprezentáció*: nyelvészeti függőségek formális kódolása,
2. *modularitás*: független általánosítások tényezőkre bontása,
3. *matematikai kezelhetőség*: bizonyítható matematikai elvek,
4. *implementálhatóság*: átlátható, hatékony implementáció.

A modularitáshoz tartozik a variabilitás és univerzalitás kérdése, miszerint a nyelvenként különböző *külső szerkezet*hez (LFG-modellje a kategoriális/összetevős szerkezet) azonos *belső szerkezet* (LFG-modellje a funkcionális szerkezet) tartozhat:

[7] Bresnan (2000), Komlósy (2001), Austin (2001)

Külső szerkezetek:	*Belső szerkezet:*
A nagy macskák üldöznek egy madarat.	PRED 'chase'
The big cats are chasing a bird.	SUBJ { PRED 'cats'; SPEC 'the'; MODS {[PRED 'big']} }
De grote katten jagen op een vogel. (holland)	OBJ { PRED 'bird'; SPEC 'a' }

5. példa: LFG – külső-belső szerkezet kapcsolata

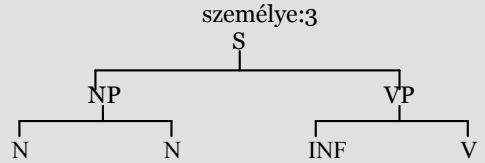
Az LFG több párhuzamos reprezentáció kifejezéseivel elemzi a mondatokat: 1.) argumentum/lexikai szerkezet vagy a-struktúra, 2.) funkcionális szerkezet vagy f-struktúra, 3.) kategoriális/összetevős szerkezet vagy c-struktúra, 4.) prozódiai szerkezet vagy p-struktúra, 5.) szemantikai szerkezet vagy σ-struktúra. Az *a-struktúra* a predikátum-argumentum viszonyokat kódolja, a predikátum argumentumainak számát és tematikus szerepeit. A *c-struktúra* az alkotóelemekkel foglalkozik, megadja a szavak közötti precedencia viszonyokat, a szavak kategóriáját és a fráziscsoportokat. Mindezt környezetfüggetlen frázisstruktúrákkal ábrázolja. A c-struktúra az LFG-ben a tulajdonképpeni felszíni szerkezet, ami nyelvspecifikus. Az *f-struktúrában* funkcionális információk szerepelnek – grammatikai funkciók mint SUBJ és OBJ, valamint diskurzus funkciók is mint például a TOPIK. Az f-struktúrák attribútum-érték párokból alkotott mátrixok, ahol az attribútumok lehetnek morfoszintaktikai jegyek (pl.: szám, személy, igeidő) vagy grammatikai funkciók.

A funkcionális relációkat a nyelvek a lexikai és a kategoriális szerkezetekben tárolják, a grammatikának pedig képesnek kell lenni ezen információk kifejtésére. Az LFG-ben ez úgy valósul meg, hogy az alapreprezentációkhoz egyrészt egy-egy üres f-struktúrát, valamint annotációkat rendel, amikből a kódolt információk kiolvashatók.

(3) Péter barátja úszni akar.

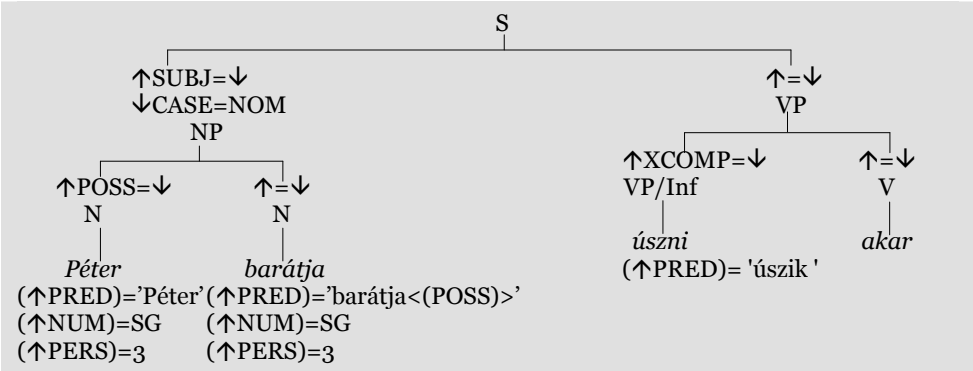
a-struktúrák:	Péter	barátja(POSS)	úszni(SUBJ)	akar(AG, TH)
kódolt	esete:NOM	vonzatai: birtokos	vonzatai: alany	vonzatai: alany, tárgy
információk:	száma:SG	esete:NOM		módja: kijelentő
	személye:3	száma:SG		alanya: E/3
		személye:3		tárgya: infinitiv

c-struktúra:

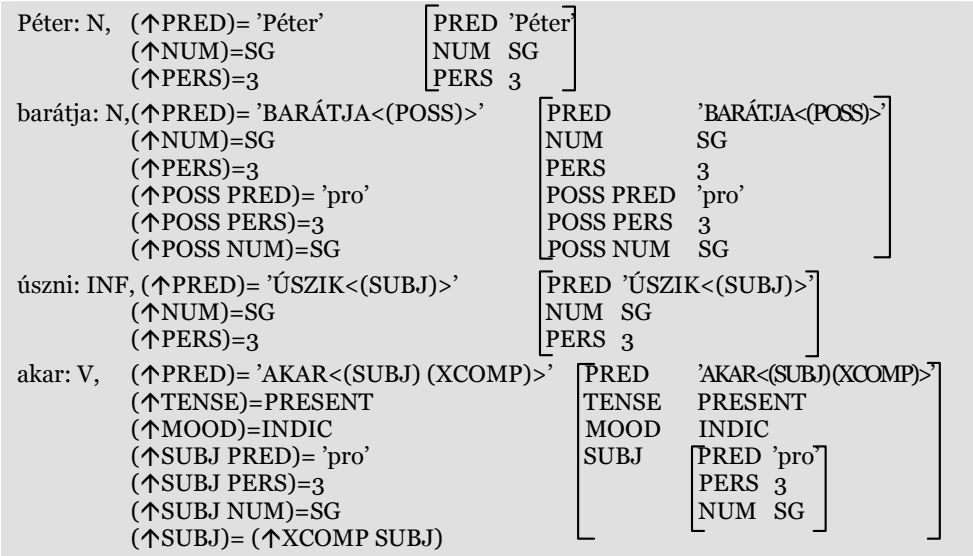


6. példa: a-struktúrák és c-struktúra a (3) mondathoz

A fenti összetevős szerkezet ábrázolható úgy is, hogy kiegészítjük a mondarészek konkrét grammatikai funkcióival, így megkapjuk az annotált c-struktúrát.



7. példa: annotált c-struktúra a (3) mondathoz



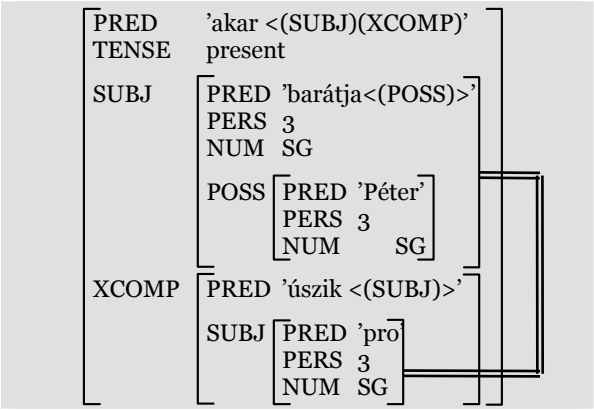
8. példa: annotált a-struktúrák és f-struktúrák a (3) mondathoz

Az f-struktúrák leírásában láthatók az attribútum-érték mátrixok. Az *akar* igén szemléltetve a szerkezet a következő: PRED jegy, amihez szemantikai érték rendelődik. Ez a szemantikai-érték az *akar* ige és a két vonzata: egy alany és egy predikatív vonzat, jelen esetben az infinitívuszi vonzat. A TENSE jegy az igeidő, ami jelen esetben jelen idő, a MOOD jegy pedig az igemód, melynek értéke itt indikatív, azaz kijelentő mód. A SUBJ jegyben az alanyáról ad információkat: az alanya harmadik személyű és egyes számú. Az annotált lexikai egységben szerepelnek ugyanezek az információk, valamint plusz információként az is, hogy az akar ige alanya azonos a predikatív vonzatának az alanyával (↑SUBJ)= (↑XCOMP SUBJ).

A 7. és 8. példában láthatók az annotációk, melyeknek formája minden esetben (↑...) vagy (↓...). Az első kifejezés mindig a szülőcsomópont egy

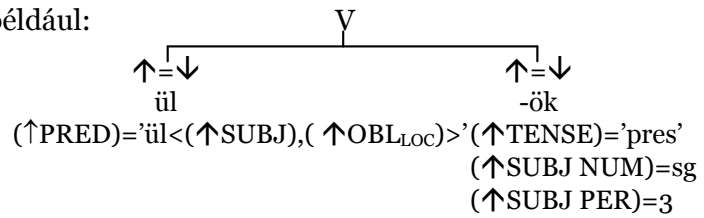
tulajdonságát mondja meg, a második pedig saját tulajdonságait adja. A barátja lexikai egység esetében például a birtokosra vonatkozó információ fent bemutatott módon azt mondja, hogy az ő szülőcsomópontjának alárendelt birtokos egy hangtanilag nem realizálódó névmás (pro), személye harmadik, száma pedig egyes szám. A ↑=↓ jelentése, hogy az annotált csomópont (↓) meghatározó eleme a fölötte lévő csomópontnak, vagyis a szülőcsomópontjának. A 7. példa NP-jéhez rendelt annotáció tehát azt mondja, hogy ő maga az alany vagy része az alanynak (↑SUBJ=↓) és a saját esete nominatív (↓CASE=NOM).

A fent bemutatott párhuzamos reprezentációkat az LFG-ben korrespondencia-leképezések kötik össze. Az annotált lexikai struktúrákból és az annotált c-struktúrákból a lexikai behelyettesítés műveletével kapjuk meg a felszíni szerkezetet. A felszíni szerkezetből kapjuk a funkcionális leírást (funkcionális egyenlőségek listáját), amelyre egy kiértékelő algoritmust alkalmazva megszületik a mondat funkcionális struktúrája.



9. példa: a (3) mondat f-struktúrája

Végül essék pár szó az LFG lexikon-értelmezéséről. Fontos elv, hogy az LFG-ben a szótár szintjén is feltételeznek generáló szabályokat, amelyek a morfémákból létrehozzák azokat a ragozott szavakat, amelyek a mondatban megjelennek. Így a lexikonban tárolt és az onnan kikerülő egységek csak ritkán azonosak. A lexikonból kimenő egységek leírásában szerepel az egység címszójának reprezentációja, a szintaktikai kategóriája, annotált funkcionális sémák listája és egyéb információk (pl.: morfológiai forma). A lexikai egységek lehetnek elemiek (8. példa) vagy szerkesztettek, például:



A szintaktikai információk lexikonban való kódolását a lexikai leképezés elmélete (Lexical Mapping Theory – LMT) fogalmazza meg. Ez a folyamat gondoskodik a helyes vonzatkeret kialakításáról. A alapötlet szerint a szintaktikai információkat analitikusan két bináris jegyre kell osztani: [±r] – ± tematikusan korlátozott és [±o] – ± tárgyjellegű, amelyeket – univerzális leképezési elvek alapján – az argumentumokkal kell társítani. E két jegy alapján a grammatikai funkciók természetes osztályokba sorolódnak:

	-o	+o
-r	SUBJ	OBJ
+r	OBL_{Θ}	OBJ_{Θ}

10. példa: LMT-természetes osztályok

A leképezési folyamat a lexikonban tárolt szemantikai argumentumokhoz (tematikus szerepekhez) történő funkcióhozzárendelés, amely három lépésben történik: 1. az argumentumok belső ismérvekkel való osztályozása, 2. morfolexiakai műveletek, 3. default besorolások. Az argumentumok osztályozása univerzálékon alapulnak, miszerint az ágens –o jegyű, tehát SUBJ vagy OBL lehet, a téma és páciens –r jegyű, tehát SUBJ vagy OBJ, a lokatív argumentum szintén –o jegyű, tehát SUBJ vagy OBJ lehet. A morfolexikai műveletek az argumentumstruktúrát változtatják, bevezetnek vagy elnyomnak egy-egy tematikus szerepet. Ilyen műveletek például a *passzivizálás*, amely az ágenst nyomja el. A default szerep-besorolás szerint a megjeleníthető argumentumok közül mindig a legmagasabb tematikus szereppel bíró lesz az alany. Ez alól csupán a lokatív inverzió jelensége a kivétel, ami azt jelenti, hogy (bizonyos nyelvekben) a tárgyatlan mondatban az alany és a határozói vonzat felcserélődik.

2.3 HPSG

A transzformációt elutasító elméletek[8] közül még az HPSG-ről (Head-Driven Phrase Structure Grammar [Fejvezérelt Frázisstruktúra Nyelvtan]; Sag-Pollard 1987, Szécsényi 1997) szólnék röviden.

Az HPSG hasonlóan az LFG-hez jegy-érték párokból felépülő jegymátrixokkal dolgozik. A lexikai egységeket és a frázisszerkezeteket egyaránt ezek segítségével fogalmazzák meg. A lexikonban tárolt egységeken lexikai szabályok működnek, amelyek bevezethetnek illetve megváltoztathatnak már létező jegyeket. Ezután a

[8] Ezen a ponton érdemes azt megjegyezni, hogy a korábban említett Joshi-féle faépítő nyelvtan (TAG, Tree Adjoining Grammar; Joshi–Levy–Takahashi 1975) szintén elutasítja a transzformációt, mindössze kész fákat épít össze.

default szabályok működnek, amelyek arról gondoskodnak, hogy az egység azon jegyei, amelyek a lexikonban nem kaptak értéket, mégis kapjanak valamilyen alapértéket. A szintén jegystruktúrákból álló frázisstruktúrákat a közvetlen dominanciaszabályok állítják elő, amely után a frázissal unifikálódó elvek – szintén frázisstruktúraként leírva – meghatározzák a szülőcsomópontok jegyeit. Ha az unifikáció lehetséges, vagyis a frázis minden elvnek megfelel, akkor újból alkalmazni kell a korábban említett default szabályokat. A szülőcsomópont gyermekeinek sorrendjéről a lineáris megelőzési szabályok gondoskodnak.

