Pécsi Tudományegyetem Bölcsészettudományi Kar

Nyelvtudományi Doktori Iskola

Alkalmazott Nyelvészet Program

A TOTÁLIS LEXIKALIZMUS ELMÉLETÉTŐL A KÍSÉRLETI IMPLEMENTÁCIÓIG

PhD értekezés

Kleiber Judit

Témavezető: Dr. Alberti Gábor

Pécs, 2008.

Tartalomjegyzék

1 Bevezetés

1.1 A dolgozat tárgya

Dolgozatomban egy négyéves számítógépes nyelvészeti kutatás tanulságairól számolok be. Célunk az volt, hogy kipróbáljunk egy új típusú grammatikát elméletben és gyakorlatban egyaránt. Nyelvtanunk megvalósítja a „totális" lexikalizmust, ami a végsőkig fokozása az elmúlt évtizedekben megfigyelhető hangsúlyeltolódásnak szintaxis és lexikon között. Minden információt a szótári komponens tartalmaz, így nincs szükség frázisstruktúra-szabályokra. Az egyetlen művelet az *unifikáció*, amely a mondatok összeépítésének a motorja.

Az elmúlt években a nyelvtan implementációján dolgoztunk, készült egy elemző Prolog programnyelven, amely kis adatbázison magyar és angol mondatokat elemez, hozzájuk szemantikai reprezentációt társít, illetve köztük a gépi fordítást is megvalósítja (GeLexi projekt). Miután úgy tűnt, a mechanizmusok működnek, hatékonyabb adattároláson kezdtünk dolgozni (SQL adatbázis), hogy lássuk, hogyan birkózik meg a totális lexikalizmus nagyobb mennyiségű lexikai egységgel (LiLe projekt). Az eredményeink bíztatóak, de adatbázisunk mérete még nem érte el azt a szintet, hogy az ismert módszerekkel értékelni lehessen, ezért az újabb négyéves kutatás célja tovább haladni ezen az úton, és kipróbálni a totálisan lexikalista megközelítést olyan méretű adatbázison, amely már megmutathatja mechanizmusunk hatékonyságát is (ReALIS projekt).

Kutatásunk az elméleti és a számítógépes nyelvészet határán helyezkedik el. A tisztán elméleti nyelvészek hiányolják belőle az egyes nyelvi jelenségek sokkal részletesebb vizsgálatát és a kezelési mód választásának alátámasztását. Továbbá úgy gondolják, ha minden adatot a lexikonban tárolunk (nincsenek általános szabályaink), akkor a nyelvtanunk semmitmondó (ami nem áll, hiszen az univerzális elméleti keret kidolgozásához szükséges a nyelvi általánosítások minél alaposabb megragadása). A tisztán számítógépes nyelvészek pedig hiányolják a rendszer kiértékelését, mennyire pontos, mennyire hatékony, mekkora a lefedettsége, és gyakran kérdezik, miért nem egy már meglévő rendszerben készítjük el az elemzőnket. A dolgozat felépítésére is jellemző ez a kettősség: bizonyos fejezetek elméleti kutatásokról számolnak be, amelyek kapcsán talán fel sem merül az implementálhatóság kérdése; míg mások olyan nyelvtechnológiai alkalmazásokat mutatnak be, amelyek sokszor – akár bármiféle nyelvészeti ismeret felhasználása nélkül – csupán statisztikai alapon elemzik a szövegeket. Vannak azonban olyan elméletek, amelyek formalizálhatóságát implementációkkal bizonyítják, és olyan számítógépes nyelvészeti fejlesztések is, amelyek szigorú elméleti alapokon épülnek. Ezekről fogok több szót ejteni, mert a mi célunk is ez: első lépésként inkább elméleti („kísérleti"), az implementációval a nyelvtan egzaktságát bizonyítani; majd ha ez sikerült, következő lépésként azon dolgozni, hogy a program minél hatékonyabb legyen, és a nyelvtechnológia kihívásainak is meg tudjon felelni.

1.2 Motiváció

Elméleti kiindulópontunk egy dinamikus szemantikai elmélet volt, a DRT (Discourse Representation Theory, van Eijck–Kamp 1997), amely nem csupán mondatok, hanem szövegek szemantikai elemzésére is alkalmas. Alapötlete, hogy nem a mondatokat, hanem a hallgatói információállapot változását reprezentálja, a korábbi információállapotokat tekintve kontextusnak. Ennek egy továbbfejlesztett változata alkotja a nyelvtan és az implementáció alapját, egy szintén dinamikus szemantikai elmélet, az LDRT (Lifelong Discourse

Representation Theory, Alberti 2000), ahol a diskurzusreprezentációs struktúrák hallgatókhoz vannak rendelve, és amelyeket az interpretálók élethosszig építenek.

A totálisan lexikalista nyelvtan (GASG, Generatív Argumentumstruktúra Grammatika, Alberti 1999) megalkotását a Frege-féle kompozicionalitási elv betartásának igénye motiválta, vagyis hogy a szintaxis és a szemantika között egyértelmű megfeleltetés (homomorfizmus) legyen létesíthető.

A kutatók sokáig kerestek egy chomskyánus szintaxissal kompozicionális szemantikát, de tökéletes sikert nem értek el. Montague szemantikája (Montague 1970) kompozicionálisnak tekinthető ugyan, de egy nagyon erős eszközt (lambda-absztrakciót) alkalmaz, amely segítségével a szemantikát bármilyen rendszerrel izomorffá lehetne tenni, és amely olyan alakzatokat is létre tud hozni, amilyenek biztosan nem fordulnak elő egyetlen nyelvben sem. A DRT ezzel szemben egy sokkal „nyelvközelibb" szemantikai elmélet, amely viszont nem kompozicionális Chomsky hierarchikus szintaxisával (illetve szintén csak a λ-absztrakció bevonásával tehető azzá), mert ő maga nem hierarchikus felépítésű.

Alberti (1999) a másik utat követte: a DRT-féle nem-hierarchikus szemantikához keresett egy kompozicionális szintaxist, amely nem épít fákat, és ahol a nyelvtan csupán a lexikai egységek tulajdonságainak (köztük a mondattá való összekapcsolódásukhoz szükséges megszorításoknak) a gyűjteménye. A szórendről természetesen a GASG is számot tud adni, mégpedig szomszédossági rangparaméterek segítségével, amelyek szintén a lexikai egységek leírásában tárolódnak.

A GASG úgy izomorf a DRT-vel, hogy nem alkalmaz λ-absztrakciót, vagy bármilyen hasonlóan „túl erős" eszközt. Az izomorfia egy megszorítottabb változatát teljesíti tehát, amely ténylegesen mond is valamit a nyelvről. A GASG megalkotása a kompozicionalitás problémájára természetesen csak egy alternatíva a sok közül, célunk, hogy kipróbáljuk, képes-e a nyelvet egészében leírni, illetve, hogy alapjain megalkotható-e egy hatékony nyelvelemző és gépi fordító rendszer. A döntést azok az évtizedek hozhatják meg, ahol nem találunk ilyen szigorú értelemben vett izomorf szemantikát valamiféle szintaxishoz. Nem állítjuk, hogy a helyes hozzáállás mindenképpen az, hogy egy jól működő szemantikához kell keresni vele izomorf szintaxist, a lényeg – a nyelvelméleti cél – az, hogy végül megtaláljuk a leginkább egymáshoz igazodó szintaxis-szemantika párt (anélkül, hogy bármelyiket is előre rögzítenénk).

1.3 Célok

A GASG tehát egy olyan grammatika, amely a DRT-féle szemantikákkal izomorfnak tekinthető. Minden információt a lexikonban tárol (totálisan lexikalista), vagyis egy *homogén* nyelvtan. Kipróbál egy nagyon erős Univerzális Grammatikai megszorítást, azt, hogy csak lexikon van, és nincs szintaxis. Akárcsak a Kamp-féle DRT-ben, ahol csak referensek és állítások vannak, és nincs például λ–absztrakció. Nem biztos, hogy egy homogén nyelvtan hatékonyabb, de azt valljuk, hogy az alapkutatás egyik kiemelt célja kell, hogy legyen az uralkodó elképzelések megkérdőjelezése, leágazó ösvények bejárása, új eljárások kipróbálása. Ezt tesszük mi is, a totális lexikalizmus elméletének és implementációjának kidolgozásával. Ez utóbbi mellett azért döntöttünk, mert egy működő implementáció a legjobb bizonyítéka egy rendszer egzaktságának, formalizálhatóságának, működőképességének.

Elméleti célunk tehát annak igazolása, hogy a GASG egy olyan grammatikai modell, amely alkalmas bármely nyelv leírására, definíciói pontosak, mechanizmusai jól működnek; illetve hogy a totálisan lexikalista megközelítés hasznos eszköze lehet a nyelvleírásnak. Az elmúlt évtizedekben a korábbi szintaxisközpontú grammatikák irányából erőteljesen a szótári komponens minél részletesebb kidolgozása felé tolódott el a hangsúly. A lexikalista

elméletek sikeressége azt mutatja, hogy érdemes ebben az irányban vizsgálódni, és kipróbálni egy totálisan lexikalista grammatikát.

Gyakorlati célunk létrehozni egy nyelvelemző programot a GASG alapján, amelynek a központi komponense egy (diskurzus)szemantikai reprezentáció, így alkalmas olyan magasabb szintű nyelvtechnológiai célokra is, mint például a kérdés-megválaszolás, vagy a jó minőségű gépi fordítás. Széles körben elfogadott nézet, hogy intelligens alkalmazásokhoz nem elégségesek a pusztán statisztikai alapokon működő rendszerek, azokhoz már elméleti (nyelvészeti) alapokon nyugvó alkalmazásokra van szükség. Programunktól azt várjuk, hogy teljesítse a generatív alapfeladatot: a beírt szósorról tudja eldönteni, hogy grammatikus-e, és amennyiben az, rendeljen hozzá kimenetként morfológiai, szintaktikai és – ami a legfontosabb – szemantikai reprezentációt. Az a hipotézisünk, hogy ha elég részletes az így kapott szemantikai reprezentáció, segítségével bármely két nyelv között megvalósítható a gépi fordítás. Célunk egy olyan rendszert létrehozni, ahol a keret nyelvfüggetlen, így újabb nyelvek bevonásakor csupán a lexikai egységeket kell rögzítenünk összes tulajdonságukkal együtt, és nem kell minden nyelvhez külön mechanizmust írni az elemzéshez, illetve minden nyelvpárhoz külön algoritmusokat definiálni a gépi fordításhoz.

Projektünkben tehát elméleti és alkalmazott nyelvészeti célok fonódtak össze elválaszthatatlanul. A grammatikát a számítógépes implementálhatóság legitimálja (célunk ezt megmutatni), másrészt minden nyelvészeti ötlet azt a célt is szolgálja, hogy minél jobb, minél többféle célra felhasználható nyelvelemző rendszerünk legyen. Munkánkkal azt is szeretnénk bizonyítani, hogy a számítógépes nyelvészetnek érdemes visszafordulnia a tiszta (generatív) nyelvelméleti alapok felé.

1.4 Eredmények

Jelenlegi elemzőnk néhányszáz szavas adatbázison működik. A lexikai egységek morfémák, vagyis tövek és toldalékok, összes (fonológiai, morfológia, szintaktikai és szemantikai) tulajdonságukkal együtt. A program a beírt mondatokhoz négyféle reprezentációt társít: morfológiai (szavak morfémákra bontása), szintaktikai (régens-vonzat viszonyok, szabad bővítmények), szemantikai (egy DRS); továbbá a mondat ún. kopredikációs hálózatát is elkészíti, amely egy szintaxis és szemantika közötti szint, és a gépi fordítást segíti.

A program magyar és angol nyelvű mondatokat elemez, és köztük a gépi fordítást is megvalósítja, mindkét irányban. Ehhez alapvetően a kopredikációs hálózatot használja, amely nagyon hasonló az azonos jelentésű magyar és angol mondatok esetében. Ha kérjük (és van), több megoldást is ad.

Az elemzőt Prolog programnyelven írtuk. A Prolog egy magas szintű programnyelv, sok számítógépes nyelvészeti projekt használja. Mi alapvetően azért választottuk, mert – akárcsak a GASG esetében – működésének motorja az unifikáció.

A totális lexikalizmus elvének gyakorlatban való kipróbálásán két projekt is dolgozott. Az első programot a GeLexi-pojekt készítette, szigorúan a GASG alapján; a fő cél az elmélet igazolása volt. Kicsivel később a LiLe-projekt új adatbázist épített, szintén totálisan lexikalista alapokon, de a technológiai szempontok figyelembe vételével (bővíthetőség, rugalmasság, webes megjeleníthetőség stb.). A lexikont relációs (SQL) adatbázisban tároltuk, az elemző programot pedig Delphi programnyelven írtuk. Elsődleges célunk egy többféle célra felhasználható adatbázis (és program) létrehozása volt. A LiLe-projekt lexikonának a részletes kidolgozása csak a morfológiai szintig jutott, és csupán néhányszáz lexikai egységet tartalmaz. Ennek oka, hogy 2006-ban a két projekt egybefonódott, és jelenleg új lexikon és program készül az eddigi tapasztalatok alapján (ReALIS-projekt). A cél továbbra is kettős: az elmélet igazolása és egy hatékony nyelvelemző (és gépi fordító) program megalkotása.

Az új rendszer az elmúlt négy év tanulságai alapján készül, és továbbra is a totális lexikalizmus implementációjának tekinthető. Központi komponense természetesen a szemantika marad, viszont a korábban használt LDRT-nél „fejlettebb” dinamikus szemantikai elméletet használ, a ReALIS-t (Reciprocal and Lifelong Interpretation System, Alberti 2005a), amelyre alaposabb kidolgozottság jellemző, tekintetbe vesz olyan faktorokat, mint a kölcsönösség az interpretáció során (a hallgató és a beszélő tudása egymásról), a retorikai viszonyok, vagy különféle pragmatikai ismeretek. Hipotézisünk az, hogy egy ennyire részletes szemantikai reprezentációt nyújtó elméletet mint *interlingvát* használva elérhető lehet a nyelvpárfüggetlen gépi fordítás.

1.5 A nyelvtechnológia kihívásai

A számítógépes nyelvészet témakörébe nagyon különböző területek tartoznak. Kezdve az olyan elméleti kutatásoktól, amelyek számítógépre alkalmazva hatékony eszközei lehetnek szövegek elemzésének, egészen az olyan nagyon konkrét alkalmazásokig, mint a beszéd-felismerő és beszédszintetizáló szoftverek. Némely feladat már többé-kevésbé megoldottnak tekinthető, de a legtöbb közülük még további kutatásra szorul.

A legkomplexebb számítógépes nyelvészeti feladat olyan nyelvelemző programok megalkotása, amelyek magasabb szintű részfeladatok teljesítésére is képesek: például kinyerni az információt a szövegből, vagy kérdésekre válaszolni, vagy jó minőségű gépi fordítást adni. Ehhez morfológiai, szintaktikai, sőt a legtöbb esetben valamiféle szemantikai elemzésre is szükség van. Ezekben az esetekben már általában nem elégségesek a statisztikai módszereken alapuló megközelítések, „igazi” nyelvészeti elemzésre van szükség.

Jelenleg nincs még olyan rendszer, amely nagy adatbázison képes nagy pontossággal ilyen típusú elemzéseket készíteni, vagy két (nem hasonló) nyelv között jó minőségben fordítani. Napjainkban a legígéretesebbnek a lexikalista (unifikációs) megközelítések tűnnek. Több elemző és gépi fordító rendszer készül például LFG vagy HPSG alapokon. Sokkal pontosabbak, mint a korpusz alapú megközelítések programjai, és lefedettségben is kezdik felvenni a versenyt velük.

Úgy tűnik tehát, hogy a lexikalizmus jó irány lehet a gépi fordítás problémájának megoldása felé. Ezért is tartjuk érdemesnek kipróbálni, mennyire hatékony rendszer alkotható totálisan lexikalista alapokon.

1.6 A dolgozat felépítése

Dolgozatomban azt vizsgálom tehát, hogy a totális lexikalizmus mennyire lehet hasznos eszköz a számítógépes nyelvészetben. Először a nyelvtechnológia jelenlegi állását mutatom be (2. fejezet): mik az elvárások, milyen típusú alkalmazásokat fejlesztenek, amelyek milyen részfeladatokra kínálnak megoldást. Külön említem a magyar nyelvre készített programokat. Majd részletesebben ismertetem a lexikalista megközelítéseket és alkalmazásokat (3. fejezet): szólok többek között az LFG-ről és HPSG-ről, amelyek nem csupán elméletben sikeresek, hanem működő számítógépes alkalmazásaik is vannak különféle nyelvek elemzésére és a gépi fordításra. Ezután a GeLexi-projekt (4. fejezet), majd a LiLe-projekt (5. fejezet) céljait és eredményeit tárgyalom. Ismertetem az alkalmazott technológiát is, és példákon keresztül bemutatom a programok működését. A 6. fejezetben a kutatás újabb irányára térek ki (ReALIS-projekt), végül az összegzésben (7. fejezet) összefoglalom a legfontosabb tanulságokat: miért érdemes a totális lexikalizmus implementálásán dolgozni, mit értünk el eddig, és mi lehet a kutatás következő lépése.

2 Nyelvtechnológia

Számítógépes nyelvészetről kicsit több, mint ötven éve beszélhetünk, amikor a gépi fordítás igénye először felmerült, és az akkori kutatók számára megvalósíthatónak tűnt. Hamarosan be kellett látniuk azonban, hogy a probléma sokkal bonyolultabb, mint először képzelték, sok kutatást és rengeteg részfeladat megoldását igényli. Ekkor kezdtek mind a számítógépes szakemberek, mind a nyelvészek egyre nagyobb számban ezzel a területtel foglalkozni. Az elmúlt évtizedek alatt számos problémára születtek is megoldások, működő programok, de sok közülük máig is megoldatlan, mint például a gépi fordítás maga. Léteznek ugyan gépi fordító programok, de hatékonyságuk távolról sem éri el a kívánt szintet.

A kitűzött célok eléréséhez két oldalról is szükség volt fejlődésre: a nyelvészeti elméletek és a technológia oldaláról. Elméleti oldalról konzisztens, formalizálható, így könnyen és hatékonyan implementálható nyelvészeti elméletekre, rendszerekre volt szükség (GPSG, LFG, véges állapotú eszközök fonológiára, morfológiára). A technológia oldaláról pedig olyan praktikus módszerek kellettek, amelyekkel hatékonyabbá tehetők a számítógépes programok (leginkább különféle statisztikai módszerek, újabban tanító algoritmusok, korpuszok használata). Ki kellett továbbá dolgozni a részfeladatok körét, amelyekre bontva könnyebben elérhetőnek tűnt a kiinduló cél – a gépi fordítás – megvalósítása. Időközben természetesen egyéb fontos célok is megfogalmazódtak, mint például az oktatás támogatása.

2.1 Elméleti háttér

Amikor a formalizálhatóság igénye először felmerült, még nem voltak olyan nyelvi rendszerek, amelyek azt megvalósították volna. Ahhoz, hogy ilyenek születhessenek, szükség volt a nyelvtechnológia matematikai alapjainak lefektetésére.

2.1.1 A Chomsky-féle nyelvtan-hierarchia

Egy fontos lépés volt az egzakt, formális rendszerek kialakulása felé Chomsky nyelvtan-hierarchiájának a megalkotása (alapos ismertetőt nyújt Partee (1990), vagy magyarul Alberti (2006a)). Chomsky négy osztályba sorolta a lehetséges nyelvtanokat szabályaik bonyolultsága alapján. Ezzel egy nagyon hasznos eszközt adott a kutatók kezébe ahhoz, hogy a különféle nyelvtanokat hatékonyság alapján osztályozhassák. A következő definíciók Alberti (2006a)-ban olvashatók:

> **Definíció** (nyelvtan/grammatika): A $G=<V_T, V_N, S, R>$ négyest *(formális) nyelvtannak* nevezzük, ahol
>
> V_T és V_N diszjunkt véges halmazok (V_T-t a G nyelvtan *terminális* ábécéjének, a V_N halmazt pedig a *nem-terminális* ábécéjének nevezzük),
>
> S egy kitüntetett eleme V_N-nek (a nyelvtan *kezdőszimbóluma*),
>
> R pedig egy tetszőleges véges részhalmaza a következő halmaznak (Descartes-szorzatnak): $(\Sigma^* \times V_N \times \Sigma^*) \times \Sigma^*$, ahol $\Sigma = V_N \cup V_T$ (a nyelvtan *szabályainak* a halmaza).

Definíció (Chomsky-féle nyelvtan-hierarchia): Egy $G=<V_T, V_N, S, R>$ nyelvtan i típusú (i=0, 1, 2, 3) a *Chomsky-féle nyelvtan-hierarchia* szerint, amennyiben az R szabályhalmaz valamennyi elemére teljesül az adott típusban előírt (a szabály felépítésére vonatkozó) megszorítás.

Jelölések: α, β és $\mu \in (V_T \cup V_N)^*$, A és $B \in V_N$, és $x \in V_T^*$.

0. típus (*megszorítatlan újraíró rendszer*): nincs speciális megszorítás (a nyelvtan definíciójához képest).

1. típus (*környezetfüggő* nyelvtan): $\alpha A \beta \rightarrow \alpha \mu \beta$, ahol $\mu \neq e$.

2. típus (*környezetfüggetlen* nyelvtan): $A \rightarrow \mu$.

3. típus (*jobblineáris* nyelvtan): $A \rightarrow xB$ vagy $A \rightarrow x$.

Minél magasabb szintű egy nyelvtan, annál átláthatóbbak a szabályai, annál jobban látszik, hogyan működik, így egyszerűbb algoritmussal számolható, grammatikus-e egy adott füzér. Viszont minél magasabb szintű a grammatika, annál szűkebb azon nyelvek köre, amelyek leírhatóak vele.

Kérdés, hogy a természetes nyelvek hol helyezkednek el a hierarchiában. Nyelvtechnológiai szempontból az lenne a legjobb, ha a 3. típusba sorolhatnánk őket, mert akkor véges automatával[1] modellezhetőek lennének, amelyet egyszerű számítógépre alkalmazni, és hatékony algoritmus írható hozzájuk. Bebizonyították azonban, hogy a természetesen nyelvek tartalmaznak olyan jelenségeket, amelyeket nem lehet 3. típusú szabályrendszerrel generálni. Ilyen például az $a^n b^n$ sémát mutató (1) alatt olvasható jelenség (ismét Alberti (2006a)-ból).

(1) *a. Egy kutya összerogyott.*
b. Egy kutya, amelyiket egy kutya kergetett, összerogyott.
c. Egy kutya, amelyiket egy kutya, amelyiket egy kutya kergetett, kergetett, összerogyott.
d. ...
e. Egy kutya (, amelyiket egy kutya)^n (kergetett,)^n összerogyott.

Mikor tovább vizsgálódtak a kutatók, azt találták, hogy a környezetfüggetlen nyelvosztályba sem férnek bele a természetes nyelvek, ami kívül reked, az a svájci-német keresztező függőség, amely az $a^n b^m a^n b^m$ sémát mutatja, amit nem lehet környezetfüggetlen grammatikával generálni.

A környezetfüggő szabályrendszer azonban túl erős eszköz a természetes nyelvek leírására, olyan szerkezeteket is generálni tud, amelyek nem fordulnak elő a nyelvekben. Továbbá a 2. típusú nyelveknél csak egy kicsivel bővebb a természetes nyelvek osztálya, ezért is lenne „pazarlás" 1. típusú grammatikát használni leírásukra.

A megoldás az *enyhén környezetfüggő* nyelvosztály bevezetése. Ezek olyan – alapvetően környezetfüggetlen – nyelvtanok, ahol a megszokott apparátus kiegészül valamivel. Az *indexelt nyelvtanok* esetében például indexeket adhatunk a nemterminális szimbólumok mellé, ami olyan, mintha egy plusz memóriával toldanánk meg a 2. típusú grammatikát. Így válik kitűnően alkalmassá például egyeztetési jelenségek leírására. A

[1] Automata: egy olyan absztrakt számítógép, amely bemenetként megkap egy karaktersorozatot, és arról el kell döntenie, hogy jól formált (grammatikus, része az adott nyelvnek) vagy nem. Ahogy a szimbólumokat olvassa, különféle állapotokba kerül, és akkor fogadja el a füzért, ha végig tudja olvasni, és közben végállapotba jut. Némely automata rendelkezhet valamiféle „memóriával", vagyis meg tudja jegyezni, miket csinált az adott állapot előtt, vagy hogyan jutott oda. A véges automaták erre nem képesek, ezért gyorsabbak, viszont szűkebb nyelvosztály (a 3. típusú) felismerésére használhatók.

faépítő nyelvtan (*tree adjoining: TAG*) úgy működik, mintha egy környezetfüggetlen grammatika által generált összetevős szerkezeti fába építenénk bele ugyanilyen fákat. Különösen alkalmas arra, hogy minimális szerkesztettségű mondatokból facsatolással előállítsa az összes lehetséges összetevős szerkezeti fát. Egy harmadik igen jelentős csoportja az enyhén környezetfüggő grammatikáknak a *kategoriális nyelvtanok* osztálya, ahol a nyelvtani szabályok köre szűk (a klasszikus kategoriális nyelvtan esetében összesen kettő), generatív kapacitásuk a terminális szimbólumokhoz rendelt (lexikai) kategóriákban rejlik. A nyelvtantípust később részletesebben is bemutatom, mivel a projektünk által implementált GASG egy módosított kategoriális nyelvtannak tekinthető.

Szerencsére a természetes nyelvek leírásához mégsem szükségesek minden szinten az algoritmikusan kevésbé hatékony enyhén környezetfüggő nyelvtanok. Bár egy-egy nyelv *egészében* nem írható le 3. típusú szabályokkal, a szintaxisnál egyszerűbb komponenseik igen, például a fonológia vagy a morfológia. Véges automatával, transzducerrel, illetve egyéb véges eszközökkel több hatékony morfológiai elemzőt is készítettek már, amelyek természetesen a fonológiai váltakozásokról is számot tudnak adni.

Vannak olyan nézetek is (pl. Prószéky 2003a), melyek szerint a természetes nyelveket mégis besorolhatnánk – legalább gyakorlatban – akár a 3. típusba is, szintaktikai komponensükkel együtt. Így azokat véges eszközökkel modellezhetnénk, ami hatékonyabb elemzőket eredményezhetne. Az érv az, hogy gyakorlati szempontból nem szükséges például az a^nb^n típusú jelenségeket kezelni tudni, hiszen az (1)-hez hasonló példák már n=2 esetében sem fordulnak elő a nyelvben[2]. Mindezek ellenére nem lehet tagadni a Chomsky-féle nyelvtan-hierarchia jelentőségét a nyelvtechnológia (és a nyelvelmélet) fejlődésével kapcsolatban. Hatására olyan formalizált nyelvtanokat alkottak, amelyek implementálása ígéretes nyelvelemző és gépi fordító rendszerek kifejlesztését tette lehetővé.

2.1.2 Kategoriális nyelvtanok

Enyhén környezetfüggő apparátust több lexikalista nyelvtan is használ, mint például az LFG, a HPSG, vagy az LTAG. Ezeket az elméleteket a későbbiekben még részletesebben bemutatom. Ebben a pontban a különféle kategoriális nyelvtanokról szólok pár szót Alberti (2006a) alapján, mivel (mint már említettem) a GASG is egyfajta kategoriális nyelvtannak tekinthető.

2.1.2.1. Klasszikus kétirányú kategoriális nyelvtan

Ez a nyelvtantípus összesen két szabályt tartalmaz, amelyek segítségével az adott terminális szimbólum összevonható jobb, illetve bal oldali szomszédjával. Alkalmazhatóságuk azon múlik, hogy a kombinálandó terminálisok *kategóriái* (lexikai címkéi) összeillenek-e. Érdekessége ezeknek a nyelvtanoknak, hogy a generálásokat fordítva végezzük, és hogy egy terminális szimbólumhoz akár több kategória is tartozhat (ami megnehezíti a generálhatóság megállapítását). A következő definíció Alberti (2006a)-ban olvasható:

> **Definíció:** Egy *klasszikus kétirányú kategoriális nyelvtan* egy olyan $G=<V_T, V_A, S, F>$ négyesként definiálható, amelyben
> V_T *terminális szimbólumoknak* egy véges halmaza;

[2] Chomsky szerint ezek a konstrukciók grammatikusak ugyan, csupán percepciónk szab határt használatuknak. Ezért egy grammatikának generálni kell tudni őket, bármekkora *n* esetében, vagyis 3. típusúnál kevésbé megszorított grammatikára van szükség.

V_A az elemi *kategória szimbólumoknak* egy véges ábécéje; a kategoriális nyelvtan működtetése során azonban a kategóriáknak egy ennél bővebb C halmazát használjuk, mely az alábbi rekurzív definícióval írható le: ha $x \in V_A$, akkor $x \in C$ is fennáll; ha pedig x és y C-beliek, akkor *(x/y)* és *(x\y)* is C-beliek; az eddigi módszerek véges alkalmazásával előállítható szimbólum-füzéreken kívül azonban más eleme nincsen a kategóriák C halmazának;
S az elemi kategóriák V_A halmazának egy kitüntetett eleme;
F pedig egy függvény, amely minden terminális szimbólumhoz (tehát V_T elemeihez) néhány kategóriát rendel, azaz a kategóriák C halmaza P(C) hatványhalmazának egy-egy elemét.
Egy *összevonási* (*redukciós*) *lépés* ($\Rightarrow$) viszony két kategóriafüzér között

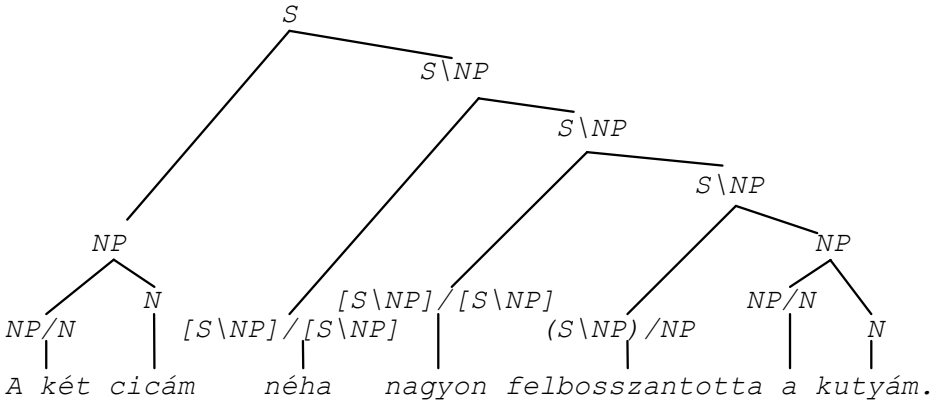
R1 $\alpha\hat{}(X/Y)\hat{}Y\hat{}\beta \Rightarrow \alpha\hat{}X\hat{}\beta$

R2 $\alpha\hat{}Y\hat{}(X\backslash Y)\hat{}\beta \Rightarrow \alpha\hat{}X\hat{}\beta$

ahol X és $Y \in C$, α és $\beta \in C^*$.

$\Rightarrow^*$ szokás szerint a két kategóriafüzér közötti (akárhány (0,1,2...) lépésben való) *összevonást* jelöli, vagyis az összevonási lépés reflexív tranzitív lezártját.
A fenti G grammatika akkor *generálja* a $w \in V_T^*$ terminális füzért, amennyiben a $w=w_1w_2...w_n$ felbontás mellett van olyan $c=c_1c_2...c_n$ kategóriafüzér, amelyre a következők teljesülnek:
— a *c* kategóriafüzér a w terminális füzérhez *tartozik* abban az értelemben, hogy $c_i \in F(w_i)$ (i=1,2,...,n),
— és c-től összevonásokkal eljuthatunk S-ig, vagyis $c \Rightarrow^* S$.

A következő példa (1. ábra, szintén Alberti (2006a)-ból) megmutatja, hogy a különböző szintaktikai kategóriákhoz milyen címkéket célszerű rendelni, és azokból hogyan épül fel a mondat. Például a tranzitív igék (S\NP)/NP címkéje azt jelenti, hogy ez egy olyan elem, amely ha jobbról megkap egy NP-t, akkor S\NP kategóriájú lesz, vagyis már csak egy újabb NP-re van szüksége, ez esetben balról, hogy mondatot (S) kapjunk.



1. ábra: Elemzés klasszikus kétirányú kategoriális nyelvtannal

A klasszikus kétirányú kategoriális nyelvtanokkal generálható nyelvek osztálya a környezetfüggetlen nyelvek osztályával egyezik meg, tehát elvileg a természetes nyelvek teljes leírására nem alkalmasak. Ezért is alakultak ki egyéb változataik.

2.1.2.2. Alternatív kategoriális nyelvtanok

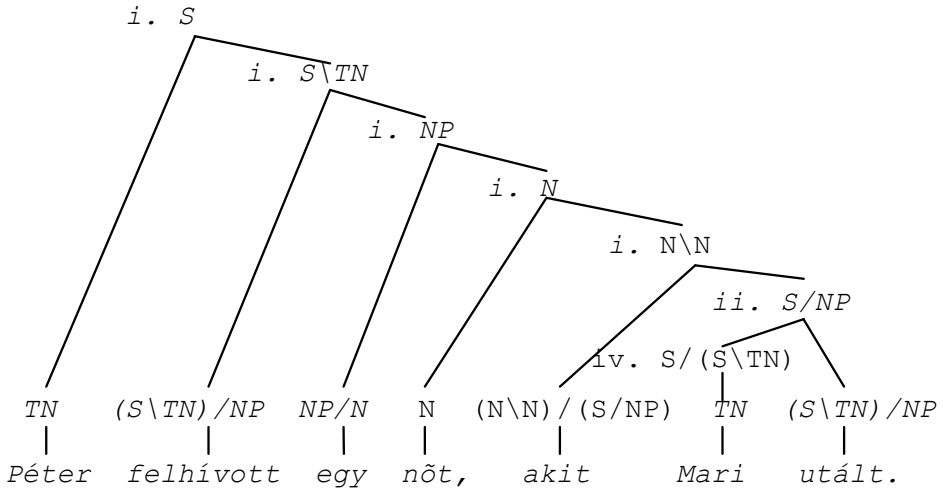
Az elméleti szemponton túl, gyakorlati szempontból is célszerűnek látszott a klasszikusnál kevésbé megszorított kategoriális nyelvtanokat kifejleszteni, például hogy a chomskyánus

generatív nyelvészetben alkalmazott mozgatásos (transzformációs) szabályokat ki lehessen váltani, hiszen azok 0. típusú (megszorítatlan) grammatikát eredményeznének[3].