A lexikai egységek PHON jegyükben tartalmazzák a fonológiai információkat, SYNSEM jegyükben pedig szintaktikai és szemantikai információkat. Ha frázisról van szó, akkor harmadik jegye ezek mellett a DTRS jegy.

$$\begin{bmatrix} \text{PHON:ad} \\ \text{SYNSEM:} \begin{bmatrix} \text{CAT:} \begin{bmatrix} \text{HEAD:V} \\ \text{SUBJ:NP}_{(1)}\text{[nom]} \\ \text{COMPS:}\langle \text{NP}_{(2)}\text{[acc]}, \text{NP}_{(3)}\text{[dat]} \rangle \end{bmatrix} \\ \text{CONTENT:} \begin{bmatrix} \text{RELN:give} \\ \text{GIVER:(1)} \\ \text{GIVEN:(2)} \\ \text{GIFT:(3)} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

11. példa: HPSG jegymátrix

A CAT(egory) jegy főleg szintaktikai információkból áll, megadja kitüntetett helyen (SUBJ jegy) az alanyt, valamint a többi vonzatot (COMPS) függőségi sorrendben, e két utóbbi jegy szolgáltatja a valencia-értékeket. A CONTENT jegy tartalmazza a lexikai egység szituáció-szemantikai leírását, a fenti példában a reláció az *'adás'* folyamata, ahol három szereplő van, aki ad, aki kap, és maga a dolog, amit adnak. Az NP-knél használt indexekkel azonosítja is a szereplőket, vagyis azt, hogy melyik szemantikai argumentum melyik vonzattal jelenítődik meg.

3. Totális lexikalizmus – GASG

A 2. pont elején említett tendenciára – a lexikon felé való elmozdulásra – legradikálisabb példa a totális lexikalizmusra épülő GASG nyelvtan-modell (Generative/Generalized Argument Structure Grammar [Generatív/Általánosított Argumentum Struktúra Nyelvtan], Alberti 1999, Alberti-Balogh-Kleiber-Viszket 2002a). A generálás, mondatépítés egyetlen, kizárólagos mozgatója a lexikon. A szavak gazdag leírása tartalmaz minden szükséges információt ahhoz, hogy a bemeneti szavak

listájából, a numerációból (Chomsky, 1995) összeépülhessen(ek) a grammatikus mondat(ok). A GASG mondatgenerálásának az alapötlete, hogy a szemantikailag összetartozó szavak keresik egymást és egymás „szomszédságát" a mondatban. A kölcsönös keresés formálisan változók és konstansok[9] bevezetésével fogalmazható meg, melyek a keresés eredményeként unifikálódnak, megadva ezzel a grammatikus mondat morfoszintaktikai és (diskurzusreprezentációs) szemantikai elemzését.

A GASG által elképzelt lexikonban tárolt lexikai egységek mindegyike egy rendezett négyes, melynek tagjai: <sajátszó, szintaktikai információk, szemantikai információk, szintaxis és szemantika közti koreferenciaviszonyok>. A lexikai egységekben a szavak megadják saját tulajdonságaikat, valamint követelményeket fogalmaznak meg az általuk keresett környezeti szavakról. Ezen információk alapján keresik egymást – kölcsönösen – a lexikai egységek. Példaként nézzük meg részletesen a (4) mondatban szereplő *szeret* ige lexikai egységét.

(4) A magyar fiú szeret egy holland lányt.

A (4) mondatban szereplő lexikai egységek – négykomponensű formulahalmazok – a következők: <Λ1, Λ2, Λ3, Λ4, Λ5, Λ6, Λ7>, ahol Λ1 az *a* névelőhöz tartozó lexikai egység, Λ2 a *magyar* melléknévhez stb.

Λ1= < {v1 = *a(z)},*
{ref.def(v1), 3(α, V1.1), n.common(α, V1.1),
immprec(**α=5,** v1, V1.1), arg(α, CASE1, V1.1, V1.2)},
{∧(α, Q1.1(x1), Q1.2.GRF1(x1) },
{corr(V1.1, Q1.1), corr(V1.2, Q1.2),
corr(CASE1, GRF1)}>

Λ2 = <{v2 = *holland},*
{adj(v2), n.common(α, V2), immprec(**α=2,** v2, V2) },
{∧(α, Dutch(X2), Q2(X2)},
{corr(v2, Dutch), corr(v2, Q2)}>

Λ3 = < {v3 = *fiú},*
{n.common(v3), 3.sg(v3), arg(α, nom, v3, V3},
{∧(α, boy(X3), Q3.subj(X3)},
{corr(v3, boy) , corr(v3, Q3)}>

Λ4 = <{v4 = *szeret},*
{fin.past(v4), v.tr(v4), n(α, V4.11), 3.sg(α, V4.11),
arg(α, nom, V4.11, v4), ref(α, V4.12), n(α, V4.21),
arg(α, acc, V4.21, v4), 3(α, V4.21), ref.indef(α, V4.22),
immprec(**α=5**, V4.12, V4.11), immprec(**α=7,** V4.11, v4),
immprec(**α=5**, V4.22, V4.21), immprec(**α=7,** v4, V4.21)},
{∧(α, love(X4.1, X4.2), Q4.1(X4.1)),
∧(α, love(X4.1, X4.2), Q4.2(X4.2))},
{corr(v5, love), corr(V4.11, Q4.1), corr(V4.21, Q4.2)}>

Λ5 = < {v5 = *egy},*
{ref.indef(v5),
3.sg(α, V5.1), n.common(α, V5.1),
immprec(**α=3,** v5, V5.1),
arg(α, CASE5, V5.1, V5.2)},
{∧(α, x5, Q5.1(x5),
∧(α, Q5.1(x5), Q5.2.GRF5(x5) },
{corr(V5.1, Q5.1),corr(v5.2, Q5.2),
corr(CASE5, GRF5)}>

Λ6 = <{v6 = *magyar},*
{adj(v6), n.common(α, V6),
immprec(**α=2,** v6, V6) },
{∧(α, Hungarian(X6), Q6(X6)},
{corr(v6, Hungarian),
corr(v6, Q6)}>

Λ7 = < {v7 = *lányt},*
{n.common(v7), 3.sg(v7),
arg(α, acc, v7, V7},
{∧(α, girl(X7), Q7.obj(X7)},
{corr(v7, girl) , corr(v7, Q7)}>

12. példa: az (4) mondat lexikai egységei

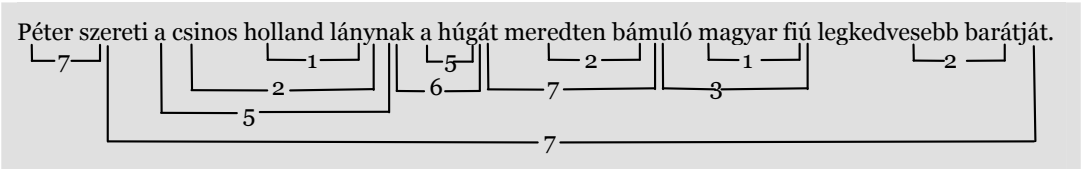
[9] A Prologban szokásos jelölés szerint a változókat nagybetűk, a konstansokat pedig kisbetűk jelölik.

A lexikai egységek tartalmát egy konkrét példán, a Λ4 lexikai egységen mutatom be. Az első komponensben láthatjuk az egység sajátszavát, a *szeret* igét. A Λ4 lexikai egység így csak olyan mondatok felépítésében vehet részt, amelyek ezt a szót pontosan ebben a formában tartalmazzák. A sajátszavak funkciója lényeges, mivel lehet például olyan eset (és ez gyakori a természetes nyelvekben), hogy azonos alakú szavak különböző grammatikai funkcióval társulhatnak. Például a *követ* szónak három teljesen különböző jelentése van, így lehet a mondatban alany (ragozatlan főnév, 'hírvivő, küldött' jelentésben), vagy tárgy (a *kő* szó tárgyragos alakjaként), vagy akár a finit ige (valaki halad valaki után). Erre a problémára az jelenti a megoldást, hogy a *követ* sajátszó három különböző lexikai egységhez tartozhat, amelyekben a fent jelzett különbségek szerint jelennek meg a viszonyok. Emellett szerepelhetnek a mondatban olyan lexikai egységek is, amelyekhez nem tartozik sajátszó. Ilyen egység tartozik például a fókusz operátorhoz, amely hangtanilag nem realizálódik, a mondatban való szereplésének azonban fontos szintaktikai és szemantikai következményei vannak.

A második komponensben szereplő információk az egység sajátszavának tulajdonságai, az egység által keresett környezeti szó/szavak tulajdonságai, valamint a sajátszó és a környezeti szó/szavak közti szintaktikai követelmények. Eszerint a *v4* sajátszóról a következőket mondjuk: múlt idejű, finit (fin.past(v4)), tranzitív és ige (v.tr(v4)). A saját tulajdonságok leírására azért van szükség, mert egy másik lexikai egység ezen tulajdonságok alapján fogja őt keresni. Még mindig a második komponensnél maradva jellemzünk egy V4.11, egy V4.12, egy V4.21 és egy V4.22 környezeti szót: a V4.11 egy egyes szám harmadik személyű főnév, amely nominatív esetben álló argumentuma a *szeret* igének; a V4.12 egy referenciális elem (a V4.11-hez tartozó névelő); a V4.21 újból egy főnév, amelyről azt tudjuk, hogy harmadik személyű, és akkuzatív esetben álló argumentuma az igének; végül V4.22 egy indefinit, referenciális elem (határozatlan névelő). Az *immprec(α=**N**, X, Y)* kifejezésekben a szomszédossági követelményeket fogalmazzuk meg, ahol az **α=N** számok a szomszédossági követelmény erősségére vonatkozó rangparaméterek. A GASG egyik legfontosabb ötlete az optimalitáselméleti alapeszméhez is igazodó rangparaméterek bevezetése, amelyeket tartalmazó formulák közvetlen vagy közvetett módon is kielégíthetőek. Ilyen rangparaméterek (immprec rangok) döntenek többek között a mondat szórendjéről. Az immprec rangok lényege, hogy egy *X* elem követelményei között szerepel, hogy szomszédos akar lenni a mondatban egy *Y* elemmel, valamint rögzíti, hogy ez a követelmény milyen erősségű, milyen rangú

($\alpha=1$, $\alpha=2$, $\alpha=3$, ...). Az $\alpha=1$ rang a legerősebb követelmény, a rang növekedésével az erősség csökken. Az *X* elem fenti követelménye tehát teljesül, ha közvetlenül szomszédos *Y*-nal. Azonban, ha ezzel egy időben ugyanazzal az *Y*-nal egy másik, *Z* elem is szomszédos akar lenni, akkor a szomszédosság közvetlenül csak az egyikre teljesülhet. Ilyen esetekben érvényesül a rangparaméterek jelentősége, vagyis a kettő közül a gyengébb ranggal szereplő elem kénytelen feladni a közvetlen szomszédosságot. Például ha a követelmények *immprec*($\alpha=3$, *X, Y,*...) és *immprec*($\alpha=2$, *X, Z,*...), akkor a helyes sorrend: *X, Z, Y*. Általánosan fogalmazva tehát, ha két elem között *N*-ed rangú *immprec* követelmény van, akkor vagy közvetlenül szomszédosnak kell lenniük, vagy beékelődhet közéjük egy (vagy akár több) elem, de csak olyan, amely $K \leq N$-ed rangú *immprec* viszonyban van valamelyikükkel. Azonban ez a „jogosan" beékelődő elem is hozhat magával újabb elemeket, amelyek azzal valahányad rangú *immprec* viszonyban vannak. Ennek szemléltetésére álljon itt egy példa:

(5) Péter szereti a csinos holland lánynak a húgát meredten bámuló magyar fiú legkedvesebb barátját.



13. példa: immprec rangok a (2) mondatban

A lexikai egység harmadik komponense tartalmazza a szemantikai jellemzést; itt két predikátum szerepel: love(X4.1, X4.2) és Q4.1(X4.1). Az első a lexikai egység saját predikátuma, miszerint X4.1 szereti X4.2-t, a második és harmadik két környezeti predikátum, amelyek azt mondják, hogy Q4.1 tulajdonságú az X4.1 változóval jelölt elem és Q4.2 tulajdonságú az X4.2-vel jelölt elem. X4.1 és X4.2 ideiglenes referensek, amelyek arra utalnak, hogy a *szeret* ige és a keresett főnevek között szemantikai kopredikációs viszony van. Ebben a komponensben meghatározott predikátumok alulspecifikált DRS (Discourse Represent Structure; Kamp-Reyle 1993, van Eijck-Kamp 1997) részletek, melyek a végső unifikáció során épülnek össze a mondat DRS-évé.

Végül a negyedik komponens megteremti a kapcsolatot a szintaxis és a szemantika között: a v4 a szeret predikátum megjelenítője, V4.11 pedig a Q4.1 predikátumé, és a V4.21 a Q4.2 predikátumé.

Összegezve tehát a *szeret* igének két vonzata van (SUBJ, OBJ), amelyeket keres a mondatban. A *szeret* ige lexikai egységében azonban négy keresett elemet látunk. Ennek magyarázata, hogy a GASG elképzelése alapján a régensek két pilléren keresik a vonzataikat (Alberti 1997), vagyis keresik az alanyi vonzat determinánsi és főnévi pillérét és a tárgyi vonzat determinánsi és főnévi pillérét. Ezt az eljárást a magyar nyelv egyeztetési tulajdonságai kívánják meg, mivel a finit ige alanyával számban és személyben egyeztet, ami a vonzat főnévi csoportjának főnevén látszik, valamint definitségben egyeztet a tárgyával, amely viszont a névelőn (determinánsi elemen) jelenik meg.

(6) a. Én szeretem a fiút.
b. A lány szereti a fiút. c. Egy lány szereti a fiút.
d. A lányok szeretik a fiút. e. A lányok szeretnek egy fiút.