A függvényalkalmazáson túl új összevonási szabályokat fogalmaztak meg, amelyek bevezetésével többnyire enyhén környezetfüggő grammatikát kapunk. Még nyitott kérdés, hogy a természetes nyelvek leírásához mely összevonási szabályokra van szükség. A már kibővített szabálykészlet (2) alatt olvasható.

(2) i Függvényalkalmazás (Function Application): $X/Y\ Y \Rightarrow X$
ii Függvénykompozíció (Function Composition): $X/Y\ Y/Z \Rightarrow X/Z$
iii Kommutativitás (Commutativity): $(X/Y)\backslash Z \Rightarrow (X\backslash Z)/Y$
iv Típusemelés (Type Raising): $X \Rightarrow Y/(Y\backslash X)$
v Geach-szabály: $X/Y \Rightarrow (X/Z)/(Y/Z)$

A szabályok fordított változatai is elképzelhetőek, ezeket / és \ következetes felcserélésével állíthatjuk elő. A 2. ábra (Alberti 2006a) ezek alkalmazását mutatja egy vonatkozó mellékmondatot tartalmazó (és így klasszikus kategoriális nyelvtannal nem elemezhető) mondaton.



2. ábra: Elemzés függvényalkalmazáson (i) kívül függvénykompozíciót (ii) és típusemelést (iv) is használó kategoriális nyelvtannal.

2.1.2.3. Unifikációs Kategoriális Nyelvtan

Új szabályok bevezetése helyett más módon is megnövelhető egy kategoriális nyelvtan generatív kapacitása, például *unifikáció* alkalmazásával (CUG: Karttunen 1986, UCG: Zeevat 1987). A kapott nyelvtan olyan, mintha a klasszikus kategoriális nyelvtant kombinálnánk az indexelt nyelvtannal. Eredményül enyhén környezetfüggő grammatikát kapunk, amire bizonyíték például, hogy a keresztező függőség generálható vele (Zeevat 1987). A nyelvtan további jellemzője az *egyszintűség*, vagyis a morfológiai, a szintaktikai és a szemantikai elemzés egy időben zajlik, a különböző megszorítások egyszerre lépnek működésbe.

Karttunen (1986) a finn nyelvhez keres unifikációs kategoriális grammatikát, amely (akárcsak a magyar) nem-konfigurációs, a szintaktikai viszonyokat alapvetően a (gazdag) morfológia kódolja. Az unifikációs mechanizmus az attribútum–érték párokba rendezett

[3] Illetve tekinthető 1. típusúnak is, mert belátható, hogy a 0. típusú AB → BA sémára épülő transzformációs szabályok könnyedén kiválthatók három 1. típusú szabállyal (AB → AC, AC → BC, BC → BA).

morfológiai jegyek ellenőrzéséért felelős, kiszűri a hibás egyeztetéseket, eseteket stb. Az elemzést a kategoriális nyelvtan függvényalkalmazás nevű művelete végzi, amely összerakja a mondat egymás melletti szavait, melléktermékként létrehozva egy közvetlen összetevős szerkezeti fát, amely csupán „analízis-fa", a róla leolvasható összetevőknek nincsen semmi nyelvészeti relevanciája. Fellép továbbá egy nagyfokú „hamis többértelműség" a legtöbb mondat esetében, mivel ugyanahhoz az olvasathoz számtalan eltérő analízis-fa tartozik, egyetlen unifikálódott morfológiai jegyrendszer mellett. „All that matters is the resulting feature set" [csakis az adódó jegyhalmaz számít, semmi más] — állapítja meg Karttunen, és azt a végkövetkeztetést vonja le, hogy egy *radikálisan lexikalista* megközelítés eredményesebb volna („A radical lexicalist approach is a better alternative."), az eddigiek mellett számítástechnikai szempontból is.

A GASG egy olyan módosított unifikációs kategoriális nyelvtannak tekinthető, amely megvalósítja ezt az elvet. A kiüresedett nyelvészeti tartalmú szintaktikai összerakó-művelet teljes kiküszöbölésén alapul: a már eleve redukált szintaxisból kitöröljük még a függvény-alkalmazás műveletét is. Nem épülnek fák, ezáltal a nyelvtan mentes a fent említett hamis többértelműségtől is. Ami marad, az csupán az *unifikáció* művelete. A szórendről is ennek segítségével ad számot: egy szó szomszédossági követelménye ugyanúgy egy unifikálandó jegy, mint a száma, vagy hogy milyen esetű argumentumot vár.

2.2 Elvárások

2.2.1 Az elmélet oldaláról

A számítógépes nyelvészet fejlődéséhez tehát szükség volt a nyelvészet matematikai alapjainak a lefektetésére, és olyan elméletek megalkotására, amelyek könnyen implementál-hatók, illetve amelyek számítógépre alkalmazva hatékony nyelvelemzők kifejlesztését teszik lehetővé. Elméleti szempontból tehát fontos elvárás, hogy mindegyik nyelvi szintről egzakt, formális leírás szülessen. Nem elégséges csupán a *fonológiai*, *morfológiai* és *szintaktikai* komponens kidolgozása, hiszen egy igazán intelligens programhoz elengedhetetlen, hogy *szemantikai*, sőt *pragmatikai* információt is tartalmazzon. Nagyon fontos továbbá a *lexikográfia* és a *diskurzuselmélet* kutatása, formalizálása és implementálása, hogy igazán jól működő szoftvereket kapjunk. Ha a cél az, hogy a mondatokhoz rendelt szemantikai reprezentáció (vagy fordítás esetében a célnyelvi mondat) minden információt tartalmazzon, akkor olyan nyelvi jelenségekről is számot kell adnunk, mint például a fókusz, vagy a mondatok közötti retorikai viszonyok.

Sok elmélet törekszik az *univerzalitásra*, vagyis hogy a különböző nyelveket egységes keretben lehessen leírni. Ez nem könnyű feladat, hiszen alig lehet találni olyan nyelvi állítást, amely minden nyelvben igaz lenne[4]. Egyedül talán a szemantikai reprezentáció szintjét lehet univerzálisnak tekinteni, de vannak nézetek, amelyek szerint az is nyelvspecifikus. Az egyetlen lehetőség tehát olyan elméletek kidolgozása, ahol a *keret* univerzális: valami olyan általánosan megfogalmazott formalizmus, amibe mindegyik nyelv „belepasszol"; és az egyes nyelvek esetében csak a paramétereket kell beállítani (mint a Chomsky-féle „Elvek és paraméterek" esetében). Léteznek is ilyen elméletek, például az LFG, ahol az összetevős szerkezet (c-struktúra) nyelvspecifikus, viszont a funkcionális szerkezet (f-struktúra) többé-kevésbé univerzális (az egyelőre kevésbé kidolgozott szemantikai (s) struktúra pedig még

[4] Éppen ezért azok, akik a számítógépes nyelvészet felé gyakorlati oldalról közelítenek, általában nem foglalkoznak ilyen rendszerek megalkotásával. Azt vallják, hogy hatékonyabb írni különböző programokat a különböző nyelvek elemzésére, mint olyan szoftvert fejleszteni, amely bármely nyelv elemzésére képes.

inkább). Továbbá a c-struktúra is univerzális elveken alapul, amit az LFG optimalitásjelölő rangokkal ellátott változata (az ún. OT-LFG) tud kihasználni leginkább, amely szerint minden nyelvben ugyanazok a megszorítások érvényesek, csak máshogy vannak rangsorolva[5].

2.2.2 A gyakorlat oldaláról

Gyakorlati szempontból egy számítógépes nyelvfeldolgozó rendszernek számos feladatot kell tudnia teljesíteni. Egy nyelvelemző rendszertől például fontos elvárás, hogy a generatív alapfeladatot teljesíteni tudja, vagyis a beírt mondatról el tudja dönteni, hogy grammatikus-e (és különféle reprezentációkat társítson hozzá). Azonban nem minden rendszer tűzi ezt ki célul, hiszen legtöbb esetben a felhasználó nem arra kíváncsi, hogy jól formált-e az adott mondat, hanem arra, hogy milyen információt közvetít. Egyre több hibás szöveggel találkozhatunk (például az interneten), ezért fontos, hogy – ha végez is az adott rendszer grammatikalitás-ellenőrzést –, azokhoz a mondatokhoz is tudjon reprezentációkat (esetleg fordítást) társítani, amelyek nem felelnek meg minden szempontból az adott nyelv szabályainak (Prószéky 2005).

A korábban általánosan megfogalmazott célt (nyelvelemzés, gépi fordítás) felbontották *részproblémákra*, hogy lépésenként haladva könnyebben megvalósíthatóvá váljon. A legtöbb számítógépes nyelvészeti projekt egyszerre egy jelenségre koncentrál, egy probléma megoldására fejleszt minél pontosabb és hatékonyabb programot. A megoldandó részfeladatok között szerepel például a *szegmentálás* (szövegek szavakra, szavak morfémákra bontása), *kategorizálás* (pl. szavak vs. számok, szavak szófajokba sorolása, tulajdonnevek felismerése), *elemzés* (morfológiai: tő és toldalékok, szintaktikai: ige és vonzatai), különféle *egyértelműsítések* (szószintű, mondatszintű), *anaforák* referenciájának megtalálása (pl. egy névmás mire/kire vonatkozik), *beszédfelismerés*, *beszéd-előállítás*, természetes nyelvi *szöveg előállítása* stb.

A gépi fordítás mellett időközben természetesen egyéb célok is megfogalmazódtak, amelyek elérését az imént említett folyamatok segítik. Ilyen például a gépi fordításhoz kapcsolódó egyéb szoftverek létrehozása, amelyek (főleg az interneten fellelhető) idegen nyelvű szövegek megértését segítik, mint például a többnyelvű elektronikus szótárak. De fontos cél olyan programok létrehozása is, amelyek képesek *visszakeresni* vagy *kinyerni információt* szövegekből, *összefoglalni* szövegrészek tartalmát, *kérdés-válasz* dialógusokat kezelni, így (akár írásban, akár szóban működő) *interaktív rendszerek* alapját képezni, vagy *multimédiás eszközök* megalkotását és használatát sokoldalúbbá és kényelmesebbé tenni. Fontos cél továbbá az oktatás támogatása, olyan szoftverek létrehozása, amelyek például a *számítógéppel segített nyelvtanulást* (vagy bármi más tanulását) hatékonyabbá tudják tenni.

2.3 Alkalmazások

Mi az, ami mindezekből már megvalósult, és mi az, amin még dolgozni kell? Mindegyik problémát érdemes még kutatni, mert ami megvalósult és megoldottnak tekinthető, azt is érdemes megpróbálni még hatékonyabbá tenni. Általánosságban azt mondhatjuk, hogy az alacsonyabb nyelvi szintek kezelésére többé-kevésbé képes a tudomány. Vannak elfogadható minőségű kimenetet adó beszéd-előállító programok, a beszédfelismerés is működik, bár a legjobb eredményeket akkor produkálja, ha az adott program be van tanítva az őt használó ember hangjára. A morfológiai elemzés sem okoz problémát a számítógépes nyelvészeknek,

[5] Ehhez nagyon hasonló a GASG megközelítése az ún. rangparaméterek alkalmazásával.

mégis készülnek újabb és újabb morfológiai elemzők, amelyek gyorsabbak és/vagy valamilyen más szempontból jobbak, mint elődjeik. Szintaktikai elemzésre képes szoftverek is léteznek, amelyek ki tudják szűrni a grammatikailag helytelen mondatokat, de ezek pontossága jelentősen elmarad egy morfológiai elemző pontosságától, így ezen a területen még inkább szükség van további fejlesztésekre. A cél sokszor csak az ún. *sekélyelemzés* (vagy részleges elemzés, shallow parsing), vagyis csupán az ige és főnévi csoportok azonosítása, nem egy komplett szintaktikai elemzés.

A legmagasabb nyelvi szintek (szemantika, pragmatika) implementálása pedig még kevésbé valósult meg eddig, pedig ezek nélkül nem hozható létre intelligens nyelvelemző vagy gépi fordító program. Ezért a mai gépi fordítók vagy szinte csak szóról szóra képesek fordítani (ezért sokszor hibáznak például a többjelentésű szavaknál és egyéb helyeken), vagy úgynevezett *fordítómemóriákból* dolgoznak, óriási párhuzamos korpuszokat használva, amelyek nem mindig bizonyulnak elégségesnek. Nem véletlen, hogy az utóbbi években a különféle szemantikai, pragmatikai kutatások és azok implementálásának lehetőségei bekerültek a számítógépes nyelvészet központi és egyik leginkább kutatott problémái közé.

2.3.1 Módszerek

A programok hatékonyabbá tételét segítő módszerek közül a legrégebben és legáltalánosabban a különféle *statisztikai módszereket* használják. Szinte minden nyelvi szinten alkalmazhatók, és jelentősen gyorsabbá teszik a szöveg feldolgozását. Egy másik terület, amely az utóbbi években egyre nagyobb teret hódít, a *korpusznyelvészet*. Különböző szövegek elemzésekor nagyon hasznosnak bizonyult óriási méretű (több millió szavas) korpuszokat (különböző helyről származó szövegeket) használni. Ezeket a szövegeket különböző mértékben annotálják (leginkább kézzel), vagyis a lexikai egységekhez különféle morfológiai, vagy akár szintaktikai, sőt szemantikai információt társítanak. Minél alaposabban annotált egy korpusz, annál jobban használható, viszont annál nagyobb munka előállítani. Intelligensebb programokhoz különféle *ontológiák* használata is szükséges lehet, amelyek a világról való tudásunkat tárolják valamilyen formában. Végül az utóbbi években terjedt el szélesebb körben a különféle *tanuló algoritmusok* használata, hiszen a leghatékonyabb az lenne, ha a számítógép is – hasonlóan, mint a kisgyerek – valamennyi inputból tanuló képessége segítségével meg tudná tanulni a nyelvet.

Két fő megközelítés létezik az általunk is kitűzött célokra (komplex nyelvi elemzés, fordítás): a *szabály alapú* és a *korpusz alapú*. Az előbbi a számítógépes nyelvészet korai éveire volt jellemző, mígnem rájöttek, hogy nem tudnak megfelelő mennyiségű nyelvi adatot rögzíteni. Ekkor kezdtek már meglévő szövegeket felhasználni nyelvtechnológiai célokra, amelyekből egyre több és változatosabb „műfajú” áll rendelkezésre az interneten. Sikerrel alkalmazták a korpusz alapú megközelítéseket mind az elemzés mind a gépi fordítás területén. Sokszor azonban a szószintű annotálás nem elég, például nem mindig találhatók meg a frázisok határai, vagy a különírt szóösszetételek. Jobban hasznosítható a mondatszerkezettel is annotált korpusz – *treebank* –, amely (nevével ellentétben) nem csupán faszerkezetre vonatkozó információt tartalmazhat, hanem egyéb szintaktikai információt is, mint például grammatikai funkció, predikátum-argumentum viszonyok. Ha pedig a treebankot szemantikai jellemzőkkel is annotálják, akkor még eredményesebb elemzőt kaphatunk (Fujita et al. 2007).

A korpuszon alapuló alkalmazások nagyon hatékonyak tudnak lenni, de két szempontból problémásak. Az egyik, hogy akkor igazán hatékonyak, ha a korpusz alaposan annotálva van (treebank), ami nagyon sok munkát igényel. Ennek kiküszöbölésére kezdtek el olyan algoritmusokat kidolgozni, amelyek lehetővé teszik az ún. *felügyelet nélküli tanulást* (unsupervised learning), vagyis azt, hogy a gép a bevitt adatokat önállóan fel tudja címkézni,

és ne legyen szükség annotálásra. Bár az eredmények nagyon ígéretesek, a módszer nem tart még ott, hogy komplex nyelvi elemzést tudjon adni. A másik probléma, hogy csupán korpusz felhasználásával nem érhető el tökéletes eredmény. Csak egy példát említve: biztosan nem található meg minden magyar szó összes lehetséges ragozott alakjában, ami nem jelenti azt, hogy nem létezik (pl. a „kottáitokban" szó nem található meg a Magyar Nemzeti Szövegtár[6] 187 644 886 szavas adatbázisában). Egy morfológiai elemző beépítése tehát szükségesnek tűnik (a magyarhoz hasonló gazdag morfológiájú nyelvek esetében mindenképpen). Ha a korpuszt magasabb szintű feladatokra akarjuk használni, még több „nyelvészetre" van szükség. Így az ilyen rendszerek már inkább szabály alapúnak tekinthetők, amelyek kisebb-nagyobb mértékben használnak korpuszokat (pl. egyértelműsítésre).

A másik alapvető szembenállás a *statisztikai* és a *mélynyelvészeti* megközelítések között található (Frank 2003). Az előbbi sekély (csonkokon alapuló) elemzést használ, így csak részleges elemzést ad; nagyon robusztus, de kevéssé precíz és alapos. Mivel az elemzés kis komplexitású (gyengébb formalizmust igényel), ezért nagyon hatékony. Ezzel szemben a mélyelemzés alapos, pontos, nagyobb környezetet vesz figyelembe (még akár távoli függőségeket is); nagyon precíz, de kevésbé robusztus. Az elemzés komplexitása jóval nagyobb (bonyolultabb a formalizmus is), ezért kevésbé hatékony. Egyre több alkalmazás próbálja ötvözni a kettőt, hogy mindkettő előnyei meglegyenek[7].

A mélyelemzés egyik fontos jellemzője, hogy kiszűri a grammatikailag helytelen mondatokat, szemben a sekélyelemzőkkel, amelyek bármilyen bemenethez elemzést társítanak. Ez a tulajdonság bizonyos szempontból kívánatos (nem generál hibás szöveget, jobban modellálja a nyelvet), viszont így az elemzés robusztussága jelentősen csökken, hiszen hibás szöveg esetében általában semmilyen kimenetet nem ad. A bemenő szöveg pedig gyakran hibás, legfőképpen beszélt nyelvi input, vagy beszédfelismerővel nyert szöveg esetében. Így az ilyen elemzők fedése jócskán elmarad a sekélyelemzőkétől, ami miatt – hiába a nagy pontosság – a mélyelemzés nem lehet igazi versenytársa a sekélyelemzésnek (Zhang et al. 2007). Ezért a mélyelemzést gyakran kiegészítik olyan módszerekkel, amelyek lehetővé teszik a (kis mértékben) hibás input elemzését is.

A mélynyelvészeti megközelítés terjedését mutatja az is, hogy az utóbbi években egyre több konferenciát és workshopot szerveznek ebben a témában (pl. az ACL 2007[8] konferenciának is volt ilyen workshopja). Egy fontos előnye az ilyen típusú alkalmazásoknak, hogy a folyamat *megfordítható*, tehát nem csak elemzésre, hanem generálásra is használható. A *pontosság* és a *komplex elemzés* ára, hogy a programokat kézzel kell írni, és hogy nem annyira hatékony a futása, mint a statisztikai módszereket használó elemzőké. Az elmúlt években azonban jelentős javulás tapasztalható ezeken a pontokon is. A számítástechnika fejlődésével egyre jobb gépek állnak a kutatók rendelkezésére; folyamatosan fejlesztik az elemző és a generáló algoritmusokat is; illetve statisztikai módszerek integrálásával jelentősen megnövelhető az addig csak nyelvészeti módszereket használó programok hatékonysága. Így tehát napjainkban a mélyelemzést végző alkalmazások egyre hatékonyabbak: egyre több nyelvre alkalmazzák őket, egyre több területen és egyre széleskörűbb feladatokra.

Ezek alapján elmondható, hogy ha pontos és komplex elemzést szeretnénk a programunktól, hogy intelligensebb nyelvtechnológiai célokra is alkalmas legyen, akkor a

---

[6] http://corpus.nytud.hu/mnsz/

[7] Egyre több olyan projekt van, ahol pl. egy mélyelemző, hogy hatékonyabb legyen, statisztikai előfeldolgozást használ, vagy gépi tanulással nyer adatokat; illetve a másik oldalról: ahol egy statisztikai alapú rendszerben egyre több a nyelvészeti tudás, és így olyan problémákat is meg tud oldani, amilyeneket előtte nem.

[8] 45th Annual Meeting of the Association for Computational Linguistics

legígéretesebbnek olyan alkalmazás fejlesztése tűnik, amely nyelvészeti tudáson alapszik (alapvetően szabályokat implementál) és mélyelemzést végez. A megfelelő hatékonyság elérése érdekében azonban statisztikai módszerek bevonására is szükség van, és (minimálisan adattöltésre és egyértelműsítésre) korpuszok használata is elengedhetetlennek látszik. Több működő alkalmazás is létezik már ezeken az elveken, amelyek közül a leghatékonyabbak *lexikalista* elméletet implementálnak, mert a szótár alapú, *unifikációs* mechanizmusokat használó megközelítések jobbnak bizonyultak magasabb szintű elemzések céljára (Mitkov 2003); többek között azért, mert *egyszerűbb szabályok* megfogalmazását teszik lehetővé, mert *egyszintűek* és mert *megfordíthatók*. A lexikalista elméletekről és alkalmazásokról a 3. fejezetben részletesebben is szó lesz.

2.3.2 Gépi fordítás

A gépi fordító rendszereket is két nagy csoportra oszthatjuk: *szabály alapú* és *példa alapú* megközelítésekre (Mitkov 2003). A kezdetekre szabály alapú rendszerek fejlesztése volt a jellemző, míg az utóbbi években inkább hatalmas korpuszok és fordítómemóriák segítségével próbálják létrehozni a minél kisebb emberi beavatkozást igénylő gépi fordító programokat.

A szabály alapú rendszereken belül többféle megközelítést alkalmazhatnak. A fordítás történhet (1) direkt módon, kis elemzéssel, (2) közvetítőnyelven keresztül, illetve (3) transzferrel, amikor a forrásnyelvi mondatból egy absztrakt forrásnyelv-közeli reprezentáción, majd egy célnyelv-közeli reprezentáción keresztül állítják elő a mondat célnyelvi megfelelőjét.

A fordításhoz két lépésben lehet eljutni. Az első lépés az *analízis* (elemzés), amely a forrásnyelvi szöveg szintaktikai elemzésén túl tartalmazhat morfológiai és valamilyen mértékű szemantikai elemzést is; ez utóbbira a többértelműségek kezeléséhez van (minimálisan) szükség. A második lépés pedig a *szintézis* (generálás), amikor a mondat célnyelvi megfelelőjét állítják elő lehetőleg ugyanazzal a mechanizmussal. Így a *modularitás* és a *megfordíthatóság* fontos tulajdonságaik ezeknek a rendszereknek.

Miután a szabály alapú megközelítés nem bizonyult kellően hatékonynak, még statisztikai módszerek bevonásával sem, továbbá rengeteg befektetett munkát igényelt a nyelvtaníró részéről, a 90-es évektől kezdtek tért hódítani a példa alapú fordítórendszerek. Hatalmas korpuszok születtek, amelyeket minél alaposabban annotáltak, annál jobb minőségű elemzőket lehetett rájuk építeni, viszont a ráfordítási idő is jelentősen megnőtt. Közülük leghatékonyabbnak a treebankok (mondatszerkezettel is annotált korpuszok) bizonyultak, hiszen azok tartalmazzák a lehető legtöbb információt; viszont előállításuk nem egyszerű feladat, sok munkaórát igényel. Éppen ezért napjainkban egyre több olyan rendszert fejlesztenek, amely annotálatlan korpuszból képes „megtanulni” a nyelv elemeit és azok tulajdonságait különféle módszerek (analógia, disztribúciós módszer) segítségével, így kevés ráfordítással készíthetnek hatékony elemzőket. Nagyon elterjedt továbbá gépi fordításkor a párhuzamos korpuszok, illetve különféle fordítómemóriák használata, melyeket például az interneten fellelhető több nyelven elérhető anyagok szinkronizálásával állítanak elő.

A korpusz alapú megközelítések sem hoztak teljes sikert. Nem minden jól formált kifejezés található meg bennük, különösen a gazdag morfológiájú nyelvek esetében, mint a magyar, és nagyon ritka az igazán alaposan és jól annotált anyag, vagy megfelelően illesztett párhuzamos korpusz. Emiatt születnek különféle hibrid megoldások: alapvetően korpuszra támaszkodó, de valamilyen szintű elemzést is végző, vagy szabály alapú, de a többértelműségek kezelésében korpuszt is használó rendszerek.

Napjainkban egyre többen kezdenek visszatérni a szabály alapú megközelítéshez, de nem sekélyelemzést használnak, ami csak részleges elemzést ad, és hiába robusztus, ha nem annyira precíz (Frank 2003), így komplexebb célokra (pl. igazán jó minőségű fordítás) nem alkalmas. A mélyelemzést végző rendszerek sokkal alaposabbak és pontosabbak, és az utóbbi időben lefedettségben is kezdik felvenni a versenyt a sekélyelemző rendszerekkel. Az ilyen nyelvészeti alapú, kézzel írt nyelvelemzők készítéséhez szükséges kezdeti nagyobb energiabefektetés pedig megtérül később, például amikor új nyelvekre dolgozzák ki (Forst et al. 2005). Az igazán jól működő és hatékony mélyelemző rendszerek *unifikációs* mechanizmusokat használnak, így előnyeik között szerepel, hogy egyszerűbb szabályokat lehetséges megfogalmazni, gyorsabb, egyszintű, lexikalista és megfordítható (Mitkov 2003). A többértelműség kezelésére ezek a rendszerek is használhatnak korpuszt, mivel sokszor szabályba nem foglalható tényezők szükségesek az egyértelműsítéshez (pl. kontextus), illetve optimalitásjelölő rangokat, hogy a ritka alakzatokat is elfogadja az elemző, de alapesetben a gyakoribb elemzés mellett döntsön (Forst et al. 2005).

A legtöbb gépi fordító rendszert egy adott nyelvpárra dolgozzák ki, sőt, akár csak az egyik irányban működnek, de léteznek többnyelvű rendszerek is, ahol a mechanizmus univerzalitására törekszenek.

Jelenleg a legsikeresebb az ún. *frázis alapú gépi fordítás*, ahol a forrásnyelvi mondatból minél hosszabb szósorokat (frázisokat) helyettesítenek be célnyelvi megfelelőjükkel, a cseréket addig folytatva, míg el nem jutnak a célnyelvi mondathoz. Ehhez (megfelelően illesztett) párhuzamos korpuszokat használnak. Ezzel a megközelítéssel azonban, ha a fordításhoz nem használnak fel nyelvészeti tudást, nem érhető el teljes siker, mert csak nyelvészeti módszerekkel lehet bizonyos szükséges általánosításokat megtenni, hogy az alacsony előfordulású szerkezeteket is felismerje a program, és sikerrel alkalmazza hasonló, de nem teljesen azonos nyelvi közegben (Hopkins–Kuhn 2007).

Ezért sokféle hibrid modellel kísérleteznek a kutatók (Hopkins–Kuhn 2007): az alapvetően nyelvészeti megközelítést egészítik ki statisztikaival, vagy a korábban tisztán statisztikai alapú rendszerüket igyekeznek a szükséges nyelvészeti információkkal pontosabbá tenni. Az első lehetőség – amikor egy klasszikus, szabály alapú rendszerhez adnak hozzá példa alapú eszközöket – ígéretesnek tűnik jó minőségű, speciális célú gépi fordításra, de új nyelvpárra alkalmazni nagyon költséges. A másik lehetőség – amikor egy statisztikai MT modell mélyebb szintaktikai elemzésen alapul – szintén nagyon hatékonynak tűnik elméletben, azonban egyelőre nem sikerült a frázis alapú megközelítésbe megfelelő módon integrálni a szintaktikai információt, így jelenleg a hibrid modell nem teljesít jobban a csak statisztikai módszereket használónál. Egyik út sem tűnik tehát minden szempontból megfelelőnek, ezért újabb és újabb megközelítést használó alkalmazások látnak napvilágot.

2.3.3 Magyar programok

A nyelvtechnológia magyarországi helyzetét alapvetően az MSZNY (Magyar Számítógépes Nyelvészeti) konferenciák kötetei alapján mutatom be. A Szegeden tartott konferenciát 2003-ban rendezték meg először, és azóta is minden évben nagy érdeklődés övezi. Kisebb és nagyobb projektek egyaránt bemutatkozhatnak, így mindenki értesülhet arról, jelenleg hol tart a magyarországi számítógépes nyelvészet, milyen problémákon milyen sikerrel dolgoznak az egyes műhelyek.

2.3.3.1. Fonológia, morfológia

A számítógépes nyelvészet alapjait a különböző nyelvi szintek egzakt elméleti leírása adja. Az alacsonyabb nyelvi szintek (fonológia, morfológia) – kis megszorításokkal – 3. típusú

szabályokkal leírhatók (Chomsky-féle nyelvtan-hierarchia, 2.1.1. fejezet), így kezelésük elméletileg véges állapotú eszközökkel megoldható. Ennek legnagyobb előnye, hogy a számolási idő *lineáris*, vagyis a bemenet hosszával egyenesen arányos (szemben a polinomiális vagy a még rosszabb exponenciális futásidejű algoritmusokkal). A *fonológia* modellezésére a gyakorlatban *véges állapotú transzducereket* használnak leginkább, amely olyan, mint egy véges állapotú automata, azzal a különbséggel, hogy nem csak olvasni, hanem írni is tud, így elemzésre és generálásra is használható (részletesebben: Kálmán et al. 2002. 1. fejezet).

A ragozó nyelvek (mint a magyar) hatékony számítógépes kezeléséhez *morfológiai* komponensre is szükség van. Nélküle akár több millió szóalakot is tárolni kellene a lexikonban, ami erősen hely- és (kereséskor) erőforrás-igényes. Ennél takarékosabb módszer, ha a szótár csak morfémákat (töveket és toldalékokat) tartalmaz, és valamiféle szabályrendszer segítségével ismerjük fel vagy állítjuk elő a különféle ragozott formákat. A szavak tárolására a leghatékonyabbak ismét csak a véges állapotú automaták (vagy keresőgráfok), mert lineáris idő alatt megtalálható bennük a keresett szó.

A szabályrendszer megadásának három fő módja lehetséges (Prószéky–Kis 1999). Az első az ún. *kétszintű morfológia*, amely a felszíni és a szótári alak közötti leképezéseket valósítja meg. A morfémasorrendet véges automatával ellenőrzi, az elemzést pedig véges állapotú transzducer segítségével végzi. Előnye, hogy megfordítható: elemzéskor a felszíni alakhoz hozzá tudja rendelni a lexikai reprezentációt, generáláskor pedig a tő szótári alakja és a kért morfológiai jegyek alapján elő tudja állítani a kívánt szóalakot. Ilyen rendszer az *Intex/NooJ* nyelvészeti fejlesztőkörnyezet magyar nyelvi modulja (Vajda et al. 2004). Jegy alapú rendszert alkalmaz a szokásos paradigmaosztályok helyett, ahol a jegyeket véges állapotú transzducer dolgozza fel. Időnként változókat is használnak, hogy a bemenet is módosítható legyen (pl. *lány+nyá*).

A másik szabálymegadási modell ún. *folytatási osztályok* definiálásán alapul, ahol minden morfémához meg kell adni, milyen más morfémák következhetnek utánuk. Például a *bokor* tövet követheti a *-ban* morféma, de nem követheti a *-ben* vagy a *-t* rag. Ilyen típusú szabálymegadást követ a *Szószablya* projekt keretében fejlesztett *HunTools* eszközrendszer (Németh et al. 2004), amelyet több alkalmazás is használ. Komponensei a Hunspell helyesírás-ellenőrző, a Hunstem tövező és a Hunmorph morfológiai elemző. Legjelentősebb előnyei, hogy az említett három feladatot egyetlen rendszer tudja ellátni, és hogy a keretrendszer nyelvfüggetlen: bármilyen komplex agglutináló morfológiával rendelkező nyelv leírására képes. A tövekről tárolja, hogy milyen affixumok járulhatnak hozzájuk, de lehetőség van további megszorításokat is tenni. Az elemzést rekurzív affixumleválasztással végzi, amely hatékonyabb, mint komplex toldalékegyütteseket tárolni.