Adott lexikai egységeknek a bemeneti numerációba való beválogatása után megkezdődik az unifikációs eljárás, amely során ezek a lexikai egységek keresik egymást az alapján, hogy egy adott sorrend mellett egymás környezeti feltételeit kielégítik-e. Az unifikáció során a környezeti szavakat sajátszavakkal kell azonosítani, a környezeti predikátumokat sajátpredikátumokkal, a változó típusú referenseket pedig konstans referensekkel.

v1	V4.12
v2	
v3	V4.11, V1.1, V2
v4	V3, V7
v5	V4.22
v6	
v7	V4.21, V5.1, V6
Hungarian	
boy	Q4.1, Q1.1, Q2
Dutch	
girl	Q4.2, Q5.1, Q6
love	Q1.2, Q3, Q5.2, Q7
x1	X2, X3, X4.1
x5	X6, X7, X4.2
a végső DRS:	
x1 ∧ Hungarian(x1) ∧ boy(x1) ∧ love(x1, 5) ∧ x5 ∧ Dutch(x5) ∧ girl(x5)	

14. példa: unifikáció az (1) mondathoz

4. A totális lexikalizmus implementációja

A totális lexikalizmus számítógépes implementációjának kidolgozásán a GeLexi kutatócsoport[10] tagjaként dolgozom. A GASG implementációjára a Prolog programnyelv a legalkalmasabb, mivel ugyanazokat a technikákat alkalmazza (unifikáció, rekurzió). A generatív alapfeladattal összhangban a program célja, hogy egy adott szósorról egyértelműen eldöntse annak grammatikalitását, valamint grammatikus mondatok esetén közölje annak morfoszintaktikai és (diskurzus-) szemantikai elemzését. A szemantikai reprezentációt DRS-ek formájában, a DRT formalizmusával adjuk vissza.

Számítógépre korábban általában frázisstruktúra nyelvtanokat alkalmaztak, melyek azonban a magyarra is jellemző szabad szórendet nehezen kezelik. Ezzel ellentétben a lexikalista nyelvtanok megközelítésmódja és eszközrendszere lehetővé teszi az ilyen típusú nyelvek hatékonyabb kezelését.

4.1 A Prolog nyelv

A Prolog (PROgramming in LOGic – programozás logikában; Dömölki 1986, Márkusz 1988) matematikai logikára, az elsőrendű predikátumkalkulusra épülő magas szintű programnyelv, amely napjainkban az egyik legfontosabb kelléke a mesterségesintelligencia-alkalmazások programozásának, valamint a számítógépes nyelvészeti kutatásoknak. A Prolog az 1970-es évek elején született meg Franciaországban[11], az 1980-as években még több logikai programozási nyelv követte – például a Mercury nyelv, főleg ipari alkalmazásokra vagy a CLP (Constraint Logic Programming) nyelvcsalád.

A programozási nyelveket alapvetően két nagyobb csoportba sorolják: imperatív nyelvek[12] és deklaratív nyelvek[13] csoportjába. Az imperatív nyelvek esetében az algoritmus pontos megadásán van a hangsúly, vagyis azt kell definiálni, hogy hogyan oldjuk meg az adott problémát. A Prolog a deklaratív nyelvek osztályába tartozik, amely abban különbözik az algoritmikus nyelvektől, hogy nem azt kell definiálni, hogy hogyan, hanem azt, hogy mit csináljon a program, és a Prolog saját nyelvi rendszerébe van beépítve az eljárás, amely a megoldáshoz vezet.

---

[10] A kutatócsoport további tagjai: Alberti Gábor, Kleiber Judit és Viszket Anita. A csoport 2000 szeptemberétől dolgozik együtt a Pécsi Tudományegyetem Nyelvtudományi Tanszékén.

[11] Az ötlet Robert Kowalskitól származik, az első megvalósítás a marseille-i egyetemen született 1972-ben, Alain Colmerauer kutatócsoportja által.

[12] C, C++, Fortran, Algol stb.

[13] Ezen belül is két fő csoport a *funkcionális* nyelvek: LISP, MSL stb., valamint a *logikai* nyelvek: Prolog, SQL, CLP stb. nyelvek csoportja.

Ez az eljárás a deduktív bizonyítás, ahol kevés számú adott állításból, logikusan, egymásra épülve következnek újabb és újabb állítások.

A Prolog alapvető működési stratégiája a *rezolúció* és az *unifikáció* (illesztés változókkal). A rezolúció egy olyan módszer, amely először az elsőrendű logikai formulák halmazát klózok halmazává alakítja, majd az axiómahalmazt kiegészíti a bizonyítandó tétel negáltjával. A következtetési eljárás során ha egy klóz tartalmaz egy atomi állítást, egy másik pedig annak negáltját, akkor ezek a predikátumok elhagyhatók, a megmaradt részek pedig egy klózzá alakíthatók. Ha ezt a módszert folytatva eljutunk a hamis állítást képviselő üres klózig, akkor a formulahalmaz kielégíthetetlen, vagyis eredeti tételünk ellentmondásmentes, tehát a tétel logikai következménye axiómáinknak. A rezolúció tehát egy cáfoló eljárás, amely a premisszák és a konklúzió tagadásának felvételével ellentmondásra próbál jutni, erről a halmazról próbálja bebizonyítani, hogy kielégíthetetlen. Ha ez sikerül, az állítás igaz. A rezolúció előnye az is, hogy konstruktív bizonyítási eljárás, vagyis nem csak arra ad választ, hogy van megoldás, hanem meg is konstruálja azt. A rezolúció mellett a másik fontos eljárás az unifikáció, amely egy komplex illesztési mechanizmus predikátumok és aktuális tények egymásnak való megfeleltetésére. A Prolog a klózokat „érti meg", azok között keres úgy, hogy illeszteni próbálja a bennük szereplő terminusokat (változókat, konstansokat). Ha valahol több lehetősége van, először mindig az elsőt próbálja meg illeszteni, és ha ez nem sikerül, *visszalép* (backtrack), és a következőt próbálja illeszteni.

Ahogy a logikai formulák között is megkülönböztetünk atomi ($P(x)$) és összetett ($P(x) \vee Q(x)$, $P(x) \rightarrow Q(x)$ stb.) formulákat, a Prolog-klózok között is két típussal találkozunk. Az egyik típus a *tényállítás* ($P(x)$), a másik pedig a *szabály*. Ez utóbbi a logikai implikációnak felel meg. A Prolog-ban ezt úgy fejezzük ki, hogy „q akkor, ha p", jele: *q :- p*. A szabályok alakja a következő: $predikátum_1(terminus_1, ..., terminus_n)$:- $predikátumnév_2(terminus_1', ..., terminus_k')$, ..., $predikátumnév_3(terminus_1'', ..., terminus_s'')$, ahol a terminus lehet változó vagy konstans, a vessző pedig a logikai *és*-t helyettesíti[14]. A program végén egy *célállítást* fogalmazunk meg (goal), ez felel meg körülbelül a logikai konklúziónak.

[14] A változóknak nincsen „állandó" értékük, csak egy klózon belül jelölik ugyanazt.

```
Tényállítások:
anyja("Anna", "Péter").
anyja("Mari", "Kati").
anyja("Kati", "Feri").
házastárs("János", "Anna").
házastárs("Zoli", "Mari").
házastárs("András", "Kati").
Szabályok:
apja(X, Y) :- anyja(Z, Y), házastárs(X, Z).
nagyapja(X, Y) :- apja(X, Z), apja(Z, Y).
nagyapja(X, Y) :- apja(X, Z), anyja(Z, Y).
goal
apja("János", "Péter").
```

15. példa: A Prolog programok felépítése

Az 15. példában azt kérdezzük, hogy igaz-e az, hogy Péternek János az apja. A választ úgy keresi meg, hogy meghívja az *apja* predikátumot úgy, hogy az *X* helyére *János*-t, az *Y* helyére pedig *Péter*-t *illeszti*. Ezután megpróbálja a szabály jobb oldalán álló predikátumokat *kielégíteni*, megnézi, hogy ha *X* minden előfordulásába *János*-t, *Y*-éba pedig *Péter*-t helyettesít, nem jut-e ellentmondásra. Először az *anyja(Z, "Péter")*-t próbálja. A tényállítások között megnézi, hogy talál-e olyat, amely erre illeszthető[15]. Megtalálja az *anyja("Anna", "Péter")* klózt. A *Péter* konstanst sikeresen illeszti az azonos *Péter* konstanssal, a *Z* változót pedig az *Anna* konstanssal. Ezután visszatér az *apja* predikátumhoz, és miután az *anyja* predikátumot sikerült kielégítenie, megnézi a *házastárs* predikátumot is. Ebben az esetben már mindkét változó helyén konstans szerepel (hiszen a *Z* helyére már *Anná*-t helyettesítettünk), tehát egyetlen mód a klóz kielégítésére, ha a tényállítások között szerepel olyan, hogy *házastárs("Anna", "János")*. Mivel ez a klóz tényleg szerepel, a *házastárs* predikátum is kielégíthető. Így, mivel az *apja("János", "Péter")* kielégítésére vonatkozó mindkét feltétel teljesült, a program kiírja, hogy *Yes*, és leáll. Tehát a célállítás igaz. Ha minden lehetőség végigpróbálása után nem talált volna helyes megoldást, akkor *No* válasszal állt volna le, ami annyit jelent, hogy a célállítás hamis.

A Prologot nem csak állítások igazságértékének megállapítására használhatjuk, hanem arra is, hogy kiegészítendő kérdésekre kapjunk választ. Az előző példánál maradva, megkérdezhetjük például, hogy ki az apja Péternek. Ebben az esetben a célállítás a következő: *apja(X, "Péter")*. Ekkor eredményként megkapjuk, hogy létezik-e megoldás, és ha igen, azt is, hogy milyen helyettesítés mellett. Az 15. példa esetében az *apja(X, "Péter")* célállításra adott válasz: *X=János, Yes*. Így akár az összes apa-gyermek párt is lekérdezhetjük a következőképpen: *apja(X, Y),* ahol a válasz: *X=János, Y=Péter; X=Zoli, Y=Kati; X=András, Y=Feri; 3 solutions, Yes*.

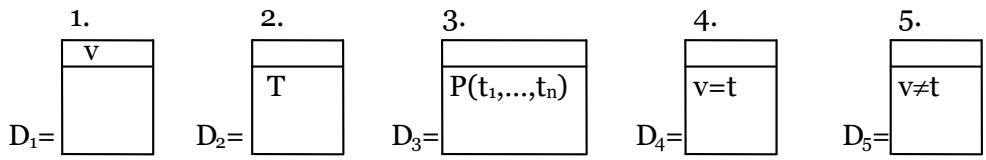
[15] Változót bármivel, konstanst azonos konstanssal lehet illeszteni, de a típusoknak is meg kell egyezni.

4.2 DRT – alapok röviden

Programunkban a szemantikai elemzést a Kamp-Reyle (1993) és van Eijck-Kamp (1997) féle Diskurzus Reprezentációs Elmélet struktúráival, DRS-ekkel reprezentáljuk. A DRS-ek felépítése a következőképp történik. Adott három halmaz: C: konstansok halmaza, eleme c; U: változók (diskurzus referensek) halmaza, eleme v; **P**: predikátumnevek halmaza, eleme P. Minden DRS meghatározott atomi, vagy proto-DRS-ekből épül fel. DRS-ek a következők:

1. $<\{v\}, \emptyset>$, ha v referens (U-beli);
2. $<\emptyset, \{T\}>$;
3. $<\emptyset, \{Pt_1,\ldots, t_n\}>$, ha P egy n-argumentumú predikátum és $t_1,\ldots,t_n$ termek;
4. $<\emptyset, \{v=t\}>$, ha v referens és t term;
5. $<\emptyset, \{v\neq t\}>$, ha v referens és t term.

A könnyebb áttekinthetőség kedvéért nézzük ezeket a DRS-eket az úgynevezett „dobozos ábrázolással" (a kettéosztott dobozok felső részében szerepelnek a változók, az adott diskurzus referensei, az alsó részben pedig a predikátumok, vagyis az adott diskurzus referenseiről tett állítások):



A proto-DRS-ekből két szabály állíthatja elő a DRS-t:

1. ha $D=<X,C>$, $D'=<X',C'>$ DRS-ek, akkor $D\oplus D'$ is DRS, ahol $D\oplus D':=<X\cup X', C\cup C'>$;
2. ha D egy DRS, akkor $<\emptyset, \neg D>$ is egy DRS.

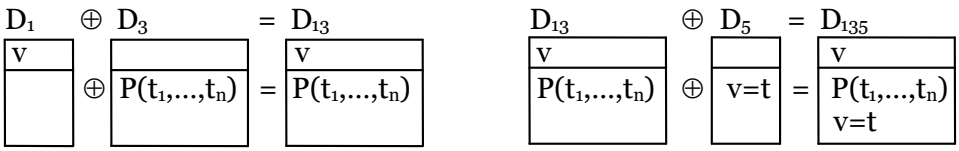
Tehát a proto-DRS-ek és a szabályok alapján előállítható például:

$D_1\oplus D_3 = D_{13}$: $<\{v\}, \emptyset>\oplus<\emptyset, \{Pt_1,\ldots, t_n\}>=<\{v\}, \{Pt_1,\ldots, t_n\}>$

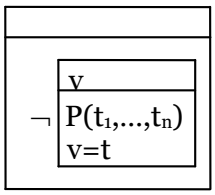
$D_{13}\oplus D_5 = D_{135}$: $<\{v\}, \{Pt_1,\ldots, t_n\}>\oplus<\emptyset, \{v=t\}>=<\{v\}, \{Pt_1,\ldots, t_n, v=t\}>$

negáció: $D_{n135} = < \emptyset, \neg D_{135} > = < \emptyset, \neg<\{v\}, \{Pt_1,\ldots, t_n, v=t\} >>$

Dobozokkal mindez:



negáció D_{135}-re:



4.3 Implementáció a szavak szintjén

Hasonlóan más számítógépes alkalmazásokhoz a GASG implementációja is az elmélet leíró adekvátságát hivatott igazolni, továbbá alkalmassá kívánja tenni az elméletet és annak gyakorlati megvalósulását arra, hogy minél többet tudjon elmondani a nyelvről és funkciójáról.