Végül a harmadik típusba az *unifikációs modellek* tartoznak, amelyekről Prószéky–Kis (1999) részletesebben is ír; ez az a mechanizmus, amelyet a MorphoLogic által fejlesztett *Humor* („High speed Unification Morphology”) morfológiai elemző is használ (Novák 2003). Lényege, hogy minden lexikai egységnél (amelyek allomorfok, illetve morfok) egy összetett adatstruktúrát tárolnak az egység releváns tulajdonságaival, és két elem akkor „rakható össze”, ha a jegyeik kompatibilisek, azaz ha adatstruktúráik *unifikálhatók*. A morfémák helyes sorrendjét ebben a modellben is véges automata ellenőrzi. A következő példák (Prószéky–Kis 1999) jól mutatják, milyen jegyekre lehet szükség a magyar morfológia unifikációs alapú leírásához (az első lista tartalmazza az adott elem bal oldali szomszédjától elvárt tulajdonságokat, a második pedig a jobb oldalitól; ezért üres tövek esetében az első, míg toldalékok esetében a második lista).

szó: []
[+névszó +fn +szótári –elöl –kerek –PL –PERS +ACC –ACCkötő +DAT +INS:V]

szav: []
[+névszó +fn –szótári –elöl –kerek +PL +PLkötő +PERS –ACC –DAT –INS]

képez: []
[–névszó +szótári +elöl –kerek –ÁS]

képz: []
[–névszó –szótári +elöl –kerek +ÁS]

és: [–névszó +elöl +ÁS]
[+névszó +főnév +szótári +elöl –kerek +PL +PLkötő +ACC –ACCkötő +DAT +INS:S]

ak: [+névszó –elöl –kerek +PL +PLkötő]
[+névszó –elöl –kerek –PL –PERS +ACC +ACCkötő +DAT +INS:K]

val: [+névszó –elöl +INS:V]
[]

sel: [+névszó +elöl +INS:S]
[]

A morfológiai elemzőt hasznos kiegészíteni egy olyan eszközzel (guesser), amely lehetővé teszi, hogy a program a szótárban nem megtalálható tövű szavakhoz is elemzést tudjon társítani (és az így megtalált új tövet esetleg fel is tudja venni). Ennek oka, hogy a morfológiai elemző által használt lexikon még elméletben sem tartalmazhatja az összes magyar szót, hiszen minden nyelvben vannak nyílt szóosztályok (a magyarban például a főnevek, az angolban a főnevek és az igék is). A Humor morfológiai elemző is kiegészül egy ilyen modullal (Novák et al. 2003), hogy az alkalmazás robusztusabb legyen.

2.3.3.2. Lexikográfia, korpusz, ontológia

A számítógépes lexikográfia két alapvető kutatási területet takar. Egyrészt annak vizsgálatát, hogyan lehet az emberek által készített szótárakat nyelvtechnológiai célokra felhasználni; másrészt pedig olyan számítógépes technikák kifejlesztését, amelyekkel a szótárírást hatékonyabbá lehet tenni. Mindkettőre számos példa található a magyar számítógépes nyelvészeti kutatásokban. Sok projekt használja különféle nyelvi információ kinyerésére pl. az Értelmező kéziszótárt (pl. Mihálcz 2003), Elekfi László ragozási szótárát, vagy Vonyó Attila magyar–angol szótárát (Halácsy et al. 2005). A szótárírókat pedig többféle számítógépes technológia segíti. Például az Akadémiai nagyszótár (Pajzs 2003) morfológiai elemzést is használ (a Humor programmal), hogy könnyebben és hatékonyabban lehessen keresni, továbbá címszógyakorisági listát is tartalmaz. A szócikkek adatait XML-ben tárolja, melyből a nyomtatott és az elektronikus változat is könnyen előállítható. A jelölőnyelvek (mint az XML) előnyös tulajdonságairól Vöröss–Trepák (2005) részletesebben is ír (a MobiMouse Plus szótárak készítése közben szerzett tapasztalatok ismertetése kapcsán), és ajánlja minden szótáríró figyelmébe, mert segítségével pontosabb (kevesebb hibát tartalmazó), egységesebb és egyszerűbben feldolgozható szótár alkotható, amelyben lényegesen könnyebb keresni is.

Több korpusz is készült már a magyar nyelvre, az első és legnagyobb a *Szeged Korpusz* (Csendes et al. 2003), amely egy több, mint egymillió szövegszót tartalmazó, szófajilag egyértelműsített, szintaktikailag elemzett adatbázis. Szintaktikai elemzés alatt az ún. lapos (shallow) elemzést érti: megtalálja a főnévi csoportokat (a CLaRK rendszerrel, lásd lejjebb) és képes a tagmondathatárok bejelölése. Az elmúlt években azon is dolgoztak, hogy a mondatokhoz alaposabb elemzést tudjanak társítani, így jött létre a *Szeged Treebank*

(Csendes et al. 2005), amely kb. 82 ezer mondat teljes szintaktikai elemzését tartalmazza (kézzel annotálva).

Egy másik jelentős nagyságú adatbázis a magyar nyelvre a *Hunglish* korpusz és szótár[9] (Halácsy et al. 2005). Alapvető célkitűzésük egy statisztikai alapon működő nyersfordító fejlesztése, melynek első lépéseként megalkottak egy mondatszinten illesztett magyar–angol párhuzamos korpuszt. Nagyméretű adatbázis létrehozása volt a cél, amihez nem állt rendelkezésükre megfelelő mennyiségű párhuzamos korpusz. Ezért összegyűjtöttek az internetről angolul és magyarul is fellelhető szövegeket: szépirodalmi műveket, jogi szövegeket, dokumentációkat, filmfeliratokat, többnyelvű magazinokat stb. Ezeket feldolgozták: először kinyerték belőlük a nyers szöveget, majd mondatszinten párhuzamosították őket. Az illesztéshez kifejlesztették a Hunalign programot, amit aztán elérhetővé tettek szabad felhasználásra. A program kiinduló szótárként Vonyó Attila magyar–angol szótárát használja, morfológiai elemezéshez pedig a Hunmorph programot (morphdb.hu (Trón et al. 2005), illetve morphdb.en). Az így nyert korpuszt kézzel még javítani kellett, de még így is nagyon rövid idő alatt sikerült egy nagyméretű párhuzamos korpuszt és szótárt létrehozni.

Egy speciális szótártípus az *ontológia*, amely nyelvi és/vagy világismereti tudást rögzít. A legismertebb nyelvi tudást rendszerező ontológia a *WordNet*, amelyet az 1990-es évektől kezdtek fejleszteni. Először az angol nyelvre fejlesztették ki (Princeton WordNet, PWN), majd hét másik (holland, olasz, spanyol, német, francia, cseh, észt) európai nyelvre is kidolgozták (EuroWordNet, EWN), végül öt délkelet-európai nyelv (bolgár, görög, román, szerb, török) bevonásával megalakult a BalkaNet, amely a magyar nyelvű WordNet alapját is képezi (Mihálcz 2005). A WordNet eredetileg a mentális lexikon modellálására készült: főnevek, igék, melléknevek és határozószók szemantikai hálózata, ahol a fogalmi csomópontokat szinonima-halmazok (ún. synsetek) alkotják, pl. {léc, deszka}, {fut, szalad, rohan}. A synsetek között szemantikai kapcsolatok is definiálva vannak, mint például alá-, illetve fölérendeltség (hiper-, illetve hiponímia, pl. {toll}–{írószer}), rész–egész viszony (mero- és holonímia, pl. {fa}–{erdő}), ellentét (antonímia, pl. {megszületik}–{meghal}); illetve egyéb (speciálisabb) viszonyok, mint például tulajdonság (melléknév) és a neki megfelelő attribútum (főnév) közötti reláció (pl. {piros}–{szín}).

A magyar WordNet (HuWN) építését 2000-ben kezdték meg a kutatók. Többféle módszer alkalmazására volt szükség. Ahol lehetett, az ún. kiterjesztéses módszert választották, vagyis a magyar nyelvű synsetek létrehozásához az angol megfelelőik fordításával jutottak el (természetesen sok utómunkálattal). Így készült például a főnévi adatbázis (Mihálcz 2003), mivel a főnevek terén az angol és a magyar fogalmi rendszer kellően hasonló egymáshoz. Megtalálni a helyes synset-párokat természetesen így sem volt egyszerű. Két forrásból dolgoztak: alapvetően a Morphologic által készített angol–magyar szótári adatbázist használták, de – különösen a jelentésegyértelműsítéshez – a Magyar értelmező kéziszótárt is segítségül vették. Az egyértelműsítéshez automatikus módszereket fejlesztettek ki, melyek többféle kombinációját alkalmazták, ami után még kézzel ellenőrizni kellett az eredményt.

Az igei synsetek kidolgozásához (Kuti et al. 2005) azonban nem bizonyult megfelelőnek a kiterjesztéses módszer. Ennek oka, hogy az eredeti rendszer több következetlenséget tartalmaz a jelentések elkülönítésében, illetve hogy a magyar és az angol igei rendszer között jelentős különbségek vannak. Ezért a másik, ún. összevonásos módszert alkalmazták, vagyis különféle szótárak (pl. Értelmező Kéziszótár, Magyar Nemzeti Szövegtár) segítségével maguk állították össze az igék sysnsetjeinek rendszerét, beleértve a

[9] Megtalálható: http://szotar.mokk.bme.hu/hunglish/search/corpus.

synsetek között fennálló relációkat is (az implikáció művelete és az időszerkezet alapján). Az igék egyik legfontosabb tulajdonsága az eseményszerkezetük, például hogy az adott ige állapotot fejez ki (*akar*), vagy folyamatot (*fut*), vagy pontszerű eseményt (*elesik*) stb.; fontos továbbá, hogy az adott eseményszerűségnek van-e előkészítő fázisa és/vagy utóállapota. Az említett tulajdonságok több esetben is hasznosnak bizonyulhatnak: tőlük függ például, hogy milyen időhatározókkal bővíthető az adott predikátum, vagy létezhetnek kompozicionális képzők egy nyelvben az egyik típusból a másikba való váltásra stb. A gépi fordításhoz is fontosak ezek az információk, például ezek alapján tudjuk, hogy milyen igeidőt használjuk a magyarról angolra fordításkor (a *futott* helyes fordítása (alapesetben) *he was running*, az *elesett* fordítása viszont *he fell*). Ezért is fontos, hogy egy ontológia ezeket az információkat is tartalmazza. Egy lehetséges osztályozási és kódolási rendszert ismertet Kuti et al. (2006).

A melléknevek rendszerének legmeghatározóbb relációja nem a főneveknél használt alá-, illetve fölérendeltség, hanem az ellentét (antonímia); többek között különféle szóasszociációs kísérletek eredményeit figyelembe véve jutottak erre a következtetésre. Alapvetően a PWN mintájára készült a magyar verzió (Gyarmati et al. 2006), azonban bizonyos pontokon változtatni kellett: ahol nem volt tökéletesen megfeleltethető egymásnak az angol és a magyar synset-struktúra, ahol valamely fogalomra nincs külön szó valamelyik nyelvben, vagy esetleg van, de nem melléknév (pl. az *afraid* melléknév magyar megfelelője a *fél* ige). Két új relációt is felvettek, amivel következetesebbé tették a melléknévi hálózatot.

A WordNet felhasználhatósági köre igen jelentős, többféle nyelvtechnológiai célra tud nyelvi adatot szolgáltatni. Egy ilyen terület például a főnévi csoportok koreferencia-viszonyainak a feltérképezése (Mihálcz et al. 2007). Annak ellenére, hogy napjainkban hasonló alkalmazásoknál inkább az adatalapú megközelítést használják nagyobb számban és jobb eredménnyel, ők mégis a tudásalapú felől közelítettek, mert nem állt rendelkezésükre megfelelő mennyiségű adat. Többféle tudásra támaszkodnak, a már említett WordNeten kívül használják a MetaMorpho (amelyről részletesebben a 2.3.3.4 pontban lesz szó) mondatelemzőjének morfológiai, szintaktikai és szemantikai jegyeit, illetve elemzéseit; és saját (specifikus) szabályokat is megfogalmaznak, a koreferencia-feloldással elméletben foglalkozó nyelvészek kutatásának felhasználásával. A program 70%-os pontosság körül teljesít, és az arány a jövőben különféle módszerek bevonásával tovább javítható.

2.3.3.3. Szintaxis, szemantika

Ahogy egyre magasabb nyelvi szint feldolgozása a cél, egyre komplexebb a megoldandó feladat, így egyre több részproblémát lehet megfogalmazni, amelyek megoldása közelebb viheti a kutatókat (fejlesztőket) a különféle elemzések számítógépes megvalósításához. A szintaktikai elemzéshez például (a legtöbb megközelítés esetében) szükség van arra, hogy a *főnévi csoportokat felismerje*, azonosítsa és megfelelően annotálja a program. Erre a célra alkalmas például a CLaRK rendszer, amely egy XML alapú korpuszfeldolgozó eszköz. Véges állapotú (reguláris) grammatika alkalmazásával azonosítja és annotálja a főnévi csoportokat (címkéi például: fej, jelző, determináns, birtokos). A magyar nyelvre való alkalmazását Váradi (2003) mutatja be.

Fontos részfeladata a szintaktikai elemzésnek a *tulajdonnév-felismerés* is. Fontosságát csak egy példával szemléltetve: gépi fordításkor a legtöbb esetben nem kell (és nem is szabad) őket lefordítani, ezért fontos, hogy a program valahogyan megjelölje őket. Több tulajdonnév-felismerő rendszer is létezik a magyar nyelvre, például a Farkas–Szarvas (2004) által leírt, statisztikai alapon működő alkalmazás. Gépi tanuló algoritmussal dolgozik, a Szeged Korpuszon lett tanítva és tesztelve. Kis módosításokkal angol szövegeket is kezelni tud (Farkas–Szarvas 2006), illetve – ha megfelelő méretű és kellően annotált tanítókorpusz áll rendelkezésre – bármelyik nyelven elvégzi a feladatot. A rendszer három fő összetevőből

áll: tulajdonnevekre jellemző tulajdonsághalmazok kinyeréséből (mint pl. nagy kezdőbetű, gyakoriság, mondatbeli pozíció, tulajdonnevek listája), statisztikai modellek betanításából és ezek legoptimálisabb összekombinálásából. Az eredményei nemzetközi viszonylatban is kimagaslóak. Ha nem áll rendelkezésre megfelelő mennyiségű kellő mértékben annotált korpusz, részben felügyelt tanulási technikák alkalmazásával is közel hasonló eredmények érhetők el (Farkas 2007).

Egyéb speciális egységek megtalálása és címkézése is hasznosnak bizonyulhat, például a címek elkülönítése a szöveg egyéb mondataitól. A magyar nyelvben nem annyira jelentős a különbség[10], az angol címekre azonban speciális szintaxis jellemző. A sajátos nyelvi szerkezet – mint például hiányos névelőhasználat, egyes segédigék elhagyása – problémát okozhat az elemzésnél. A megoldás a szintaktikai szabályok paraméterezése lehet (Pohl–Ugray 2004). Először fel kell ismerni, hogy az elemzendő mondat egy cím, majd egy „gyengébb" nyelvtant kell írni rá. A felismeréshez korpuszt használnak, a különböző, címekre jellemző jegyek alapján tanítják a programot: például hogy a bekezdés kevesebb szóból áll, nincs a végén írásjel vagy több nagybetűt tartalmaz. Csak az első heurisztikát alkalmazva 90%-os pontosságot lehet elérni a felismerésben, ha azt mondjuk, hogy a kilenc szónál rövidebb bekezdés címnek felel meg.

Többszavas kifejezések megtalálása is fontos részfeladata a szintaktikai elemzésnek (Oravecz et al. 2004). Nem egyértelmű, mely nyelvi elemek tartoznak ebbe a kategóriába, ezért felismerésük sem könnyű feladat. Jellemzőik között szerepel, hogy nem teljesen kompozicionálisak (*húzza a lóbőrt*), formájuk többé-kevésbé rögzített (*hiszi a piszi*), és megsérthetik a szintaxis szabályait (*ezt nevezem!*). Egy kontinuum mentén kell őket elképzelni, melynek egyik végén a teljesen kompozicionális szókapcsolatok, a másikon a teljesen rögzített idiómák találhatók. Azonosításukhoz valamiféle osztályozásra lenne szükség, és akkor egy megfelelően annotált korpuszból talán kinyerhetőek lennének. A probléma jelenleg még távolról sincs megfelelően megoldva.

Végül egy inkább elméleti részproblémája a szintaktikai elemzésnek a *vonzatok és a szabad határozók megkülönböztetése*. Mivel a kérdés elméletileg sem megoldott, ezért a gyakorlatban is többféle megközelítést alkalmaznak rá. Egyes elméletek (alkalmazások) csak a mindig kötelező elemet tekintik vonzatnak, mások minden olyan elemet, amely előfordulhat a predikátummal. Gábor–Héja (2005) nem a predikátumból indul ki, hanem az esetragból: különféle tesztekkel eldönti, hogy a rag jelentése mennyire kompozicionális, az alapjelentésnek mennyire felel meg; ha nagy mértékben, akkor szabad határozónak tekinti, ha csak kis mértékben vagy egyáltalán nem, akkor vonzatnak. Korpusz alapú megközelítések is léteznek a vonzatkeretek megtalálására, a Sass (2006) által bemutatott rendszer például a Magyar Nemzeti Szövegtárból képes vonzatkereteket kinyerni, amelyekhez gyakorisági adatokat is társít. Így a különféle igék vonzatkereteit tartalmazó szótár nem csupán szintaktikai elemzők fontos komponense lehet, hanem egyéb célokra is használható. Segítségével például jelentéseket felsoroló szótárak (mint az Értelmező kéziszótár) használhatóságát javíthatjuk azzal, hogy a gyakoribb jelentéseket szerepeltetjük hamarabb (Sass 2005).

Olyan megközelítés is létezik, ahol a korpuszt nem csupán részfeladatok teljesítésére használják, hanem akár komplett szintaktikai elemzésre is. Hócza (2004) teljes szintaxis felismerésére alkalmaz szabály alapú tanuló módszert, a Szeged Korpusz felhasználásával. A program ismeretlen mondatokhoz hozzá tudja rendelni a legvalószínűbbnek ítélt szintaxis fát. A projekt egy automatikus információkinyerést megvalósító programlánc része, melynek

[10] Apróbb különbségek vannak, például a címek nem tartják be az adott–új információra vonatkozó szabályokat: „Megszökött a terhes nő" – olvashattuk a szalagcímet, pedig nem volt előtte szó semmilyen terhes nőről, így a határozott névelő használata nem volt indokolt.

moduljai különféle természetes nyelvi feladatokat oldanak meg, például szegmentálás, morfológiai elemzés, szófaji egyértelműsítés, szintaxis felismerése, ontológiai elemzés és szemantikai keretek illesztése.

A korábban már említett *Intex* rendszer magyar nyelvű fejlesztésének nem csupán morfológiai, hanem szintaktikai modulján is dolgoznak a kutatók (Váradi–Gábor 2004). Tartalmaz tulajdonnév-felismerőt, és képes részleges szintaktikai elemzésre (a dependencia-viszonyok feltérképezésére), amelynek megvalósítása – akárcsak a morfológiai komponens esetében – szintén véges állapotú technológiával történik. Az elemzés részfeladatai a következők: a frázisok megtalálása és címkézése, a tagmondathatár megtalálása, az igei argumentumszerkezet azonosítása, végül az igéhez tartozó szabad határozók, tagadószók, illetve a mondathatározók megtalálása. A program jelenleg csak a mondatok egy meghatározott csoportjának elemzésére képes, de a rendszer alkalmasnak ígérkezik kutatási és oktatási célokra is.

A MorphoLogic által fejlesztett szintaktikai elemző a *HumorESK* (Kis et al. 2003), amelyet több alkalmazásuk is használ. Adatvezérelt program, vagyis bármely nyelv adataival feltölthető. Alulról felfelé elemez, építi a szintaktikai fát, és közben tárolja a releváns jegyeket. A szabályokat véges mintahalmaz formájában adják meg, amelyek alulspecifikált elemekből épülnek fel. Elképzelhető, hogy egy jelenségre egy általánosabb és egy specifikusabb minta is illeszkedik, ilyenkor a specifikusabb „győz”, ami által csökken a túlgenerálások száma (pl. a *Fekete Péter* lehet jelzős főnév is, de valószínűbb, hogy tulajdonnév, és a program ennek is fogja azonosítani, mert ez a speciálisabb). Az elemző tartalmaz tulajdonnév-felismerőt és NP-felismerőt, továbbá igevonzatok azonosítására is képes (korpusz alapján).

Szintén több alkalmazás használja a *Hunpars* projekt által fejlesztett szintaktikai elemzőt (Babarczy et al. 2005), amely magyar nyelvű mondatok frázisainak és a frázisok közötti viszonyoknak az azonosítását végzi el automatikusan. Ezt frázisstruktúrákat létrehozó szabályok és lexikális függőségi adattárak alkalmazásával éri el (ez utóbbiban tárolja például a vonzatkeretekre vonatkozó megszorításokat). Az előfeldolgozási fázisban morfológiai elemzést végez (a korábban már említett Hunmorph segítségével), majd a szöveget tagmondatokra bontja. Ezután megkeresi az igei frázist és az egyéb régenseket, majd a különböző egyéb szófajú csoportokat. Ezekből a szabályrendszer segítségével összerakható a mondat. Az eredményeik nagyon bíztatóak.

A Hunpars elemzőhöz szemantikai modult is készítenek (Babarczy et al. 2006), amely jelenleg az elemzés kimeneteként kapott frázisstruktúrákat címkézi tematikusan, vagyis a különféle frázisokhoz hozzárendeli a tagmondatbeli szerepüket. A fejlesztés elméleti nyelvészeti alapokon nyugszik, és egy egyszintű, lexikalista nyelvtant is felhasznál (a Konstrukciós nyelvtant). Ahol elégséges, csak a két szemantikai makroszerepet használja (actor és undergoer), csak ott vezet be újat, ahol a kontraszt miatt szükséges. Ezzel elkerüli a sokszor ellentmondásos, megérzésen alapuló hagyományos thematikusszerep-címkéket.

Igazán intelligens nyelvtechnológiai alkalmazásokhoz valamiféle *szemantikai elemzésre* (reprezentációra) is szükség van. Teljes szemantikai jellemzés nemigen adható a szavakról, mert az mindig tovább finomítható lenne. Ezért (a teljesség helyett) a helyességre kell törekedni, illetve a monoton építkezés lehetőségét kell megteremteni (Varasdi–Gábor 2003). Egy lehetséges irány egy ilyen típusú leíráshoz az, ha kiindulunk az ontológiai alaptípusokból, és jelentésposztulátumok segítségével kialakítunk egy hierarchikus lexikonkoncepciót (öröklődéssel). Jackendoff szemantikai alapkategóriáiból indulnak ki, amelyek: thing (dolog), place (hely), path (pálya), event (esemény), action (cselekvés), manner (mód), state (állapot), property (tulajdonság) és amount (mennyiség), de a halmaz szükség esetén bővíthető. A rendszerben könnyedén kifejezhetők olyan általánosítások, mint például: ha egy dolog bejár egy pályát valamilyen esemény keretében, akkor az a dolog az

esemény végén a pálya végpontjában található. Az ilyen típusú információra pedig szükség lehet például akkor, ha egy rendszer kérdésekre válaszol, vagy információt nyer ki különféle szövegekből (intelligens módon, és nem csupán a relevánsnak ítélt mondatok megismétlésével).

Érdekes kísérletről számol be Gábor–Héja (2006a). Az elmúlt évtizedek lexikalista elméletei (pl. LFG) azt feltételezték, hogy az ige jelentése nagymértékben meghatározza azt a szintaktikai környezetet, amelyben előfordulhat. Arra voltak kíváncsiak, hogy ha az igéket jelentéskomponenseik alapján szemantikai osztályokba soroljuk (az angol adatokat – Levin rendszerét – használva), ugyanolyan vagy legalábbis hasonló vonzatkereteket találunk-e az egyes osztályok tagjainál. Vizsgálatukhoz a Magyar Nemzeti Szövegtárat használták, és két igeosztályt néztek meg: a mentális állapotváltozást jelentő igéket és a „spray/load" típusú igéket. A két csoportnál jelentős eltéréseket találtak, de nem annyira egységes a kép, mint az angol osztályok esetében. A mentális állapotváltozást jelentő igék például többféle vonzatkerettel is előfordulnak, amelyek alapján ezek az igék alosztályokba sorolhatók, de kérdés, hogy található-e olyan metapredikátum, amely megléte vagy hiánya alapján megjósolható, melyik alosztályba sorolható az adott ige.

Gábor–Héja (2007) ennek a kísérletnek a „fordítottját" végezte el: az igék szemantikai klaszterezését bővítménykereteik alapján, hogy Levinéhez hasonló igeosztályokat találjanak a magyar nyelvben (mivel a fent említett kísérlet azt mutatta, hogy a magyarra nem alkalmazhatók teljes mértékben Levin angol nyelvre kidolgozott klaszterei). Ehhez a Szeged Treebankot használták, és nem felügyelt tanítási módszert alkalmaztak. Azt az eredményt kapták, hogy az igék egy része besorolható csupán néhány, igen népes osztályba, míg a maradékuk általában egy közeli szinonimájával vagy ellentétpárjával alkot egy csoportot. Következő lépésként megvizsgáltak három osztályt (a létigék, a modálisok és a mozgásigék csoportját), és azt találták, hogy az igeosztályok koherensek szemantikailag, és található olyan jelentéskomponens, amelyben osztoznak.

Összefoglalva tehát: a magyar nyelvre léteznek szintaktikai elemzők, szabály alapúak és korpusz alapúak egyaránt, jelentős részük sekélyelemzést végez, de sok esetben cél a teljes szintaktikai elemzés. Mindegyik alkalmazás frázisstruktúra-nyelvtanon alapszik, ezért bizonyos jelenségek (mint a távoli függőségek) kezelésére csak kiterjesztésekkel alkalmas. A legtöbb program használ különféle jegyeket, amelyeknek unifikálódni kell, sőt némelyikük kimondottan lexikalista szemléletűnek is tekinthető, de (egyelőre legalábbis) nem a lexikalista elméletek és alkalmazások dominálnak a magyar programok között, és nem készül magyar nyelvi komponens a napjainkban annyira sikeres mélynyelvészeti, lexikalista elemzőkhöz és fordítókhoz sem (3. fejezet). A szemantikai elemzés is minimális a magyar nyelvű programokban, gyakorlatilag néhány szemantikai jegy ellenőrzéséből áll, de vannak már törekvések (készülnek elméletek) arra, hogy a jövőben alaposabb szemantikai elemzés legyen végezhető, és akár valamiféle szemantikai reprezentáció is rendelhető legyen az elemzett mondatokhoz.

2.3.3.4. Egyéb alkalmazások, gépi fordítás

Számos egyéb alkalmazás is született a magyar nyelvre, több terülten. Egy ilyen például az *adatbányászat* (text data mining), amelynek célja különféle szövegek rejtett információinak a kinyerése, valami újat létrehozni, új összefüggéseket megtalálni. Mártonfi (2003) például a Magyar értelmező kéziszótár XML-változata segítségével gyakorisági információkhoz, illetve nagy valószínűséggel igaz általánosításokhoz jut (pl. öt fonéma két szótagot alkot, vagy a szaknyelvi címkével ellátott szó főnév, akárcsak a szláv jövevényszavak).

Az egyik leginkább kutatott terület az *információkinyerés* (Information Extraction, IE), amely nem keverendő össze az internetes keresők által megvalósított információ-

visszakereséssel (Information Retrieval, IR), ahol csupán a releváns dokumentumot kapjuk meg. Egyre nagyobb az igény arra, hogy a gép megtalálja a felhasználó számára releváns információkat a szövegben, és el tudja tárolni azokat valamiféle adatbázisban; hiszen a felhasználónak sokszor nincs ideje arra, hogy minden kapcsolódó anyagot végigolvasson. Rengeteg ilyen témájú cikk található a magyar nyelvvel kapcsolatban is, több kutatás célozza meg ezt a területet, így többféle szoftver készült már erre a célra.

Prószéky (2003b) bemutatja a *NewsPro*-t, amely elektronikus szövegekből nyer ki információt tartalomelemzési eljárások kidolgozásával úgy, hogy a kinyert információt aztán adatbázisba tölthető tudássá tudja alakítani. Nem a releváns szövegeket tartalmazza tehát, hanem a releváns adatokat. A programnak nem kell teljes elemzést végezni ahhoz, hogy végrehajtsa ezt a feladatot, de morfológiai és valamilyen szintű szintaktikai elemzésre szükség van, és hasznos, ha az elemzés során szemantikai információkra is lehet hivatkozni. A program tanításához egy megfelelően annotált korpuszra volt szükség, és így (első alkalmazási területként) gazdasági rövidhírekből képes információt nyerni. A program egy lehetséges továbbfejlesztéséről számol be Gábor et al. (2004). Hogy ne csupán igei állítmánnyal kifejezett eseményeket ismerjen fel a program, kidolgozták, hogyan lehetne igeneves szerkezetekből is kinyerni a releváns információt. Egy előfeldolgozó modul teljes propozícióvá alakítja az igeneves szerkezeteket, így azokra is tud az alkalmazás szemantikus keretet illeszteni, így az információ kinyerhetővé válik. A szabályok elkészítéséhez és teszteléséhez (és minden egyéb feladathoz is) az Intex korpuszfeldolgozó eszközt használták.

Egy másik rendszer ugyanerre a célra, az Alexin et al. (2004) által bemutatott *IEToolChain*, amely egy információkinyerési kutatásokat támogató programcsomag. Több modulból áll, melyek közül a legfontosabbak: a szegmentáló, a morfológiai elemző, a szófaji egyértelműsítő, a felszíni szintaktikai elemző, a szemantikai bővítménykezelő, az eseménymintákat felismerő mintaillesztő és a webes megjelenítő. A modulok fejlesztéséhez a Szeged Korpuszt használták, és egy egyszerű felépítésű ontológiát is alkalmaztak. Első lépésként üzleti rövidhírekre alkalmazták a programot, amelyen 70% körül teljesít, de dolgoznak a hibák kiküszöbölésén és így az arány javításán. A szemantikuskeret-illesztő modulról (Frametagger) részletesebben Farkas et al. (2004) számol be. Az előre definiált keretek eseményeket írnak le azok szereplőinek szintaktikai és szemantikai megkötésein keresztül. Az információ kinyeréséhez meg kell találni a keret célszavát (pl. *megvásárol*) és a szereplőket (pl. ki, mit, mennyiért), amelyekhez pozícióbeli megkötések is tartozhatnak, illetve prioritási értékkel is meg lehetnek jelölve, amely megmutatja, hogy a kérdéses elem mennyire fontos szereplője az adott keretnek.

Multimodális és multimédiás rendszereket szintén fejlesztenek a magyar nyelvre is, ahol nem csupán a szöveg, hanem a gesztusok és az érzelmek (arc) modellálása is cél, hiszen az emberi kommunikációban ezek a faktorok nagyon jelentős szerepet játszanak. Tatai–Laufer (2003) bemutatja az általuk fejlesztett beszélgetőrendszerbe integrált érzelmi elemző és generáló alrendszert (*Gala*), amely a válaszadás során az érzelmet is figyelembe veszi. A megfelelő reakciókat korpuszból tanulja, előtte azonban sok kézzel történő rögzítés szükséges. Az eredményeik megfelelnek az elvárásoknak: az egyszerűbb érzelmek felismerése és generálása viszonylag jó pontosságú, a bonyolultabbak azonban még gondot okoznak. Egy másik multimédiás alkalmazás egy magyar nyelvű, vizuális szövegfelolvasó, a Czap–Mátyás (2003) által bemutatott „beszélő fej”. Az animáció háromdimenziós fejmodell mozgatásán alapul, és a természetesség javítása érdekében álvéletlen fejmozgás, pislogás és szemöldökmozgás is bele lett programozva. Viszonylag jó pontossággal képes az alapérzelmek produkálására is.

Az egyik legfontosabb, és legrégebb óta kutatott nyelvtechnológiai alkalmazás a *gépi fordítás*. Erre a célra is több program készül, leginkább a magyar–angol nyelvpárra. A legsikeresebbnek a MorphoLogic által fejlesztett *MetaMorpho* tekinthető. A program

angolról magyarra fordító verziójáról, amelyet 2000-ben kezdtek fejleszteni, Tihanyi (2003) számol be részletesebben. A MetaMorpho egy olyan alapvetően szabály alapú rendszer, amely mintapárokat tartalmaz, ahol a minta illeszkedhet nyelvi elemek nagyobb osztályára is (ekkor nyelvtani szabálynak tekinthető), csupán néhány lexikai egységre is (ekkor az elem egyedi tulajdonságát rögzíti), de akár egy egész mondatra is, ekkor fordítómemória-szerű használatot nyerünk. Így a program valójában nem csupán a szabályalapú (RBMT), hanem a minta alapú (EBMT) fordítórendszerek közé is besorolható. A program környezetfüggetlen szabályokat tartalmaz praktikus kiegészítésekkel (nem célja tehát valamely elmélet szigorú implementálása). Az elemzés balról jobbra halad és lentről felfelé építkezik, eredménye a felcímkézett fa. Generáláskor fentről lefelé járja be a fát, miközben az elemzőszabályok generáló párját alkalmazza. A MetaMorpho kezdetben egyetlen komplex programként működött, de később modularizálták, így született meg a (korábban már említett) Humor és HumorESK morfológiai, illetve szintaktikai elemző program. A program korai verzióját egy Lisp alapú szakértői rendszerben írták, majd később – a hatékonyság növelése érdekében – áttértek a C++ fejlesztőkörnyezetre. Folyamatosan bővítik a programhoz tartozó lexikont, és beépítettek egy guessert is (Novák et al. 2003). Jelentés-egyértelműsítő modul is készült az alkalmazáshoz (Mihálcz 2004), amely lokális és globális jegyeket is használ (lokális pl. a szó környezete, globális pl. a szöveg témája). 75% felett teljesít, ami nemzetközi viszonylatban is nagyon jónak számít.