A kutatás során kifejlesztett első programverzió semleges magyar mondatokat képes elemezni (szintaktikai és szemantikai reprezentációval) szavak szintjén, beépítve a GASG alapelképzelését. A program két lényeges pontja – a GASG elméletét követve –, hogy a mondatban megjelenő szavakhoz lexikai egységeket rendel, melyekben azokat az információkat tárolja, hogy az adott elem milyen grammatikai viszonyokat tud és akar létesíteni – így a grammatikalitás-ellenőrzést lényegében úgy végzi, hogy rendre megvizsgálja, hogy ezek az elvárások teljesülnek-e. A másik lényeges pont a közvetlen megelőzési viszonyok kezelése, amely a helyes szórend kialakításáért felelős, és a szabad szórendű nyelvek esetében a korábbi eljárásoknál – mint például a kategoriális nyelvtanok függvényalkalmazási művelete (Zeevat 1991) – hatékonyabb módszernek bizonyult. A szemantikai reprezentációt DRS-ekkel reprezentáljuk. A szemantikai elemzés bemeneteként minden esetben egy konkrét morfoszintaktikai elemzés szolgál.

A program bemenetként egy szósort kap. Első lépésként a szavakat beszámozza *p(szám, szó)* párokat alkotva, erre a szavak egyértelmű azonosításához van szükség[16], valamint ezt a számot fogja használni a program a szemantikai elemzés során a referensek jelölésénél. A számozás után a bemeneti szósor alapján a sajátszavakhoz kiválogatja a lexikai egységek saját tulajdonságokat tartalmazó részét.

```
lexitem("holland",adj("Dutch")).
lexitem("szerető", lif("love")).
lexitem("egy",det("a","indef")).
lexitem("fiú",ln("boy","NOM",arg(f("sg",3),s("none",0,"none")))).
lexitem("fiút",ln("boy","ACC",arg(f("sg",3),s("none",0,"none")))).
lexitem("lánynak",ln("girl","DAT",arg(f("sg",3),s("none",0,"none")))).
lexitem("fiú",ln("boy","POSS",arg(f("sg",3),s("none",0,"none")))).
lexitem("fiúnak",ln("boy","GEN",arg(f("sg",3),s("none",0,"none")))).
lexitem("barátomat",ln("friend","ACC",arg(f("sg",3),s("sg",1,"none")))).
lexitem("szeretnék",lf("love",feat("pres","cond"),arg(f("pl",3),s("none",0,"def")))).
lexitem("szeretnék",lf("love",feat("pres","cond"),arg(f("sg",1),s("none",0,"indef")))).
```

16. példa: lexikai egységek szavak szintjén

[16] Gyakori, hogy azonos alakú szavak szerepelnek a mondatban pl.: *A holland követ egy zöld követ követ*, és az elemzés minden egyes pontján fontos ezek egyértelmű megkülönböztetése

A lexikai egységek első eleme mindig az egység sajátszava, az a teljesen ragozott szó, ami a mondatban megjelenik. A sajátszót követi az egység „címkéje” az egységre vonatkozó saját tulajdonságokkal. A címkék első helyén mindig az angol megfelelő szerepel, majd a saját jegyei. A lexikai egységek felépítése: *melléknevek*nél és *igenevek*nél: adj(angol_név) illetve lif(angol_név), *determinánsok*nál: det(angol_név, definitség), *főnevek*nél: ln(angol_név, eset, arg(f(szám, személy), s(birtokos_száma, birtokos_személy, kitöltetlen_hely[17])), *finit igék*nél: lf(angol_név, feat(igeidő, igemód), arg(f(alany_száma, alany_személye), s(kitöltetlen_hely, kitöltetlen_hely, tárgy_definitsége)).

A lexikai egységek grammatikai követelményei a `relations` predikátumokban vannak tárolva.

```
relations(N, SZERET, lf(X, F, A),
[gr("subj", "d", N, K), gr("subj", "n", N, M), gr("obj", "d", N, L),
gr("obj", "n", N, J)], LIL2):-
      subjd(Q, SZERET), in(li2(K, Q, LAB), LIL2),
      prec(li2(K, Q, LAB), li2(N, SZERET, lf(X, F, A)), LIL2),
      subjn(R, SZERET, LIL2), in(li2(M, R, LAB2), LIL2),
      prec(li2(M, R, LAB2), li2(N, SZERET, lf(X, F, A)), LIL2),
      objd(S, SZERET, LIL2), in(li2(L, S, LAB3), LIL2),
      prec(li2(N, SZERET, lf(X, F, A)), li2(L, S, LAB3), LIL2),
      objn(Z, SZERET, LIL2), in(li2(J, Z, LAB4), LIL2),
      prec(li2(N, SZERET, lf(X, F, A)), li2(J, Z, LAB4), LIL2),
      prec(li2(K, Q, LAB), li2(M, R, LAB2), LIL2),
      prec(li2(L, S, LAB3), li2(J, Z, LAB4), LIL2).
```

17. példa: relations *predikátum szavak szintjén*

A *szeret* ige keresi két vonzatának (SUBJ, OBJ) determinánsi és főnévi pilléreit. A grammatikai viszonyai úgy töltődnek ki, hogy keressük azokat az elemeket, amellyel az ige *subjd, subjn, objd, objn* viszonyban lehet, majd megnézzük, hogy az adott elem és a hozzá tartozó szám szerepelnek-e a numerációban; ha igen, akkor az a szám töltődik be a rendezett hármas utolsó helyére, ha nem, akkor addig keres, míg egy ilyet nem talál. Egy előzetesen javasolt lexikai-egység listát folyamatosan figyelni kell, így biztosítjuk, hogy mindig az adott numeráció keretein belül maradunk. Hogy milyen viszony létesítéséhez mik a feltételek, azt a következő predikátumok határozzák meg:

[17] Erre a kitöltetlen helyre azért van szükség, hogy így párhuzamosan tudjuk leírni a főnevek és igék argumentumszerkezetét, ahol az első argumentum mindig az alany (főneveknél ez önmaga), a második argumentum pedig a tárgy (igéknél) illetve a birtokos (főneveknél):

lexitem(sajátszó, ln(ang_név, "ACC", **arg(f("sg",3), s("sg",1,"none"))**)).
lexitem(sajátszó, lf(ang_név, feat("pres","cond"), **arg(f("pl",3), s("none",0,"def"))**)).

```
subjd(X, Z):-
      ref(X), verb(Z).
subjn(X, Z, LIL2):-
      in(li2(_, X, ln(_, "NOM", arg(Y, _))), LIL2),
      in(li2(_, Z, lf(_, _, arg(Y, _))), LIL2).
objd(X, Z, LIL2):-
      in(li2(_, X, det(_, Y)), LIL2),
      in(li2(_, Z, lf(_, _, arg(_, s(_, _, Y)))), LIL2).
objn(X, Z, LIL2):-
      in(li2(_, X, ln(_, "ACC", _)), LIL2), transitive(Z).
stb.
```

18. példa: grammatikai viszonyok feltételei

A 18. példában bemutatott predikátumok tartalmazzák az egyeztetés ellenőrzését, például az ige és alanya közötti *subjn* relációban egyrészt rögzítjük azt, hogy a relációban szereplő főnévnek nominatív esetben kell állnia, másrészt azt, hogy számban azonosnak kell lennie. Az azonos Y változó biztosítja, hogy ez a jegy mindkét lexikai egységben ugyanaz.

Mindezek alapján (a korábban már beszámozott) szavak listájához meghatároz egy lexikaiegység-listát, amelyben a `satisfy` predikátum ellenőrzi egyrészt a kölcsönös keresést, valamint a helyes szórendért felelős `immprec` relációkat.

A kölcsönös vagy globális keresés ellenőrzése egy egyszerű lépés, ha a grammatikai-viszonyok listájában szerepel egy *gr(„relációnév”, N, M)* hármas – ahol N és M számok a relációban szereplő szavak számai –, akkor szerepelni kell egy *gr(„relációnév”, M, N)* hármasnak – természetesen azonos relációnév mellett. A globális keresés fontos a lokális keresés elégtelensége miatt, ami elfogadná például a **A lány a fiú ül* agrammatikus szósort, hiszen minden szó talál a feltételeinek megfelelő környezeti szavakat, az egyeztetési és esetrelációk is megvalósulnak. A lány és a fiú köznevek megtalálják az igét *subjn* viszonyra, a két névelő szintén megtalálja az igét *subjd* viszonyra, az ige talál egy determinánst és egy főnevet *subjd* és *subjn* viszonyokra. Így előáll a következő grammatikai-viszony lista: *[gr(subj, d, 1, 5), gr(subj, n, 2, 5), gr(subj, d, 3, 5), gr(subj, n, 4, 5), gr(subj, d, 5, 1), gr(subj, d, 5, 2)]*, ahol a *gr(subj, d, 3, 5)* és *gr(subj, n, 4, 5)* viszonyoknál nem valósul meg a kétirányúság.

A második lépés a helyes szórend ellenőrzése az `immprec` predikátummal, amihez a grammatikai relációkhoz tartozó rangparamétert a `grimmprec` predikátumok szolgáltatják, vagyis itt rögzítjük, hogy milyen relációhoz milyen erősségű szomszédosság a feltétel.

```
grimmprec(1, "adj", li2(A, X, _), li2(B, Y, _), GRALIST):-
        nationname(X), noun(Y), in2(gr("adj", A, B), GRALIST).
grimmprec(2, "adj", li2(A, X, _), li2(B, Y, _), GRALIST):-
        otheradj(X), noun(Y), in2(gr("adj", A, B), GRALIST).
grimmprec(3, "adj", li2(A, X, _), li2(B, Y, _), GRALIST):-
        numeral(X), noun(Y), in2(gr("adj", A, B), GRALIST).
grimmprec(4, "possn", li2(A, _, _), li2(B, _, _), GRALIST):-...
grimmprec(5, "detn", li2(A, _, _), li2(B, _, _), GRALIST):-...
grimmprec(6, "genn", li2(A, _, _), li2(B, _, _), GRALIST):-...
grimmprec(7, "subjn", li2(A, _, _), li2(B, _, _), GRALIST):-...
grimmprec(7, "objn", li2(A, _, _), li2(B, _, _), GRALIST):-...
grimmprec(7, "obln", li2(A, _, _), li2(B, _, _), GRALIST):-..
```

20. példa: az immprec predikátum működése

A fenti példában látható, hogy a nemzetiségnevekkel alkotott melléknévi kapcsolat rangja a legerősebb (1), majd 2-es rangú a melléknevekkel és 3-as rangúak a számnevekkel alkotott jelzős szerkezetek. A rövid birtokos 4-es, a névelő 5-ös, a hosszú birtokos 6-os, végül minden vonzat 7-es erősségű ranggal keresi a környezeti szót. Ezek a rangok biztosítják a helyes szórendet a (5) példamondatban is, valamint itt derül ki a (8)-(10) mondatok agrammatikalitása.

(7) Péter szereti a csinos holland lánynak a húgát meredten bámuló magyar fiú legkedvesebb barátját.

(8) *Péter szereti a holland csinos lánynak ...

(9) *Péter szereti a csinos holland a lánynak húgát ...

(10) *... a húgát bámuló meredten magyar fiú legkedvesebb barátját.

Az `immprec` predikátum működése a következőképp szemléltethető:

```
a.)   immprec(α, X, Y):-
            immprecfact(X, Y).
      immprec(α, X, Y) :-
             [immprec(β, X, Z) ∨ immprec(β, Z, Y], α ≥ β,
            in(Z, [X,...,Y]),
            legitimate([X,...,Z]), legitimate([Z,...,Y]).
b.)   legitimate([X, Y]).
      legitimate([X,...,Y]) :-
             [immprec(_, X, W) ∨ immprec(_, W, Y],
            in(W, [X,...,Y]),
            legitimate([X,...,W]), legitimate([W,...,Y]).
```

21. példa: az immprec predikátum működése

Vagyis α rangú szomszédossági követelmény teljesül X és Y között, ha azok ténylegesen szomszédosak a mondatban, vagy ha nem szomszédosak, akkor a köztük lévő Z elem β≤α rangban akar szomszédos lenni valamelyikőjükkel, és az X-től Z-ig terjedő lista, valamint a Z-től Y-ig terjedő lista legitimek. Egy lista legitim, ha mindössze két elemből áll, vagy a két szélső elem közti elem(ek) szomszédossági

viszonyban van(nak) valamelyikőjükkel. Ezen a szinten az immprec reláció rangja bármekkora lehet.

Ha minden fent említett predikátum kielégíthető, akkor a mondat morfoszintaktikailag grammatikus. A `goal` szekcióban lekérdezhetjük külön a morfoszintaktikai elemzését egy szósornak, ekkor eredményként megkapjuk a numerációban szereplő lexikai egységeket, és a mondatban az adott lexikai egységhez rendelhető grammatikai relációkat. Az elemek közti viszonyokat az adott szavakhoz tartozó számokkal reprezentáljuk. A 22. példában a következő mondathoz rendelt morfoszintaktikai elemzést látjuk:

(11) A lány barátja talált három Marira büszke fiút.

```
li2(1,"a",det("the","def"))
        [gr("detn",1,3),gr("subjd",1,4)]
li2(2,"lány",ln("girl","POSS",arg(f("sg",3),s("none",0,"none"))))
        [gr("possn",2,3)]
li2(3,"barátja",ln("friend","NOM",arg(f("sg",3),s("sg",3,"none"))))
        [gr("detn",3,1),gr("possn",3,2),gr("subjn",3,4)]
li2(4,"talált",lf("find",feat("past","indic"),arg(f("sg",3),s("none",0,"indef"))))
        [gr("subjd",4,1),gr("subjn",4,3),gr("objd",4,5),gr("objn",4,8)]
li2(5,"három",det("three","indef"))
        [gr("detn",5,8),gr("objd",5,4)]
li2(6,"Marira",ln("Mary","OBL",arg(f("sg",3),s("none",0,"none"))))
        [gr("obld",6,7),gr("obln",6,7)]
li2(7,"büszke",adj("proud"))
        [gr("adj",7,8),gr("obld",7,6),gr("obln",7,6)]
li2(8,"fiút",ln("boy","ACC",arg(f("sg",3),s("none",0,"none"))))
        [gr("detn",8,5),gr("objn",8,4)]
```

22. példa: morfoszintaktikai elemzés szavak szintjén

A morfoszintaktikai elemzés mellett azonban a szemantikai reprezentációt is képes a program az adott mondathoz rendelni. Ezt a műveletet a `semantics` predikátumok végzik.

```
sem(N, _, det(A, _), _, [provideref(A, ref(N))], _, _).
sem(N, _, ln(A, _, arg(f(_, 3), s("none", 0, "none"))),
    [gr("detn", N, K), _], [pred(A, [ref(K)])], _, _).
sem(N, X, ln(A, _, arg(f(_, 3), s("none", 0, "none"))), [_, _],
    [provideref("propn", ref(N)), pred(A, [ref(N)])], _, _):-
        nprop(X).
sem(N, _, lf(A, feat(B, _), _),
    [gr("subjd", N, K), gr("subjn", N, M), gr("objd", N, L), gr("objn", N, _)],
    [provideref("state", ref(N)), pred(A, [ref(N), ref(K), ref(L)]),
    pred(B, [ref(N)])], _, LIL):-
        in3(li(M, X, _, _), LIL), human(X).
```

23. példa: semantics predikátumok szavak szintjén

Minden `semantics` predikátum az előzetesen elképzelt, majd ellenőrzött morfoszintaktikai elemzést is használja, elsőként szemantikai

szelekciókat végez a mondaton, majd végső kimenetként megkapjuk a mondat DRS-ét. Ezek a predikátumok DRS részleteket (proto-DRS-eket) definiálnak. DRS hozzárendelés mondatokhoz a DRT szerint történik. A szemantikai elemzés során mindig azt kell rögzítenünk, hogy a sajátpredikátum honnan kap referenst. A referens forrása, bevezetője általában a determinánsi elem. A finit elem is bevezet egy szituációs referenst; a többi referenseit, melyekről állítást tesz, a vonzatainak determinánsi pillérén keresztül kapja. Ugyanígy a köznevek is a névelőjükön keresztül kapnak referenst, a szabadon csatolt melléknév pedig az általa módosított köznév névelőjén keresztül.