Néhány évvel később a MorphoLogic megkezdte a magyarról angolra történő fordítás fejlesztését is, amelyet Merényi (2005) mutat be. Akárcsak a másik irány esetében, ennél a programnál is több különböző formalizmust használnak. Az *mmo* az alacsony szintű alapformalizmus, amely egy elemzősorból és egy vagy több generálósorból áll, amelyek különféle (lexikális, morfológiai, szintaktikai) jegyekre is hivatkoznak. Ha az elemzősor hiányzik, *mmc* formalizmusról beszélünk; erre akkor van szükség, amikor olyan elemet kell megjeleníteni a célnyelvi mondatban, aminek nincs forrásnyelvi megfelelője. Végül a harmadik típusú formalizmus az *mmd*, amely a könnyebb adatbevitelt hivatott szolgálni: nem tartalmazza a sok technikai részletet, amire az elemzéshez szükség van, ezeket (például öröklődéssel) kiszámolja a program, majd hozzáadja a bevitt *mmd* sorhoz, így jön létre a minden információt tartalmazó *mmo* alapformalizmus. A magyar mondat környezetfüggetlen grammatikával való elemzése több tényező miatt sem egyszerű. Ilyen a viszonylag szabad szórend, a szétszakított összetevők vagy a zéró formában is megjelenhető vonzatok esete. Az első problémát a program úgy oldja meg, hogy bármilyen szórendi variánst elfogad, és vállaltan nem tesz különbséget például a fókuszos és a nem fókuszos mondatok között (mindegyiket semlegesre fordítja). Az utóbbi kettőt pedig az úgynevezett kiterjesztett argumentum bevezetésével oldja meg, amely egy absztrakt, csupán jegyeket tartalmazó elem, felszíni alak nélkül (viszont a generálásnál tekintetbe kell venni). A programot folyamatosan fejlesztik, hogy egyre több nyelvi jelenséget legyen képes kezelni, melyek közül már (többé-kevésbé) megvalósult a szabad határozók, a vonatkozó mellékmondatok, a névszói állítmány és a melléknévi igeneves szerkezetek kezelése is (Tihanyi–Merényi 2006).

Az angol nyelvű mondat generálásakor problémát okozhatnak a magyar mondatokban előforduló zéró névmások, hiszen a legtöbb esetben nem tudjuk, milyen testes névmást kellene használnunk, az élő (emberi) szereplőre utaló *he/she* vagy az élettelen (nem emberi) szereplőre utaló *it* névmást. Erre a problémára kínál egy megoldást Sass (2007). Viszonylag jó eredményt ad egyetlen faktor figyelembe vétele: ha a korpuszban az adott ige egyes szám harmadik személyű formájának gyakorisága 90%-nál nagyobb, akkor valószínűleg élettelen alannyal áll (pl. *elromlik*). A baj ezzel a módszerrel az, hogy főleg a kellemetlenebb irányba hibázik, vagyis élő helyett élettelennek ítél bizonyos szereplőket. Kis korrigálással egy pontosabban működő algoritmust nyerhetünk, amely tekintetbe veszi az igével használt vonatkozó névmásokat is: ha talál *aki* névmást, nem fogja az alanyt élettelennek tekinteni.

Ezzel az algoritmussal az alkalmazás az élettelen keretek 70%-át megtalálja, és szinte soha sem hibázik a rosszabbik irányba. Az így kapott adatok bekerültek a MetaMorpho által használt lexikonba mint alapértelmezett értékek, melyeket a program elő tud venni, ha a forrásnyelvi mondat nem tartalmazza a kívánt információt.

A MetaMorpho lexikonja szemantikai jegyeket is tartalmaz, hogy a többértelműségeket kezelni tudja (Orosz 2006). Igei és főnévi jegyeket egyaránt használ a program, ebben a cikkben a főnévi jegyekről olvashatunk részletesebben. Kéttípusú jegy felvételét találták szükségesnek. Az egyik típusba tartozik a megszámlálhatóság és a nem, amelyeket a szintaktikai komponens használ, leginkább generáláskor (magyarról angolra fordításkor szükségesek ezek a jegyek). A másik típusba pedig a vonzatkeret-leírásoknál használt jegyek sorolhatók, amelyeket egyértelműsítésre használ a program, hogy a megfelelő jelentést ki tudja választani; erre inkább elemzéskor van szükség. A jegyek a következők: abstract, animate, bodypart, dynamic, company, currency, human, mass, measure, time és weather. Például ha a *felad* tárgya nem absztrakt, akkor annak angol megfelelője a *post* ige, míg az OBJ(…abstract=YES) esetben a *give up* a keresett célnyelvi kifejezés. Egy másik, igen szemléletes példa az *Anyák kiborultak* mondat, amelynek elvileg négy lehetséges értelmezése (és így fordítása) lehetne, de ebből csak kettőt fog a program felajánlani, ahol az *anya* főnévi jegye megegyezik a *kiborul* tárgyára vonatkozó szemantikai megszorítással (vagy mindkettő élő, vagy egyik sem). A jegyek definiálását (hogy pontosan milyen főnevek tartozzanak az adott jegyekhez) a WordNet ontológia segítségével végezték (az intuitív jegydefiníciók figyelembe vételével).

A már említett *NooJ* (korábban Intex) rendszer a morfológiai és a szintaktikai elemzés mellett részleges gépi fordítást is képes adni, amelynek előnye, hogy gyors, mert (akárcsak a többi nyelvi szint esetében) véges állapotú technológiával (transzducerrel) dolgozik (Váradi 2006). Alulról felfelé haladó minta alapú rendszernek tekinthető, ahol a minta fogalmába tartoznak az egyedi lexikai egységek is és a szóosztályokra jellemző szintaktikai szabályok is. A fordítást lokális grammatikák kétnyelvű felhasználásával éri el. A rendszer robusztusságát az adja, hogy folyamatosan próbálja illeszteni a lokális grammatikákat, és ha egyiket sem sikerült, a szavak célnyelvi megfelelőjét akkor is visszaadja. A főnévi csoportokat teljes egészében fordítja, ami a magyar nyelv esetében a kötött szórend miatt egyszerűbb, mint a mondatfordítás, (egy-két kivételtől eltekintve, mint az igeneves szerkezetek) véges állapotú technológiával kezelhető. A rendszer alapja a kétnyelvű lexikon, amely valójában az egynyelvű szótárak összekapcsolásával áll elő, és ahol egy szótári elem több szóból is állhat (pl. *házi feladat*).

Egy harmadik gépi fordító rendszer a *Hunglish* nyílt statisztikai magyar–angol nyersfordító (Halácsy et al. 2004). Nem céljuk a jó minőségű fordítás, csupán annyi, hogy az angolul nem vagy csak kevéssé tudó felhasználók is hozzáférjenek az interneten fellelhető angol nyelvű anyagokhoz. A fordításhoz a legfontosabb eszköz egy angol–magyar szótár (párhuzamos korpusz), amelyet korábban már röviden bemutattam (Halácsy et al. 2005). A szótári többértelműség problémájának megoldására rejtett Markov modellt alkalmaznak.

Napjaink legelterjedtebb fordítástámogató eszközei a *fordítómemóriák* (Translation Memory, TM). Ha rendelkezésre áll megfelelően illesztett párhuzamos korpusz, akkor nagyon jó eredményeket lehet velük elérni. A magyar nyelvre is készülnek TM-alapú alkalmazások, ahol a legnagyobb kihívást a mondatok (mondatrészek) szinkronizációja jelenti. Erre Pohl (2003) egy olyan hibrid megoldást mutat be, amely a két, szinkronizációs célokra leggyakrabban használt módszert ötvözi. Egyrészt használja a szövegegységek hosszának összehasonlítását, ami elég robusztus, de rosszul kezeli a beszúrásokat és az elhagyásokat; másrészt alkalmazza a horgonykeresést is, vagyis próbál szavakat (kifejezéseket) megfeleltetni egymásnak a két szövegben, amivel (önmagában) csak részleges szinkronizáció végezhető. Teljesen automatikus módszerekkel azonban nem érhető

el tökéletes eredmény. A Pohl (2004) által javasolt rendszerben minimális emberi beavatkozással (korrektor) és a folyamat iteratívvá tételével a pusztán gépi szinkronizálásnál jobb eredmények érhetők el.

A MetaMorpho szabályalapú fordítóprogramra támaszkodik a *(Meta)Morpho TM* intelligens fordítómemória (Hodász–Grőbler 2003), amely a szónál nagyobb, de a mondatnál kisebb egységek feldolgozását, tárolását és összeillesztését végzi. A szabálybázis homogén, vagyis nem különbözteti meg a szabályt a lexikai tulajdonságtól (mindent egységesen szabályként kezel, ami több vagy kevesebb elemre tud illeszkedni), ezért a fordítómemória egyaránt (és egységes módon) képes egyetlen elem vagy akár egy mondatváz fordítására is. Adatbázisában tárol tehát egész mondatpárokat, főnévi csoportokat és (NP nélküli) mondatvázakat is. A főnévi csoportok szinkronizációja mind közül a legnehezebb. Elméletben megoldható lenne (viszonylag jó pontossággal és fedéssel) a fordítóprogram által használt magyar nyelvtan alkalmazásával, viszont ez a módszer (egyelőre) túlságosan időigényes (Pohl 2006). A cikkben többféle megközelítést vet össze egymással a szerző, majd egy gépi tanuló algoritmust ítél a legjobbnak.

Összefoglalva a gépi fordítás magyarországi helyzetét azt lehet mondani, hogy (az angol–magyar nyelvpárra) létezik több működő alkalmazás (szabály alapú és példa alapú is), de ezek vagy nem is célozzák, vagy a folyamatos fejlesztés ellenére sem érték még el a jó minőségű fordítás szintjét; arra azonban mindenképpen alkalmasak, hogy megkönnyítsék az angolul nem (vagy nem jól) értő felhasználók helyzetét. Több közülük (kisebb-nagyobb mértékben) lexikalista, de (akárcsak az elemzés esetében) nem tekinthetők szigorúan egyik lexikalista elmélet implementációjának sem. Úgy tűnik tehát, hogy lenne igény arra, hogy egy alaposabb, több faktort számításba vevő elemzést használó rendszer szülessen a magyar nyelvre (is), amely azokhoz a mondatokhoz is képes lenne jó minőségű fordítást rendelni, amelyek a jelenlegi programok számára még gondot okoznak. Erre a feladatra pedig napjainkban a mélynyelvészeti és azon belül is a lexikalista alapú megközelítések tűnnek legalkalmasabbnak.

3 Lexikalista elméletek

Az elmúlt évtizedek elméleti sikerei után napjainkban a nyelvtechnológia területén is egyre sikeresebbek a különféle lexikalista elméletek. Mivel céljuk a minél részletesebb elemzés, ezért alkalmasabbak intelligens nyelvtechnológiai célokra, mint a sekélyelemzéssel működő rendszerek. A számítástechnika és a különféle statisztikai módszerek fejlődése lehetővé tette, hogy a nagyobb erőforrásigény ellenére hatékonyak tudjanak lenni az alaposabb nyelvi elemzést végző rendszerek, a megfelelően annotált korpuszok létrehozása pedig a rengeteg szükséges adat rögzítését tudja könnyebbé és gyorsabbá tenni. A lexikalista megközelítés sikeressége igazolni látszik, hogy érdemes kipróbálni annak a legvégsőkig vitt változatát – a totális lexikalizmust – elméletben és gyakorlatban egyaránt. Ebben a fejezetben a lexikalizmus eszméjének ismertetése után bemutatom a legjelentősebb lexikalista elméleteket, és az ezek implementálásával készült legsikeresebb elemző és gépi fordító programokat.

3.1 Lexikalizmus

A nyelvelmélet egyik kiemelt célja, hogy általánosításokat fogalmazzon meg kifejezések osztályaira, és így minél pontosabban jellemezze a nyelvi rendszert. A nyelvészeti elméletek különböznek abban, hogyan kellene, illetve hogyan lehetne ezeket megragadni. A korábbi elméletekben kis szerep jutott a lexikonnak, pusztán egy listának tekintették, amely a szavak formáját és a ki nem számítható tulajdonságaikat tartalmazza. Úgy vélték, a nyelvészeti általánosítások transzformációsan fejezhetők ki legjobban.

Chomsky és követői az ötvenes évektől azon dolgoztak, hogy újraíró és transzformációs szabályokkal írják le a nyelveket, minimális szerepet hagyva a lexikonnak. Minden nyelvi jelenséget ebben a szellemben kezeltek, hogy mi egy mondat alanya, vagy mi történik passzivizációkor stb. Az angol nyelvre a módszer többé-kevésbé működött is – annak konfigurációs volta miatt –, de amikor olyan nyelvre próbálták alkalmazni, amely a grammatikai viszonyokat inkább a morfológiában kódolja (és ahol a szórend sokkal kötetlenebb), nehézségekbe ütköztek; illetve felismerték, hogy egy szótárközpontúbb megközelítéssel eredményesebben (hatékonyabban) lehetne leírni az ilyen típusú nyelveket.

Elméleti szempontból is problémás a chomskyánus generatív grammatika. A transzformációk miatt az építkezés nem monoton (struktúraváltoztató műveletek), ezáltal számítógéppel is nehezen modellezhető; és a Chomsky-féle nyelvtan-hierarchiában a 0. (illetve az 1.) típusba fér csak bele, holott bizonyított, hogy a természetes nyelvek az 1. és a 2. típus között helyezkednek el (közelebb a 2. típushoz). Az elmélet további gyenge pontjai, hogy üres kategóriákat is feltételez, nem érvényesül a lexikai integritás elve (vagyis hogy a szintaxis nem láthatja a szavak belső szerkezetét), és a konfigurációnak (hierarchiának) központi szerepe van (Trón 2001).

A 70-es évektől egyre inkább teret hódítottak a nemtranszformációs elméletek. Az általánosításokat transzformációk helyett ún. lexikai redundancia szabályok segítségével fejezték ki (pl. aktív és passzív forma között)[11]. A lexikon így többé már nem a nyelvtan azon része, amely kivételes, idioszinkratikus, így érdektelen. Hirtelen a középpontba került, mint a szintaktikai általánosítások fontos kifejezője.

---

[11] „A lexikai szabály olyan művelet, amely (esetlegesen morfológiai szabály működésével egyidejűleg) jelentéstani változtatást végezhet a szótári jelentésen.” (Kálmán 2001: 47)

Több lexikalista elmélet is született, amelyek legfontosabb jellemzője, hogy megszorítás alapúak, vagyis nem egy sikeres deriváció, hanem egy kielégíthető követelményhalmaz a grammatikus nyelvi formák ismérve. A legfontosabb megszorítás alapú elméletek az LFG, a GPSG, majd „utódja", a HPSG, a kategoriális nyelvtanok (CG) és a konstrukciós nyelvtan. A lexikon irányába való erőteljes elmozdulást jól mutatja, hogy egyes – korábban a lexikonra kis hangsúlyt fektető – elméleteknek, mint például a TAG-nek (Tree Adjoining Grammar) elkészült a lexikalizált változata (LTAG, Lexicalized Tree Adjoining Grammar). A nyelvtan „atyja", Aravind Joshi, napjainkban már szintén azt vallja, hogy „Grammar ≈ Lexicon" (Joshi 2003).

Egyes elméletek még ennél is tovább mentek: Karttunen (1986) radikális lexikalizmust hirdetett: amellett érvelt, hogy ha elég gazdag a lexikon, akkor pusztán függvényalkalmazás és unifikáció segítségével is összeépíthető a mondat, nincs szükség egyéb szintaktikai műveletre, továbbá a frázisstruktúra-fa sem fog többé valódi információt hordozni. Ennek a lexikalizációs folyamatnak a „végpontja" a GASG, amely megvalósítja a totális lexikalizmust, ahol annyira gazdag a szótári komponens, hogy már a függvényalkalmazás műveletére sincs szükség, és frázisstruktúra már egyáltalán nem is épül.

3.1.1 A lexikalista elméletek jellemzői

A lexikalista elméletek négy legfontosabb tulajdonsága Trón (2001) alapján a következő:

(1) *Deklarativitás* (megszorítás alapúság), vagyis nem átalakító jellegű szabályokkal, hanem jólformáltsági megszorításokkal dolgozik. Nem a levezetés (deriváció) szabja meg, mi lesz grammatikus; jól formált kifejezés az, ami minden megszorítást kielégít. Kérdés, hogy milyen jellegű megszorításokat enged meg az adott nyelvtan. Annak ellenőrzésére, hogy a különféle jegyekkel bíró elemek kombinálódhatnak-e az adott mondatban, az unifikáció mechanizmusát használja. Az unifikáció (számítástechnikában használatos) definíciója a következő[12]:

> **Unifikáció**: A mintaillesztés általánosítása. Amikor két kifejezést unifikálni kell, össze kell őket hasonlítani. Abban az esetben, ha mindkettő konstans, az unifikáció akkor sikeres, ha értékeik megegyeznek, egyébként sikertelen. Ha az egyik változó, akkor az értéke a konstans értékével fog megegyezni, és az unifikáció sikeres lesz. Ha mindkét kifejezés valamilyen (több részből álló) szerkezet, akkor részeiket páronként (rekurzíven) unifikáljuk, és a művelet akkor lesz sikeres, ha minden rész sikeresen unifikálódik. Az unifikáció eredménye tehát vagy hiba vagy siker, és a művelet az utóbbi esetben visszaadja a változók lehetséges értékeit tartalmazó halmazt, amit unifikálónak (unifier) nevezünk. Sok esetben több ilyen megoldás is lehetséges, de legfeljebb egy „legáltalánosabb unifikáló" (most general unifier) lesz, a többi tartalmazni fog egyéb lehetséges változólekötéseket (amik az eredetiben változók maradnak).