Ezek alapján a (11)-es mondathoz rendelt szemantikai reprezentáció a következő – ami ekvivalens a DRT „dobozos" ábrázolásával:

A programban:

```
[providerefe("the",ref(1))]
[pred("girl",[ref(301)])]
[providerefe("the",ref(301)),
    pred("friend",[ref(1),ref(301)])]
[providerefe("state",ref(4)),
    pred("find",[ref(4),ref(1),ref(5)]),
    pred("past",[ref(4)])]
[providerefe("three",ref(5))]
[providerefe("propn",ref(6)),
    pred("Mary",[ref(6)])]
[pred("proud",[ref(5),ref(6)])]
[pred("boy",[ref(5)])]
```

DRT szerint:

r4, r1, r301, r5, r6
state(r4) find(r4, r1, r5) girl(r301) friend_of(r1, r301) proud_of(r5, r6) boy(r5) r6=Mary

24. példa: szemantikai elemzés kimenete

Érdekes terület a birtokos szerkezetek leírása, ugyanis morfoszintaktikai és szemantikai ábrázolásuk során több alesettel kell számolnunk a magyar birtokos szerkezetek sokfélesége miatt. Mind a morfoszintaxisban, mind a szemantikában külön kell kezelni a hosszú és rövid birtokosokat valamint a beágyazott birtokos szerkezeteket.

A morfoszintaktikai elemzés során a birtokszó alapesetben keresi vonzatként a birtokosát, és annak főnévi és determinánsi pillérét. A magyar nyelvben azonban ez a birtokos lehet hosszú (*-nAk* ragos) vagy rövid (ragozatlan), és ezt a különbséget a viszonynevekben is szeretnénk elkülöníteni. Ezért a rövid birtokossal alkotott viszonyok neve *poss-n* és *poss-d*, a hosszú birtokossal pedig *gen-n* és *gen-d*. A *poss* és *gen* elnevezések abból adódnak, hogy a lexikai egységeket azonos sajátszó esetén is egyértelműen el kell különítenünk. Például a *Péter* sajátszó lehet alany és rövid birtokos is egy mondatban. Ehhez két lexikai egységet kell felvennünk, melyek közül az első egy nominatív esetben álló főnevet tartalmaz (ez lehet egy régens alanya), a

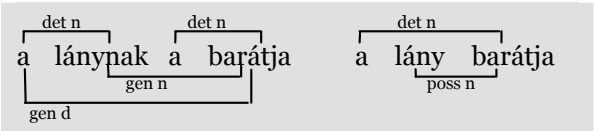
második pedig egy POSS „esetben” álló főnevet (ez lehet valaminek a rövid birtokosa). Ugyanez érvényes a hosszú birtokosra is, ahol a főnév GEN esetben áll.

```
lexitem("lány",ln("girl","NOM",arg(f("sg","3"),s("none","o","none"))))
lexitem("lány",ln("girl","POSS",arg(f("sg","3"),s("none","o","none"))))
lexitem("lánynak",ln("girl","NOM",arg(f("sg","3"),s("none","o","none"))))
lexitem("lánynak",ln("girl","GEN",arg(f("sg","3"),s("none","o","none"))))
```

25. példa: birtokos szerkezetek lexikai egységei

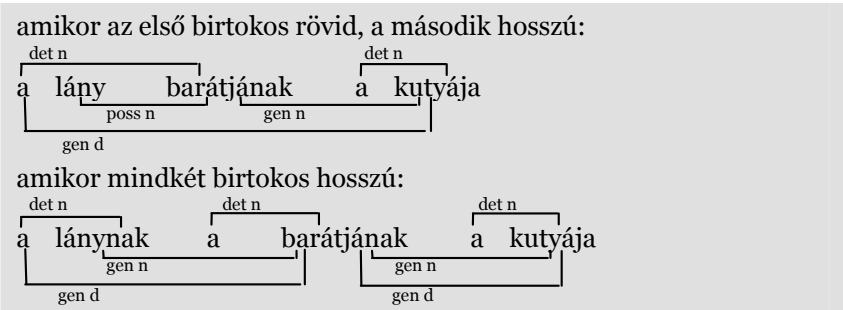
A birtokszó grammatikai viszonyai: saját névelőjével *det-n* viszonyt létesít, hosszú birtokosának főnévi pillérével *gen-n* viszonyt, valamint birtokosának determinánsi pillérével *gen-d* viszonyt. A tulajdonnév esetében mindkét pillért képes megtalálni a tulajdonnévben.

A rövid birtokos esetében nem ilyen egyszerű a helyzet, mivel a felszíni szerkezetben a rövid birtokos determináns nélküli főnévként jelenik meg, és igazodva a GASG alapgondolatához, miszerint a morfoszintaxisban csak azt írjuk le, ami a felszíni szerkezetben megjelenik, ezekben az esetekben a birtokszó nem keres *poss-d* viszonyt, csak *poss-n* viszonyt a főnévvel. (Erre később a szemantikai elemzés kapcsán visszatérek.) Kivételt képez a tulajdonnév, amely szintén egyszerre képviseli a determinánsi és főnévi pillért.



26. példa: birtokos relációk

A program tudja elemezni ezen felül a beágyazott birtokos szerkezeteket, melyeknek morfoszintaktikai elemzése a következő:



27. példa: beágyazott birtokos szerkezetek

A további két elvi lehetőséget, ahol mindkét birtokos rövid: **A lány barátja kutyája* és ahol az első birtokos hosszú, a második pedig rövid: **A lánynak barátja kutyája* agrammatikusnak ítéli.

A szemantikai elemzésnél a birtokos szerkezetek esetében a kiindulás legyen az, hogy a régensek főnévi argumentumai DP-k, vagyis két pilléren nyugszanak. DP-k esetében a referensek bevezetése triviális, mert rendelkezésre állnak a determinánsi elemek. A rövid birtokosok esetében azonban itt problémába ütközünk:

(12) a. egy [a fiú] barátja b. az [egy fiú] barátja
c. az [a fiú] barátja d. egy [egy fiú] barátja

Mind a négy eset egyértelműen rossz, ami azt jelzi, hogy a rövid birtokos számára a nyelv NP-t követel meg, de csak a forma szintjén, mert az interpretáció DP-t követel. Megoldási javaslatunk ezekre az esetekre a „hamis NP". Nézzük előbb a „valódi NP"-k problémáját:

(13) a. egy [fiút] bámuló lány b. a [fiút] bámuló lány

A melléknévi igenév vonzatai ezekben a mondatokban a forma szintjén is NP-k, valamint az interpretáció is NP-t mutat. A probléma az, hogy nincs determinánsi pillér, ami bevezethetne referenst. A köznév DRS leírása azonban nem tudhatja, hogy a köznév csupaszon lesz-e vonzat, ezért keresi a determinánsi pillért. Hogy megőrizhessük ezt az elképzelést, a régensnek *(bámuló)* kell úgy tennie, mintha lenne a tárgyi vonzatának determinánsi pillére. Vagyis ő fog bevezetni egy referenst a köznév számára, és ennek a referensnek a típusa „unspec", ami a vonzat NP-voltára utal.

A „hamis NP" kezelését úgy oldjuk meg, hogy a morfoszintaxisban eljátsszuk, hogy a vonzat NP kategóriájú, a szemantikában tehát a determinánsi pillért a régens nyújtja, azonban figyelembe kell venni, hogy a referens nem „unspec" típusú, mivel az interpretáció szerint nem NP, hanem „an" vagy „the" típusú, és éppen a birtokszó/jelzett szó által nyújtott determinánssal megegyező típusú.

4.4 Implementáció a morfémák szintjén

4.4.1 Morfofonológia

Az előző fejezetben ismertetett programváltozat bemutatta, hogy hogyan működik a totális lexikalizmus a szavak szintjén, vagyis teljesen inflektált szavak esetében. Azonban a magyar nyelv morfológiailag komplex szavai esetében a lexikai egységek számát minden egyes ragozott szóhoz külön-külön meg kell határozni, ami ezáltal gazdaságtalanul megnöveli a lexikon tartalmát. Például *n* számú intranzitív ige esetében a hat személyragos alakkal ez *6n* számú egységet jelent, holott minden igéhez ugyanazok a morfémák kapcsolódnak. Jobb megoldás tehát, ha a lexikai

egységeink nem ragozott szavak, hanem maguk a morfémák. Így a fenti példában szereplő *6n* egység *(n+6)*-ra csökken. A GASG ezen változata, a Totális Lexikalista Morfológia (TLM; Alberti–Balogh–Kleiber–Viszket 2002b) közvetlenül alkalmazza a totális lexikalizmus elvét.

A lexikonban tárolt egységek tehát ezentúl nem szavak, hanem morfémák, és a kölcsönös keresés a morfémák között zajlik.

```
lexi(m("","a","Z"),labstem("the",phonfst(1,3,3,3), 3,[])).
lexi(m("","fiú",""),labstem("boy",phonfst(2,1,2,2),1,[])).
lexi(m("","alm","H"), labstem("apple",phonfst(2,2,2,2),1,[])).
lexi(m("","l","Ó"),labstem("horse",phonfst(2,1,2,1),1,[])).
lexi(m("bok","O","r"), labstem("bush",phonfst(2,1,2,2),1,[])).
lexi(m("ny", "Á", "r"),labstem("summer",phonfst(2,2,2,1),1,[])).
lexi(m("","ül",""),labstem("sit",phonfst(1,1,1,2),2,[["NOM","LOC2"],["NOM"]])).
lexi(m("t","A","t"),labder("cause",phonfsu(2,2,0.2,2),2,ac(-1,0,1))).
lexi(m("h","A","t"),labsuff("may",phonfsu(1,1,1,2), 2, 1)).
lexi(m("","i",""),labsuff("sg3obj+def",phonfsu(1,3,1,3),2,3)).
lexi(m("V","t",""),labsuff("ACC",phonfsu(1,1,1,3), 1, 4)).
lexi(m("b","A","n"),labsuff("INESSIV",phonfsu(1,2,2,3), 1, 4)).
lexi(m("b","A",""),labsuff("ILLATIV",phonfsu(1,2,2,3), 1, 4)).
```

28. példa: lexikai egységek TLM-ben

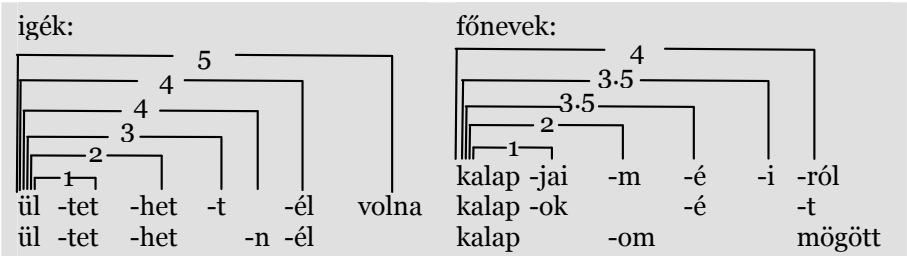
A lexikai leírások első helyén a korábbi elképzeléshez hasonlóan az egység sajátszava áll, de jelen esetben egy fonológiailag alulspecifikált változatban, ami lehetővé teszi a különböző alternációk kezelését. A legegyszerűbb eset, amikor a morféma nem változik, például az *ül* ige vagy az *-i* személyrag esetében. Más esetekben a morfémában az alternáció helyén egy változó szerepel, amely a környezeti feltételek szerint egy megadott módon realizálódhat. Például az „A” változó lehetséges realizációi: a/e, az „Á” változóé: á/a stb. A lexikai egységek következő tagja a *címkék* (labstem(...), labsuff(...), labder(...)), amelyek tartalmaznak fonológiai, morfológiai valamint szintaktikai követelményeket egyaránt.

A morfémákra épülő új programváltozat a bemeneti szósort először szavakra bontja, majd a lexikai egységeket használva meghatározza a szavakban szereplő morfémákat, és összeállít egy javasolt listát (numeráció), amin az elemzést végzi. Legelső lépésben a korábbi eljárással megegyezően beszámozza a morfémákat, de itt már kettős számozást alkalmaz, ami azt rögzíti, hogy a morféma melyik (a mondatban hányadik) szóban szerepel, és hányadik helyen. A *Péter rá-ül-tet-i a lány-t a szék-re* mondatban például a *-t* tárgyraghoz *n(4, 2, ...)* számozású egység fog tartozni. Az eljárás ebben a formában azért hasznos, mert egyrészt a két szám együttes megadásával tudunk utalni a konkrét morfémára, az első szám használatával viszont a szavakra és köztük lévő viszonyokra – szintén egyértelműen.

A fent létrehozott listán először fonológiai ellenőrzés történik. A lexikai egységek címkéinek második eleme fogalmazza meg minden esetben a fonológiai jegyeket. Ezek lehetnek saját magukra vonatkozó jegyek, illetve megadhatják azt, hogy az adott morféma milyen változást idéz elő az előtte vagy mögötte álló morfémában. A fonológiai jegyeket rendezett négyesekként adjuk meg, amelyek különbözőek a tövek és a szuffixumok esetében. A tövekhez tartozó jegyek: phonfst(elölség, kerekség, képzéshely, nyitó), a szuffixumok jegyei pedig phonsu(nyújt, rövidít, kivet, nyit). A lexikai egységekben ezeken a helyeken számok láthatóak, ahol 1=érvényes, 2=nem érvényes, 3=nem számít. Vagyis minden jegynek három különböző értéke lehet. A szuffixumoknál minden jegy arra vonatkozik, hogy az illető toldalék mit változtat a szótőn: nyújthat, rövidíthet, kivethet magánhangzót, vagy megváltoztathatja a nyíltsági fokát. A töveknél az első három jegy saját tulajdonságokat rögzít, ezeket a rájuk hivatkozó környezeti morfémák fogják használni, a negyedik jegy pedig azt tartalmazza, hogy nyitótőről van szó vagy sem.

A fenti jegyek segítségével kezelhetőek a mgh~∅ alternációk, a magánhangzó harmónia, a magánhangzó megnyúlása, illetve rövidülése. Ez alapján a program képes elemezni a *bokor-bokrot-bokorban*, *tűz-tűzzel-tüzeken* alakokat, és egyértelműen kiszűri a hibás **bokrban, *bokorben, *tűzet, *kalapakat* stb. alakokat.

A következő lépés a morfémák helyes szórendjének megállapítása a szavakon belül. Ez a szavak közötti immprec relációkhoz hasonlító rangparaméterek bevezetésével zajlik. Azonban itt egyszerűbb dolgunk van, mivel a morfémák között megfogalmazható egy egyértelmű lineáris rend. Az immprec rangok morfémák között a következők:



29. példa: morfémák sorrendje

Az igék esetében tehát a rangparaméterek által definiált morféma sorrend: igető – műveltetés – modalitás – igeidő – igemód – személy/szám – 'volna'. A főnevek esetében a probléma és a megoldás hasonló, de van különbség. A legszembetűnőbb a

tört rang[18] szerepeltetése, amelynek lényege, hogy itt az egyenlőség meg van engedve. A névutókkal ugyanúgy járunk el, mint a *volna* esetében, és bár a névutók a magyarban független szavak, a TLM rendszerben viselkedésük miatt esetjelölő szuffixumoknak tekintjük őket.

Ha az immprec relációk is ellenőrződtek a morfémák között, akkor már van egy listánk, amely morfoszintaktikailag hibátlan, a fonológiai és morfológiai követelmények ki vannak elégítve, és jó a morfémák sorrendje is a szavakon belül.

4.4.2 Szintaxis

A fent bemutatott elemzéssel/ellenőrzéssel létrehozott listával hívjuk meg a `synrel` predikátumot, ami a mondat szintaktikai elemzését végzi a benne szereplő morfémák alapján. Az elv hasonló a korábbi elképzeléshez, de döntő különbség az, hogy most nem a ragozott szavak, hanem a morfémák keresik egymást. A `synrel` predikátum meghívja a `synrelations` predikátumokat, vagyis a morfémalista minden eleme behívja a hozzá tartozó szintaktikai követelményeket. Ezzel létrehozunk egy új listát, amelyben a morfémák mellett mindenhol szerepel egy lista, amely az adott morféma által az adott mondatban létesített grammatikai relációkat tartalmazza. A grammatikai relációk leírása különbözik a korábbitól, egyrészt a morfémák kettős számozása, másrészt egységesítési törekvések miatt. A grammatikai relációk itt rendezett hetesek: *gr(X, Y, Z, N, M, K, L),* ahol azt rögzítjük, hogy mi, mit és milyen relációra hív meg. Az X változó az első helyen mindig annak az elemnek a kategóriája, amely keres, az Y változó a második helyen mindig az, amelyet X keres, a harmadik helyen álló Z változó pedig a közöttük létesített grammatikai reláció. Ezután négy szám következik, ahol N és M az X-el jelölt morfémához tartozó számok, K és L pedig Y-hoz. Ezen a ponton látszik tisztán, hogy miért is előnyös a kettős számozás. A grammatikai relációkat ugyanis a morfémák határozzák meg, de mivel minden morféma egy szóban szerepel, ezért a morfémák közötti relációk determinálják a szavak közti relációkat is. A fenti jelölés pedig alkalmas arra, hogy ezt az információt megkapjuk, hiszen N és K számok azokra a szavakra utalnak, amelyekben a morfémák szerepelnek. Tehát ha szavak közti relációkra akarunk hivatkozni, akkor ezt egyszerűen megtehetjük a következő módon: *gr(X, Y, Z, N, _, K, _),* ahol _ jelentése (Prolog formalizmus szerint) az, hogy oda bármi illeszthető, vagyis az értéke mellékes. A `synrelations`

[18] Megengedett például a *kalap-ok-é-i-é-t* alak, ahol azonban az *-i* birtoktöbbesítő megköveteli maga előtt az *-é* birtokjelet, így nem fogjuk elfogadni az olyan agrammatikus szavakat mint például a **kalap-ok-i-ról*.

predikátumok harmadik argumentumaként szerepel a számozott morfémalista, ezzel biztosítva, hogy végig az elemzett mondat keretein belül maradjunk.

A determinánsok keresnek egy főnevet és egy igét *gr("det", "noun", "free", N, M, K, L)* és *gr("det", "regent", "_", N, M, Q, Z)* relációkra.

```
synrelations(n(N,M,li(_,labstem(_,_,3,_))),
[gr("det","noun","free",N,M,K,L), gr("det","regent","_",N,M,Q,Z)],  NUMLIST  ):-
    detrel(N,M, K,L, NUMLIST), in_nlist( n(Q,Z,li(_,labstem(_,_,2,_))), NUMLIST).
```

30. példa: determinánsok szintaxisa

Determináns-főnév kapcsolat feltételeit tartalmazza a `detrel` predikátum, amely szerint a determináns minden esetben megelőzi az általa keresett főnevet.

A főnevek kezelése viszonylag egyszerű, hiszen ha nincs szuffixumuk, akkor keresnek egy régenst alanyi relációra[19]: *gr("noun", "regent", "subj", N, M, K, L)*. Az N, M számokkal jelölt főnévi szó megnézi, hogy van-e az N,_ számmal jelölt, vagyis ugyanahhoz a szóhoz tartozó szuffixum. Amennyiben nincs, akkor keresi a régenst. A régens felől a releváns információ a lexikai egység címkéje: labstem(_,_,2,_), amely azt mondja, hogy egy igei (2-es szám) tőről (labstem...) van szó. Amennyiben a főnév ragozott, a grammatikai relációkról maguk a szuffixumok gondoskodnak.

```
synrelations(n(N,M,li(_,labstem(_,_,1,_))),
[gr("noun", "regent", "subj", N,M, K,L)], NUMLIST):-
        not(in_nlist( n(N,_,li(_,labsuff(_,_,_,_))), NUMLIST)),
        in_nlist(n(K,L,li(_,labstem(_,_,2,_))), NUMLIST).
synrelations(n(N,_,li(_,labstem(_,_,1,_))),[], NUMLIST):-
        in_nlist(n(N,_,li(_,labsuff(_,_,_,_))), NUMLIST).
```

31. példa: főnevek szintaxisa

Az eddig említett mindkét esetben tövek kerestek környezeti szavakat – másik töveket –, de az igekötők esetében egy új reláció is bevezetődik. Az igekötők mint affixumok keresik egyrészt a tövet, amihez kapcsolódnak: *gr("pref", "stem", "free", N, M, N, X)*[20], másrészt egy esetragot *gr("regent", "noun", "obl", N, M, K, L)* viszonyra, pontosan azt a ragot, amelyet a saját címkéjükben szereplő szintaktikai követelmény kijelöl (pl.: labder("into",_,_, prt1("ILLATIV")) ...). Végül keresnek egy determinánst *gr("regent", "det", "obl", N, M, Q, Z)* viszonyra, és a keresett névelőnek `detrel` viszonyban kell állnia azzal a főnévvel, amelynek tövéhez az oblikvuszi viszonyra megtalált esetrag tartozik.

---

[19] Megjegyzendő, hogy a főnévi állítmány esetét még nem kezeljük, ahol szintén szuffixum nélküli főnév szerepel.

[20] a *free* vagyis *szabad* viszonynak majd a globális ellenőrzés fázisában lesz szerepe.

```
synrelations(n(N,M,li(m("","be",""),labder("into",_,_,CASE))),
[gr("pref","stem","free",N,M,N,X), gr("regent","noun","obl",N,M,K,L),
gr("regent","det","obl", N,M,Q,Z)], NUMLIST):-
        in_nlist( n(N,X,li(_,labstem(_,_,2,_))), NUMLIST),
        in_nlist( n(K,L,li(_,labsuff(CASE,_,1,_))), NUMLIST),
        stemrel(K,L, S,W, NUMLIST), detrel(Q,Z, S,W, NUMLIST).
```

32. példa: igekötők szintaxisa

A finit igetövek — attól függően, hogy az ige eredetileg milyen vonzatokat vár — *subj, obj, obl* viszonyokra keresnek egy-egy főnevet, illetve az ahhoz tartozó determinánst. Az előző mondat „eredetileg" kifejezése arra utal, hogy az igetőhöz járuló toldalékok is behozhatnak újabb vonzatokat, ezt láthatjuk például a műveltetésnél: az egyvonzatos *ül(NOM)* igéhez képest az *ül-tet(NOM, ACC)* már kétvonzatos, kiegészülve egy tárgyi vonzattal, amit a *-tAt* szuffixum hoz be.

```
synrelations(n(N,M,li(m("","ül",""),labstem("sit",_,2,_))),
[gr("regent","noun","subj",N,M,Q,Z), gr("regent","det","subj",N,M,V,W)], NUMLIST):-
        in_nlist(n(Q,Z,li(_,labstem(_,_,_,_))), NUMLIST),
        not(in_nlist(n(Q,_,li(_,labsuff(_,_,_,_))), NUMLIST)),
        detrel(V,W, Q,Z, NUMLIST).
```

33. példa: igék szintaxisa

A szuffixumok minden esetben keresik a hozzájuk tartozó tövet, grammatikai relációra pedig nem mindegyik keres környezeti morfémát. A *-hAt* igeképző például csupán a tövét keresi, más szintaktikai információt nem ad a mondathoz, az ő szerepe majd a szemantikában jön elő. A *-tAt* műveltető képzőből két fajta van, aszerint, hogy milyen igéhez kapcsolódnak. Az intranzitív igéhez kapcsolódó *-tAt* képző a fent említett módon hozzáad a vonzatkerethez egy tárgyi vonzatot, és a szintaxisban maga a *-tAt* morféma keres egy *-(V)t* tárgyragot. A másik variáns, amikor a *-tAt* képző tranzitív igéhez kapcsolódik, ekkor ugyanis egy *-vAl* ragos vonzatot hoz be, és azt is fogja keresni:

Péter beül a székbe. → Mari beül**tet**i Péter**t** a székbe.

Péter keresi a kabátját. → Mari keres**tet**i Péter**rel** a kabátját.

-tAt₁:

```
synrelations(n(N,M,li(m("t","A","t"),labder("cause",_,_,_))),
[gr("suff","stem","free",N,M,N,X),gr("regent","noun","obj",N,M,Q,Z)], NUMLIST):-
        in_nlist(n(N,X,li(_,labstem(_,_,_,[["NOM"]|L]))), NUMLIST),
        not(L="ACC"), in_nlist(n(Q,Z,li(_,labsuff("ACC",_,_,_))), NUMLIST).
```

-tAt₂:

```
synrelations(n(N,M,li(m("t","A","t"),labder("cause",_,_,_))),
[gr("suff","stem","free",N,M,N,X),gr("regent","noun","obj",N,M,Q,Z)], NUMLIST):-
        in_nlist(n(N,X,li(_,labstem(_,_,_,[["NOM","ACC"]]))), NUMLIST),
        in_nlist(n(Q,Z,li(_,labsuff("-val",_,_,_))), NUMLIST).
```

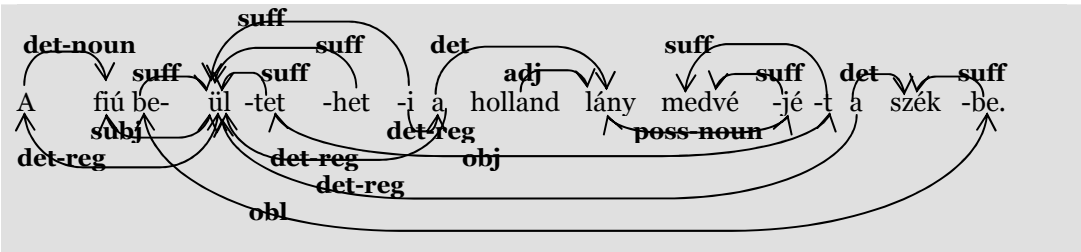
34. példa: a -tAt *képző szintaxisa*

Ugyanígy a főnévragok is keresik a főnévi tövet, valamint a *-bA* rag keresi a *be-* igekötőt, a *-rA* rag a *rá-* igekötőt, a *-(V)t* tárgyrag a *-tAt* képzőt. Amennyiben nem igekötős az ige vagy nem megfelelő az igekötő, akkor olyan igetövet keresnek, amely eredetileg *-bA*, *-rA* vagy tárgyragos vonzattal rendelkezik.

Miután minden morféma környezeti követelményei alapján megtalálta a keresett elemet/elemeket, összeáll a mondatban szereplő grammatikai relációk listája. Ezen a listán kell elvégeznünk a globális keresés ellenőrzését, ami a korábbi elképzelés szerint kivitelezhető. A *free* viszonyt létesítő elemek között nem szükséges a kölcsönösség, minden más viszonynál pedig ha van a listában egy *gr(X, Z, Y, N, _, K, _)* viszony, akkor szerepelni kell egy *gr(Z, X, Y, K, _, N, _)* viszonynak is.

A morfémák közötti kölcsönös keresést mutatja be a következő ábra:

(14) A fiú beültetheti a holland lány medvéjét a székbe.



35. példa: morfémák közötti keresés a (14) mondatban

4.4.3 Szemantika

A szemantikai elemzés alapfeladata változatlan, a lexikai egységek alapján meg kell határoznunk a mondat DRS-ét. A különbség most is az, hogy a szavak helyett a morfémák hozzák be a proto-DRS-eket. Lényeges továbblépés az, hogy a morfémaszinten jóval részletesebb elemzést tudunk adni, továbbá az, hogy a szemantikai leírást olyan módon kell megadni, hogy az egy egyszerű lépéssel beilleszthető legyen az LDRS (életfogytiglani DRS; Alberti 2000) elméletébe, ami már túllép a mondatelemzés keretein. Ezzel a módszerrel tehetjük képessé programunkat teljes szövegek elemzésére.

A determinánsok rendre bevezetnek egy referenst, amelyekről a többi elem állításokat tesz. A köznevek a referensüket attól a determinánstól kapják, amellyel a szintaktikai elemzésben *detrel* viszonyban álltak. Új elképzelés ezen a ponton annak beépítése, hogy a lexikai egységek azt is közlik, hogy a leírásukban említett referenseket az Életfogytiglani DRS világstruktúrájának mely világába kell beilleszteni, legalábbis egymáshoz képest.

Az igék szintén bevezetik a maguk szituációs referensét, és állítást tesznek adott referensekről. Érdekesebb kérdés a *-tAt* és *-hAt* képzők esete, amelyek bevezetik saját „event” referensüket, majd állítást tesznek egy referensről valamint egy másik szituációról. Konkrétan a *-tAt* képző például azt fogalmazza meg, hogy van egy esemény, ami a következő: valaki végeztet egy cselekvést egy másik személlyel, és ez a cselekvés éppen az, amit az a morfémasor ír le, amihez a képző járult. Ugyanez érvényes a *-hAt* képzőre is. Nézzük meg ezt szemléletesen a következő példán. A bevezetett proto-DRS-ekben az összefüggéseket, vagyis azonosságokat a nyilak szemléltetik:

be-	into(r11, r12)	
ül	sit(ev20; r21)	„r21 ül” esemény, amit ev20 jelöl
-tet	cause(ev30; r31, ev32)	r31 okozza ev32-t, ami éppen ev20=„r21 ül”
-het	may(ev40; r41, ev42)	r31 megteheti ev32-t, ami éppen ev32=” r31 okozza, hogy=„r21 ül”
-ed	you.sg=r51	

35. példa: DRS-építés morfémákkal

Az igekötők esetében két lépésben alakul ki a proto-DRS. Az igekötők információt adnak hozzá az ige által bevezetett DRS-hez. Ige és igekötő szoros kapcsolata miatt e kettő DRS-eiben lévő információt egyesíteni kell, és a továbbiakban egyként kezelni. Például a *be-* igekötő proto-DRS-e azt mondja, hogy az *into(r1)* információt építse be a következő morféma (az igető) által bevezetett proto-DRS-be. Ezt a műveletet azonban csak akkor lehet elvégezni, ha befejeztük a szemantikai információk, a proto-DRS-ek összegyűjtését, hiszen ekkor áll rendelkezésünkre az az információ, hogy mit hozott be az igető. A művelet során az *into(r1)* és *sit(ev20, r21)* DRS-ekből a *sit_into(ev; r20, r1)* DRS áll elő.

A szemantikai reprezentáció felépítését szemlélteti a következő példa:

(15) Egy fiú beülteti a holland lányt a székbe.

determináns: provref("old",[r(1,1,1)])
provref("<or=",[r(1,1,1),e(3,3,1)])

köznév: pred("boy",[r(1,1,1)])

finit ige: pred("sit_into",[e(3,2,1),r(4,1,1),r(7,1,1)])
provref("new",[e(3,2,1)])
provref("=",[e(3,3,1),e(3,2,1)])

-tAt képző: provref("new",[e(3,3,1)])
provref("<",[e(3,4,1),e(3,3,1)])
pred("cause",[e(3,3,1),r(1,1,1),e(3,2,1)])
determináns: provref("old",[r(4,1,1)])
provref("<or=",[r(4,1,1),e(3,3,1)])
melléknév: pred("Dutch",[r(4,1,1)])
köznév: pred("girl",[r(4,1,1)])
determináns: provref("old",[r(7,1,1)])
provref("<or=",[r(7,1,1),e(3,3,1)])
köznév: pred("chair",[r(7,1,1)])

A determinánsok rendre bevezetnek az alapvilágba („old") egy referenst, valamint meghatározzák, hogy a bevezetett referens melyik világokból érhető el, ezt jelzi a „<or=" reláció. A köznevek és a melléknevek állítást tesznek egy már bevezetett referensről. A finit ige (itt az ül) bevezet az „új" világba egy szituációs referenst, majd együtt az igekötővel állítást tesz egy vagy több referensről *(pred("sit_into", [e(3,2,1), r(4,1,1), r(7,1,1)]))*. A konkrét példában az e(3,2,1) szituáció az, hogy az r(4,1,1) referens beül az r(7,1,1)-be; r(4,1,1)-et és r(7,1,1)-et később azonosítjuk a lánnyal illetve a székkel. A *-tAt* képző szintén bevezet egy szituációs referenst az „új" világba, majd meghatározza az e(3,4,1) és e(3,3,1) közötti rendezési relációt, és állítást tesz egy referensről: az e(3,3,1) szituáció az, hogy r(1,1,1) referens okozza e(3,2,1) szituációt, ami a korábbiakban leírt „beülés" esemény.

4.4.4 Példák programfutásokra

Végül nézzük meg, hogy mindez hogy néz ki a gyakorlatban. Ennek szemléltetésére álljon itt két példa a programfutások közül. A példákban a `goal` címszó mellett mindig a bemeneti szósort látjuk, ami meghívja a grammatikalitásellenőrző predikátumot, és működésbe hozza az elemzőt. Majd a következő sorban láthatjuk a programfutás eredményét, ahol legelöl a mondat morfémái szerepelnek, majd a szintaktikai elemzés, végül pedig szemantikai elemzésként a mondat DRS-e.

```
goal A fiú kerestet egy medvét a pékkel.

n(1,1,li(m("","a","Z"),labstem("the",phonfst(1,3,3,3),3,[])))

n(2,1,li(m("","fiú",""),labstem("boy",phonfst(2,1,3,2),1,[])))

n(3,1,li(m("","keres",""),labstem("look-for",phonfst(1,2,2,2),2,[["NOM","ACC"]])))
n(3,2,li(m("t","A","t"),labder("cause",phonfsu(2,2,0.2,2),2,ac(-1,0,1))))

n(4,1,li(m("","egy",""),labstem("a(n)",phonfst(2,3,3,3),3,[])))

n(5,1,li(m("","medv","H"),labstem("bear",phonfst(1,2,2,2),1,[])))
n(5,2,li(m("V","t",""),labsuff("ACC",phonfsu(1,1,1,3),1,4)))

n(6,1,li(m("","a","Z"),labstem("the",phonfst(1,3,3,3),3,[])))

n(7,1,li(m("","pék",""),labstem("baker",phonfst(1,2,3,2),1,[])))
n(7,2,li(m("S","A","l"),labsuff("INSTR",phonfsu(1,2,2,3),1,4)))

SZINTAXIS:

gr("det","noun","free",1,1,2,1)
gr("det","regent","_",1,1,3,1)
gr("noun","regent","subj",2,1,3,1)
gr("regent","noun","subj",3,1,2,1)
gr("regent","det","subj",3,1,1,1)
gr("regent","noun","obj",3,1,5,2)
gr("regent","det","obj",3,1,4,1)
gr("suff","stem","free",3,2,3,1)
gr("regent","noun","obl",3,2,7,2)
gr("regent","det","obl",3,2,6,1)
gr("det","noun","free",4,1,5,1)
gr("det","regent","_",4,1,3,1)
gr("suff","stem","free",5,2,5,1)
gr("noun","regent","obj",5,2,3,1)
gr("det","noun","free",6,1,7,1)
gr("det","regent","_",6,1,3,1)
gr("suff","stem","free",7,2,7,1)
gr("noun","regent","obl",7,2,3,2)

SZEMANTIKA:

provref("old",[r(1,1,1)])
provref("<or=",[r(1,1,1),e(3,2,1)])
pred("boy",[r(1,1,1)])
provref("new",[e(3,1,1)])
provref("=",[e(3,1,1),e(3,1,1)])
pred("look-for",[e(3,1,1),r(6,1,1),r(4,1,1)])
provref("new",[e(3,2,1)])
provref("<",[e(3,3,1),e(3,2,1)])
pred("cause",[e(3,2,1),r(1,1,1),e(3,1,1)])
provref("old",[r(4,1,1)])
provref("<or=",[r(4,1,1),e(3,2,1)])
pred("bear",[r(4,1,1)])
provref("old",[r(6,1,1)])
provref("<or=",[r(6,1,1),e(3,2,1)])
pred("chair",[r(6,1,1)])
yes
```

```
goal  Az okos fiú beültetheti a büszke medvéjét a székembe.

n(1,1,li(m("","a","Z"),labstem("the",phonfst(1,3,3,3),3,[])))

n(2,1,li(m("","okos",""),labstem("clever",phonfst(2,1,2,1),4,[])))

n(3,1,li(m("","fiú",""),labstem("boy",phonfst(2,1,3,2),1,[])))

n(4,1,li(m("","be",""),labder("into",phonfsu(1,2,-1,2), 2,prt1("ILLATIV"))))
n(4,2,li(m("","ül",""),labstem("sit",phonfst(1,1,1,2),2,[["NOM","LOC2"]])))
n(4,3,li(m("t","A","t"),labder("cause",phonfsu(2,2,0.2,2),2,ac(-1,0,1))))
n(4,4,li(m("h","A","t"),labsuff("may",phonfsu(1,1,1,2),2,1)))
n(4,5,li(m("","i",""),labsuff("sg3obj+def",phonfsu(1,3,1,3),2,3)))

n(5,1,li(m("","a","Z"),labstem("the",phonfst(1,3,3,3),3,[])))

n(6,1,li(m("","büszke",""),labstem("proud",phonfst(2,1,3,1),4,[["SUBLAT"]])))

n(7,1,li(m("","medv","H"),labstem("bear",phonfst(1,2,2,2),1,[])))
n(7,2,li(m("J","É",""),labsuff("poss(s)he",phonfsu(1,1,1,1),1,2)))
n(7,3,li(m("V","t",""),labsuff("ACC",phonfsu(1,1,1,3),1,4)))

n(8,1,li(m("","a","Z"),labstem("the",phonfst(1,3,3,3),3,[])))

n(9,1,li(m("","szék",""),labstem("chair",phonfst(1,2,3,2),1,[])))
n(9,2,li(m("V","m",""),labsuff("possI",phonfsu(1,1,1,1),1,2)))
n(9,3,li(m("b","A",""),labsuff("ILLATIV",phonfsu(1,2,2,3),1,4)))

SZINTAXIS:

gr("det","noun","free",1,1,3,1)
gr("det","regent","_",1,1,4,2)
gr("adj","noun","free",2,1,3,1)
gr("noun","regent","subj",3,1,4,2)
gr("pref","stem","free",4,1,4,2)
gr("regent","noun","obl",4,1,9,3)
gr("regent","det","obl",4,1,8,1)
gr("regent","noun","subj",4,2,3,1)
gr("regent","det","subj",4,2,1,1)
gr("suff","stem","free",4,3,4,2)
gr("regent","noun","obj",4,3,7,3)
gr("suff","stem","free",4,4,4,2)
gr("suff","stem","free",4,5,4,2)
gr("regent","det","obj",4,5,5,1)
gr("det","noun","free",5,1,7,1)
gr("det","regent","_",5,1,4,2)
gr("adj","noun","free",6,1,7,1)
gr("suff","stem","free",7,2,7,1)
gr("suff","stem","free",7,3,7,1)
gr("noun","regent","obj",7,3,4,5)
gr("det","noun","free",8,1,9,1)
gr("det","regent","_",8,1,4,2)
gr("suff","stem","free",9,2,9,1)
gr("suff","stem","free",9,3,9,1)
gr("noun","regent","obl",9,3,4,1)
```

```
SZEMANTIKA:

provref("old",[r(1,1,1)])
provref("<or=",[r(1,1,1),e(4,4,1)])
pred("clever",[r(1,1,1)])
pred("boy",[r(1,1,1)])
pred("sit_into",[e(4,2,1),r(5,1,1),r(8,1,1)])
provref("new",[e(4,2,1)])
provref("=",[e(4,3,1),e(4,2,1)])
provref("new",[e(4,3,1)])
provref("<",[e(4,4,1),e(4,3,1)])
pred("cause",[e(4,3,1),r(1,1,1),e(4,2,1)])
provref("fixpoint",[e(4,4,1)])
pred("may",[e(4,4,1),r(1,1,1),e(4,3,1)])
provref("old",[r(5,1,1)])
provref("<or=",[r(5,1,1),e(4,4,1)])
pred("proud",[r(5,1,1)])
pred("bear",[r(5,1,1)])
pred("owns",[id("(s)he"),r(5,1,1)])
provref("old",[r(8,1,1)])
provref("<or=",[r(8,1,1),e(4,4,1)])
pred("chair",[r(8,1,1)])
pred("owns",[id("I"),r(8,1,1)])
yes
```

Összegzés

Dolgozatomban bemutattam, hogy a generatív nyelvleírásban egyre nagyobb szerep jut a lexikonnak elutasítva a transzformációs nyelvtanok elképzelését. Röviden vázoltam a legsikeresebb nem transzformációs elméletek (Kategoriális Nyelvtan, LFG, HPSG) egy-egy alapgondolatát. Ezek jelentik az önálló kutatásom alapját képező radikális lexikalizmus előzményét.

A dolgozat döntő része a kutatás eredményének bemutatása, amely a totális lexikalizmus jelenleg magyar nyelvre kidolgozott, Prolog nyelven írt számítógépes implementációja[21]. A felhasznált nyelvelméleti keret a GASG (Generative Argument Structure Grammar; Alberti 1999), amelynek legfontosabb tulajdonsága a radikális lexikalizmus, vagyis minden információ a lexikonban tárolódik, és ezen információk alapján épülnek össze a szavak mondattá. A GASG alapgondolata, hogy a lexikonban tárolt lexikai egységek gazdag leírása megadja az adott szó saját tulajdonságait, valamint követelményeket fogalmaz meg az általa keresett környezeti szavakról. Ezen információk alapján a lexikai egységek kölcsönösen keresik egymást. A GASG lexikai egységei rendezett négyesek: <sajátszó, szintaxis, szemantika, szintaxis és szemantika közti kapcsolat>, amelyben a sajátszavak a mondatban szereplő teljesen

[21] A korábbi eredményeket bemutató munkák: Alberti-Balogh-Kleiber (2002), Alberti-Balogh-Kleiber-Viszket (2002a), Balogh (2001), Balogh-Kleiber (2001).

ragozott alakok. A kutatási projekt jelenleg egy döntő fordulat után van: a ragozott szavakra épülő lexikai egységek helyett ezentúl a szavakat felépítő morfémákhoz rendelünk leírást[22].

A program jelentősége egyrészt abban ragadható meg, hogy hátterében egy elméletileg tiszta generatív grammatika áll, így a program minden egyes lépése legitimált. Másik nagy eredmény a szintaxis és szemantika párhuzamos kezelése, amellyel megvalósíthatóvá vált a szemantikai reprezentáció közvetlen megalkotása a morfoszintaktikai reprezentációból.

A program jelenleg a magyar nyelvnek csak egy kis fragmentumáról tud számot adni, a jelenlegi fázisban semleges, egyszerű magyar mondatokat képes morfoszintaktikailag és szemantikailag elemezni. A kutatás tehát ezen a ponton nem állhat meg, a közeljövő feladatai közé tartozik többek között a nem semleges mondatok kezelése, a megfelelő operátorok bevezetésével valamint az összetett mondatok és idiómák bevezetése.

A program lehető legprecízebb kidolgozása a cél. Egyrészt a rendszer több nyelvre való kidolgozásával elérhetővé válhat a számítógépes fordítás lexikalista alapokon. A jelenlegi elképzelés szerint a számítógépes fordítás az elméletünk legsikeresebbnek tűnő részén keresztül, a DRS-en keresztül válhat megvalósíthatóvá, amelyet egy angol anyanyelvű szemanta akár már most meg is ért. Ehhez a feladat tehát visszafelé megteremteni a DRS-től egy idegen nyelvű mondatig vezető utat. Másrészt egy megbízhatóan működő programmal igazolhatóvá válik a GASG – és vele együtt a totális lexikalizmus – elméleti elképzeléseinek létjogosultsága.

[22] Alberti-Balogh-Kleiber-Viszket (2002b).

Irodalom

Alberti G. (1997): Restrictions on the Degree of Referentiality of Arguments in Hungarian Sentences, in: *Acta Linguistica Hungarica* 44 (3-4).

Alberti G. (1999): GASG: The Grammar of Total Lexicalism, in: *Working Papers in the Theory of Grammar 6/1*, Theoretical Linguistics Programme, Budapest University (ELTE) and Research Institute for Linguistics, Hungarian Academy of Sciences.

Alberti G. (2000): Lifelong Discourse Representation Structures. in: *Gothenburg Papers in Computational Linguistics,* 00-5, 13-20.

Alberti, G. – K. Balogh – J. Kleiber (2002): GeLexi Project: Prolog Implementation of a Totally Lexicalist Grammar, in: de Jongh, D. – H. Zeevat – M. Nilsenová (eds.): *Proceedings of the Third and Fourth Tbilisi Symposium on Language, Logic and Computation,* ILLC, Amsterdam, and Univ. of Tbilisi.

Alberti G. – Balogh K. – Kleiber J. – Viszket A. (2002a): A totális lexikalizmus elve és a GASG nyelvtan-modell, megj. előtt: Szeged, SZTE.

Alberti, G. – K. Balogh – J. Kleiber – A. Viszket (2002b): Totally Lexicalist Morphology, talk at the Sixth International Conference on the Structure of Hungarian, 10-12 September 2002, Düsseldorf.

Austin, P. K. (2001): Lexical Functional Grammar, in: Smelser, N. J. – P. Baltes: *International Encyclopedia of the Social Behavioral Sciences*, Elsevier Sciences.

Balogh K. (2001): GASG-keretben ábrázolt birtokos szerkezetek szintaxisának és szemantikájának Prolog-implementációja, előadás: PhD-konferencia, Szeged, 2001. nov. 8–9.

Balogh K. – Kleiber J. (2001): Egy lexikalista nyelvtan morfoszintaxisának Prolog-implementációja, OTDK pályamunka.

Bresnan, J. (2000): *Lexical-Functional Syntax*, Blackwell Publishers, Oxford.

Calder, J. – E. Klein – H. Zeevat (1988): Unification Categorial Grammar: a concise, extenable grammar for natural language processing, in: Proceedings of the 12th International Conference on Computational Linguistics, 22-27 Augsut 1988, Budapest.

Chomsky, N. (1957): *Syntactic Structures,* The Hague, Mouton. [Magyarul: in: Antal L. (1986, szerk.), *Modern nyelvelméleti szöveggyűjtemény* **VI/1**. Ford.: Csongrády Zs., Lázár P., Nádasdy Á.]

Chomsky, N. (1995, ed.): *The Minimalist Program*, MIT Press, Cambridge, Mass.

Dömölki B. (1986): Logikai programozás, in: *Tudomány*, *Számítógép-szoftver* c. különszám, 1986, 136-141.

van Eijck, J. – H. Kamp (1997): Representing Discourse in Context, in: van Benthem, J. – A. ter Meulen (eds.): *Handbook of Logic and Language*, Elsevier, Amsterdam, MIT Press, Cambridge, Mass.

Gazdar, G. – E. Klein – G. Pullum – I. Sag (1985): *Generalized Phrase Structure Grammar*, Blackwell Publishers, Oxford.

Joshi, A. K. – L. Levy – M. Takahashi (1975): Tree adjunct grammars, in: *Journal of the Computer and System Sciences* **10,** No. 1., 136—163.

Kamp, H – U. Reyle (1993): *From Discourse to Logic.* Kluwer, Dordrecht.

Karttunen, L. (1986): Radical Lexicalism, *CSLI Report 86-68*, Stanford.

Kiefer F. (2002): A nyelvészet és a számítástudomány, in: *Természet Világa*, 133. évfolyam, 8. szám.

Kis Balázs (1997): Mi van a szavakon túl?, előadás: VII. Országos Alkalmazott Nyelvészeti konferencián, Budapest, 1997. április.

Komlósy A. (2001): *A Lexikai-Funkcionális Grammatika mondattanának alapfogalmai*, Tinta Könyvkiadó, Budapest.

König, E. (1996): Introduction to Categorial Grammar, Universität Stuttgart.

Márkusz Zs. (1988): *Prologban programozni könnyű*, Novotrade, Budapest.

Partee, B. H. – A. G. B. ter Meulen – R. P. Wall (1990): *Mathematical Methods in Linguistics*, Kluwer, Dordrecht.

Prószéky G. (1989): *Számítógépes nyelvészet*, Számítástechnika-alkalmazási Vállalat, Budapest

Russel, S. J. – P. Norvig (2000): *Mesterséges intelligencia*, Panem Könyvkiadó, Budapest.

Sag, I. – C. Pollard (1987): Head-Driven Phrase Structure Grammar: An Informal Synopsis. *CSLI Report 87-79,* Stanford.

Szécsényi P. Tibor (1997): A magyar mondatszerkezet leírása HPSG keretben, in: *Néprajz és Nelvtudomány* XXXVIII, Szeged.

Uszkoreit, H. (1986): Categorial Unification Grammars, in: *Proceedings of the 11th International Conference on Computational Linguistics*, 25-29 Augsut 1986, Bonn.

Zeevat, H. (1989): A Compositional Approach of Discourse Representation Theory, in: *Linguistics and Philosophy* 12.

Zeevat, H. (1991): *Aspects of Discourse Semantics and Unification Grammar*, PhD Thesis, Amsterdam.

Melléklet

Prolog fogalomtár

argumentum (argument)

Lásd struktúra.

argumentumszám, aritás (arity)

Egy eljárás vagy egy struktúra argumentumainak a száma.

atom (atom)

Egy névkonstans. A kisbetűvel kezdődő és alfanumerikus karakterekkel folytatódó sorozatot, a csupa „speciális" karakterből álló sorozatot és az aposztrófok közé zárt tetszőleges sorozatot atomnak tekinti a rendszer. Az atomok egyesítése gyors, ábrázolásuk tömör, viszont drága a létrehozásuk és szétszedésük.

atomi kifejezés (atomic term)

Lásd konstans.

behelyettesítés (binding)

Amikor egy változót egyesítünk valamivel (akár egy másik változóval is), akkor azt mondjuk, hogy a változót behelyettesítettük.

beépített eljárás (built-in predicate)

Egy rendszer által definiált reláció.

cél (goal)

Egy eljáráshívás. Egy cél vagy sikerül, vagy meghiúsul. Egy cél többféleképpen is sikerülhet.

deklaráció (declaration)

A fordítónak szóló állítások, amelyek a program értelmezését befolyásolják. Példa: operátor-deklaráció.

determinisztikus (determinate)

Egy hívás determinisztikus, ha legfeljebb egyféleképpen sikerülhet. Egy eljárás determinisztikus, ha tetszőleges hívása determinisztikus.

dinamikus eljárás (dynamic predicate)

Olyan eljárás, amelyhez futási időben adhatunk és amelyből futási időben vehetünk el klózokat.

diszjunkció (disjunction)

Vagy kapcsolat. Prologban a vagy kapcsolatban levő relációkat pontosvesszővel elválasztva egymás után írjuk, és általában zárójelekkel vesszük körbe.

egyesítés (unification)

Kifejezések mintaillesztéssel történő azonos alakra hozása.

egyszerű kifejezés (simple term)

Egy nem struktúra kifejezés, azaz változó vagy konstans.

eljárás (procedure)

Egy beépített vagy egy felhasználó által definiált eljárás.

értékadás (instantiation)

Amikor egy változót egy nem változóval egyesítünk, akkor az értéket kap. (Az értékadás lehet részleges is, amennyiben a kifejezés, amivel egyesítettünk nem tömör.)

értékkel bíró (instantiated)

Egy változó értékkel bír, ha egyesítettük egy nem változó kifejezéssel.

értékkel nem bíró (unbound)

Egy változó, amit még soha nem egyesítettünk változótól különböző kifejezéssel.

fej (head)

Egy klóz következmény része.

felhasználó által definiált eljárás (user defined procedure)

Mindazon klózok összessége, amelyek feje adott nevű és argumentumszámú.

fordítás (compile)

Az a folyamat, amikor a program szövegéből lefordított alakot állítunk elő. A lefordított alak gyorsabban fut, mint az interpretált alak.

funktor (functor)

Egy struktúra nevéből és argumentumszámából képzett pár. Írásban általában / jellel elválasztva írjuk. Például a rendező('Luis Buńuel', 1900, 1983) struktúra funktora rendező/3.

füzér (string)

Karakterkódok listája.

indexelés (indexing)

Arra szolgál, hogy egy adott hívás esetén a rendszer ne próbálkozzon olyan klózokkal, amelyek semmiképp nem sikerülhetnek. Az indexelést a rendszer automatikusan végzi.

kampó eljárás (hook predicate)

Olyan eljárás, amit a felhasználó definiál, de közvetlenül a rendszer hívja meg.

kifejezés (term)

A Prologban előforduló adatfajták gyűjtőneve.

klóz (clause)

Egy tényállítás vagy egy szabály.

konjunkció (conjunction)

És kapcsolat. Prologban az és kapcsolatban levő relációkat vesszővel elválasztva egymás után írjuk.

konstans (constant)

Egy egész vagy egy lebegőpontos szám, vagy egy atom. Az atomic/1 beépített eljárás pontosan ezekre igaz.

lista (list)

Vagy a '[]' (nil) atom, vagy egy '.'/2 struktúra, melynek második argumentuma egy lista. Ha egy lista vége nem nil, hanem változó, akkor nyitott végű listáról beszélünk.

összetett kifejezés (compound term)

Lásd struktúra.

parancs (directive)

Fordítási időben végrehajtandó Prolog hívás.

predikátum (predicate)

Lásd eljárás.

program (database)

A felhasználó által definiált eljárások halmaza.

pár (pair)

A '-'/2 struktúra. Általánosan bármely kétargumentumú struktúrát nevezhetünk párnak, de ilyenkor meg kell mondani mi a neve.

redukciós lépés (reduction step)

A végrehajtásnak az a lépése, amikor egy hívást egy illeszkedő fejű klóz törzsével helyettesítünk.

rekurzió (recursion)

Egy eljárás rekurzív, ha végrehajtásához szükség van önmaga (esetleg közvetett) meghívására.

struktúra (structure)

Egyfajta Prolog kifejezés. A strúktúráknak van egy neve és egy vagy több argumentuma. Például a személy('Weöres Sándor', 1913, 1989) struktúra neve személy és három argumentuma van. A compund/1 eljárás pontosan ezekre igaz.

szabály (rule)

Egy klóz egy vagy több feltétellel. Egy szabály azt fejezi ki, hogy a törzséből következik a feje. Például a jókedvű(attila) :- kertész(attila). szabály azt fejezi ki, hogy ha Attila kertész, akkor jókedvű. Változókat használva általánosabb szabályokat is felírhatunk:

jóhírű(X) :- tejet_iszik(X), pipázik(X).

Ezzel azt állítjuk, hogy minden pipázó és tejivó X-re igaz, hogy jó a híre.

tényállítás (fact)

Egy klóz, amelynek nincs feltétel része, azaz a törzse üres. Egy tényállítás azt fejezi ki, hogy az adott reláció fennáll az argumentumaira. Példák tényállításokra (és lehetséges interpretációjukra):

király(henrik, anglia). /*Henrik Anglia királya volt.*/

csőre_van(madár). /* A madaraknak van csőrük.*/

alkalmazott(lilla, adatfeldolgozás, 55000). /*Lilla az adatfeldolgozási osztályon dolgozik, és keresete 55000.*/

törzs (body)

Egy szabály feltétel része.

visszalépés (backtracking)

Ha egy egyszer már kielégített célt más módon újra megpróbálunk kielégíteni, akkor visszalépésről beszélünk.

változó (variable)

Egy név ami egy (esetleg ismeretlen) értéket jelöl a programban. Prologban a változókat nagybetűvel vagy aláhúzással kezdődő és alfanumerikus karaktersorozattal folytatódó szöveggel jelöljük.