(2) *Egyszintűség*, vagyis elfogadja, hogy különböző nyelvi reprezentációs szintek lehetnek, de azt vallja, hogy egy jel jólformáltságát ezek a (különböző szinteken lévő) megszorítások egyszerre szabják meg, egyszerre lépnek életbe. Az LFG-ben és a HPSG-ben is igaz, hogy minden megszorításnak egyszerre kell teljesülnie, vagyis nem lehetnek köztes, rosszul formált reprezentációk (monoton építkezés), és egyik megszorítás sem lehet erősebb, mint a másik (Kálmán et al. 2002). Az egyszintűség mellett több érv is felhozható. Például

[12] Forrás: http://www.definethat.com/define/5293.htm (Denis Howe)

hogy a pszicholingvisztikai kutatások is inkább ezt támasztják alá, vagy hogy így egységes formalizmus használható az ábrázolásban (Trón 2001). Kálmán et al. (2003) szerint pedig a legnagyobb probléma a hagyományos generatív modellekkel pontosan a modularitásuk: nehéz például jelentéstani információ nélkül igazán alapos és megbízható szintaktikai elemzést végezni, vagy ha a jelentéstant szétválasztjuk a pragmatikától, akkor nagyon bonyolult leírást kellene alkalmaznunk, stb. Olyan szintű együttműködést kellene elvárnunk a moduloktól, hogy az már nem is lenne modularitás, ezért javasol inkább (implementációs célokra is) egyszintű grammatikát.

(3) *Lexikai integritás* elvének betartása, vagyis hogy a szavakat egy független lexikai modul állítja elő, belső szerkezetük a szintaxis számára nem hozzáférhető. Ezt egyes elméletek nem tűzik ki célul, azok, ahol bármely morféma (egy toldalék is) önálló követelményekkel léphet fel, és nem az történik, hogy rátapadnak egy tőre, ekkor egyesülnek a jegyeik és az elvárásaik, így utána a morféma a szintaxis számára önállóan már nem létezik. Egy ilyen – a lexikai integritás elvét be nem tartó – elmélet például a Gambäck (2005) által leírt rendszer, amelynek célja a szemantika minél pontosabb és hatékonyabb kezelése. Minden morféma külön lexikai tétel (az affixumok is), és bármely állítást hordozó lexikai egység hozzájárulhat a szemantikai elemzéshez (és reprezentációhoz). Előnye, hogy egységesen tudja kezelni a nyelven belüli és a nyelvek közötti különböző szinteken (önálló szó vagy affixum) megjelenő elemeket. Mindezt a japán morfológián érzékelteti: azáltal, hogy az igei affixumokat is külön tételként tárolja az ún. szemantikai lexikonban, egységesen tudja kezelni őket és a posztpozicionális partikulákat. A lexikai integritás elvét elutasító elmélet a GASG morféma alapú verziója is, amelynek tehát szintén nem okoz problémát, hogy különböző nyelvekben különböző helyen található a szószint. Így egységes kezelést biztosít például a magyar és az angol műveltetésre (amelyet a magyar nyelvben jellemzően képzővel, az angolban pedig külön szóval fejezünk ki), vagy arra, hogy az alany számát és személyét egyes nyelvek névmással, míg mások (alapvetően) ragokkal fejezik ki[13].

(4) *Hierarchikus lexikon* alkalmazása, vagyis hogy a szótár elemei típushierarchiába (öröklődési hierarchiába) rendeződnek, amely a tudásreprezentációk egyik fajtája. „Fent" találhatók az általános elemek, „lent" pedig az egyediek. A csomópontok (fogalmak) leírása általában attribútum–érték párokkal történik. Az általánosabb fogalom jegyei öröklődnek a speciálisabbakra, de azok értékei eltérhetnek, felülírhatják azokat (kivételek). A lexikon tehát strukturált, nem csupán idioszinkráziák tára. Ez a tulajdonság sem igaz minden lexikalista elméletre, nem teljesül például a kategoriális és a konstrukciós nyelvtan (és elméletileg a GASG) esetében sem.

További fontos jellemzője a lexikalista elméleteknek, hogy az igei elemnek központi szerepe van, vagyis az ige és annak vonzatkerete határozza meg a mondat vázát. Egyes elméletek eltérnek abban, hogyan kezelik ezt. Például az LFG-ben az egyes igék különféle vonzatkereteit, illetve az egymással klasszikusan transzformációs kapcsolatba hozott igék vonzatkereteit lexikai műveletek hozzák létre egymásból. A HPSG-ben a lexikai műveletek megfelelői maguk is argumentumszerkezettel bíró elemek. A konstrukciós nyelvtan még messzebb megy, szerinte ezek a „lexikai" tételek maguk is különféle konstrukciók szimultán megvalósulása eredményeként állnak elő (Kálmán et al. 2002). A GASG, akárcsak a függőségi nyelvtanok, azt vallja, hogy az építkezést az ige „vezérli", nála kell kezdeni a mondat (viszonyainak) „kibontását".

---

[13] Az ilyen „szigorúan" morféma alapú elméletekről azt is mondhatjuk, hogy a lexikai integritás elvét nem nem tartják be, hanem az számukra semmitmondó, hiszen a szintaxis itt sem lát bele a valós követelményekkel rendelkező lexikai egységek belső szerkezetébe, csak azok jelen esetben nem szavak (amiket tovább lehet még bontani jelentéssel bíró egységekre), hanem tovább már egyébként sem osztható morfémák.

3.1.2 Lexikalizmus a nyelvtechnológiában

Egyetértés mutatkozik abban, hogy ha a cél egy igazán intelligens elemző vagy gépi fordító rendszer létrehozása, nem elégségesek a sekélyelemzést végző programok. Többek között azért sem, mert kevéssé képesek szemantikai elemzésre, illetve arra, hogy mondatokhoz (vagy még inkább szövegekhez) szemantikai reprezentációt rendeljenek. Alacsonyabb nyelvi szintekre működnek a különféle statisztikai módszerek, de a tudatos nyelvi működés modellálására nem bizonyultak kellően eredményesnek (Kálmán et al. 2003). Több rendszer is használ különféle szemantikai jegyeket az elemzés során, de a jelentés kinyeréséhez egyéb tényezők is szükségesek, nem elégséges egy jelentéstani jegyekkel is ellátott szintaxis. Ha egy rendszer igazán használható szemantikai elemzést akar végezni (illetve igazán alapos jelentéstani reprezentációt akar a mondathoz rendelni), tekintetbe kell vennie a pragmatikai tényezőket, az előfeltevéseket és elvárásokat, illetve a diskurzushelyzetet. Fontos továbbá, hogy a nyelv jelenségkörének teljes leírására kell törekednie, az általános szabályszerűségek mellett az egyedi esetekről is számot kell adnia (Trón 2001).

Az elmúlt évtizedekben a számítógépes nyelvészeti kutatások négy erős trend köré csoportosultak, mind az unifikációs megközelítésekben, mind úgy általában a számítógépes szemantikában (Gambäck 2005):

(1) A szemantikai információból a lehető legtöbbet a lexikonban tartani (és nem a nyelvtani szabályokban, mint ahogy tradicionálisan szokás). E mögött az a törekvés áll, hogy a nyelvtani szabályok a lehető legegyszerűbbek legyenek és számuk is a lehető legkevesebb legyen. Ez természetesen bonyolultabb lexikához vezet, amit nehéz (szinte lehetetlen) megkülönböztetni a nyelvtantól.

(2) Kompozicionálisan építeni fel a komplex struktúrákat. A kompozicionalitást lehet szigorúan értelmezni: egy frázis interpretációja nem más, mint az őt alkotó kisebb frázisok interpretációinak (logikai) összege. Egy ebben az értelemben kompozicionális szemantikai formalizmus monoton kell, hogy legyen, hiszen ez a definíció nem engedi meg a romboló változtatást. Így tehát minden terminális csomópont minden információja átadódik a felsőbb csomópontoknak. A kompozicionalitást általában ennél tágabban értelmezik, nem összeg, de még mindig függvény: a frázis interpretációja nem más, mint fejének interpretációja módosítva szabad bővítményei interpretációjával. Ha ez unifikációsan valósul meg, továbbra is monoton építkezést kapunk.

(3) A döntéseket minél későbbre halasztani, amely alapvetően a többértelműség kezelésére vonatkozik. Több okból kaphatunk egynél több jelentést: vagy magának a szónak van több jelentése, vagy egynél több szintaktikai szerkezet rendelhető egy kifejezéshez, vagy a hatóköri viszonyok nem egyértelműek. Ezek a tényezők csökkentik a hatékonyságot, merthogy mindegyik lehetőséget tárolni kell, számolni vele, amíg ki nem derül valamelyikről, hogy az adott kontextusban nem megfelelő. A problémára a legjobb megoldás az alulspecifikált reprezentáció, vagyis hogy nem egy konkrét interpretációt kódolunk (amiből esetleg többet is kell), hanem azok egy halmazát jelentő alulspecifikált formulát. A megoldás mellett pszicholingvisztikai érvek is szólnak, valószínűleg az emberi agyban is így működik a többértelműség kezelése. A szemantikai reprezentációt tehát olyan logikai nyelven kell írni, amely megengedi az alulspecifikációt.

(4) Annak kutatása, mely pontokon és pontosan hogyan lehet a különféle sekélyelemző és statisztikai módszereket alkalmazni, mert az „olcsóbb", mint a mélyelemzés. Az igazán hatékony lexikalista elemzők, úgy tűnik, megtalálták a kérdésre a választ, mert elemzőik egyre hatékonyabbak úgy, hogy pontosságban bőven túlszárnyalják a mélyelemzést nem végző rendszereket.

A gépi fordítás megvalósítására napjainkban (egyelőre) a különféle példa alapú és statisztikai megközelítések a legelterjedtebbek, de ellenük szól, hogy a legtöbb nyelvre nem létezik igazán jól használható párhuzamos korpusz (Bond et al. 2005). A pontosságon túl egy

további érv a szabály alapú gépi fordítás mellett az újrahasznosíthatóság: a gépi fordítás fejlesztésének eredményei más nyelvtechnológiai alkalmazásokban is használhatók, illetve más területek eredményeit a gépi fordítás is hasznosíthatja. Ezért ha olyan gépi fordító rendszer fejlesztése a cél, ami bármely két nyelv esetében[14] jó minőségű és pontos fordítást ad, akkor mélyelemzést végző szabály alapú rendszer alkalmazása tűnik a legjobbnak.

Napjainkban nem csupán a nagy pontosságú elemzés és az egyéb intelligens alkalmazások, hanem a jó minőségű gépi fordítás céljára is az unifikációs mechanizmusokat használó lexikalista elméletek látszanak a legmegfelelőbbnek. Róluk bebizonyosodott, hogy nem csak a konfigurációs nyelveket (mint az angol) kezelik hatékonyan, hanem a magyarhoz hasonló szabadabb szórendű nyelveket is. A jó minőségű fordításhoz szükségesnek látszó szemantikai reprezentáció hozzárendelésében is jobb eredményeket érnek el, mint a csak frázisstruktúrára épülő nyelvtanok. Továbbá jelentős részük használ öröklődési hierarchiákat, amelyeket a matematikusok részletesen tanulmányoztak az elmúlt időszakban, és kialakultak már a legfontosabb módszerek formális kezelésükre. A számítógépes nyelvészek pedig azóta több hatékony mondatelemző és mondatlétrehozó rendszert készítettek ilyen hierarchiákra támaszkodva (Kálmán 2001).

A lexikalista nyelvtanok közül a legeredményesebbek LFG vagy HPSG formalizmust használnak. Számos nyelvre léteznek már nagy lefedettségű elemzőik, amelyek nagyon jó minőségű és alapos elemzést adnak, továbbá szemantikai reprezentációt is társítanak a mondatokhoz. Különböző nyelvekre alkalmazzák ugyanazt az elméleti keretet, így tudják elérni, hogy a különböző nyelvű elemzések szinte teljesen párhuzamosak legyenek, ami jelentősen megkönnyíti az elemzőkre épülő gépi fordító rendszerek fejlesztését. Több nyelvre is kidolgoztak már működő gépi fordító rendszereket, amelyek ezeket az elemzőket (és generálókat) használják. Legtöbbjük transzfer alapú, így minden nyelvpárra és mindkét fordítási irányra külön transzfer-komponenst kell kidolgozni. Legnagyobb részük még kísérleti fázisban tart, de eredményeik nagyon ígéretesek.

A következő alpontban bemutatom a legfontosabb lexikalista elméleteket, külön kitérve a nyelvtechnológiai eredményeikre.

3.2 Elméletek és implementációik

A 70-es évektől tehát egyre több megszorítás alapú nyelvtan született. Középpontjukban a lexikon áll, de frázisstruktúrát is használnak, pusztán a transzformációk létét tagadják, illetve egyéb tényezők (pl. morfológiai információ) szerepét hangsúlyozzák. Elegánsan (és egységes keretben) adnak számot egy adott nyelv jelenségeiről és a nyelvek közötti különbségekről is. Nem csupán az alacsonyabb nyelvi szintek kezelésére képesek, hanem akár szemantikai elemzésre is, illetve szemantikai reprezentáció hozzárendelésére mondatokhoz, vagy akár szövegekhez. Az ilyen típusú elméletek alapjain fejlesztett elemzők pontosabbak egyéb elemzőknél, és lefedettségben is kezdik felvenni a versenyt velük.

A legfontosabb ma is fejlesztett lexikalista elméletek az LFG, a HPSG, az LTAG, a kategoriális és a konstrukciós nyelvtanok. Néhány szóval érdemes a GPSG-t is megemlíteni, amely a HPSG közvetlen elődjének tekinthető. Jelentősége abban áll, hogy megfogalmazta az igényt az explicit módon formalizált elméleti keretek iránt, és reális alternatívája tudott lenni a chomskyánus grammatikának. A GPSG környezetfüggetlen grammatika, amely könnyen implementálható. Fontos felfedezése, hogy a frázisstruktúra-szabályok két komponensre bonthatók: közvetlendominancia és sorrendezési szabályokra (Trón 2001).

[14] Rokon, vagy nagyon hasonló nyelvek közötti fordításkor elég lehet a sekélyelemzés, ha nagyfokú a morfológiai és szintaktikai hasonlóság. Például csehről szlovákra lehet szinte szóról szóra fordítani (Homola – Kuboň 2004).

Nyelvtechnológiai szempontból napjainkban talán a HPSG tekinthető a legsikeresebbnek; egy olyan megszorítás alapú nyelvészeti formalizmus, amely típusos jegy–érték struktúrákat használ. Szintén sikeresek az LFG-n alapuló rendszerek, mert az elmélet egyszerre tud univerzális és nyelvspecifikus lenni azáltal, hogy különböző szinteken különböző nyelvi információkat ábrázol (összetevős szerkezet és függőségi viszonyok). Az LTAG alapjain is készülnek nyelvelemző programok, de nem olyan számban, mint az előző két elmélet esetében. Leginkább lexikonközpontúnak a különféle kategoriális nyelvtanok tekinthetők, ahol a lexikai tételek kombinálódásáért csupán pár egyszerű szabály felelős; viszont alapjaikon (matematikai gyökereik miatt) inkább statisztikai módszereket használó elemzők készülnek. Fontos különbség a korábban említett elméletektől, hogy a kategoriális nyelvtanoknak nem céljuk a redundanciamentesség (különféle szóosztályokra vonatkozó általánosítások kifejezése), nem merül fel a lexikon strukturáltságának (lexikai tételek gazdaságos leírásának) problémája (Trón 2001). Készülnek nyelvtechnológiai alkalmazások a konstrukciós nyelvtan alapjain is, amely egy olyan elmélet, ahol a nyelvi objektumokat konstrukciók engedélyezik (konstrukció: valamiféle formai és jelentéstani információ társítása), és amely elutasítja a mag és periféria szétválasztását, az általános elvek az egyedi esetek rögzítése alapján körvonalazódnak (Trón 2001). (A GASG is így jár el.)

Napjainkban a hatékonyság és a lefedettség növelése érdekében ezek az alkalmazások is használnak kisebb-nagyobb mértékben statisztikai módszereket és esetleg sekélyelemzést is (pl. előszűréshez). Egyre több megközelítés alkalmaz továbbá (optimalitáselméletből átvett) megsérthető megszorításokat, de nem teljesen ugyanazt értik alatta. Az LFG ilyen irányzata az OT-LFG, amely rangsorolja a megszorításokat, hogy több jó elemzés közül ki tudja választani a legjobbat. A HPSG esetében optimalitáselméleti megoldás az ún. default specifikáció, vagyis hogy egy nagyobb osztályra tett megszorítást felülbírálhat egy speciálisabb elemének a követelménye; de ez csak az egyszerűbb megfogalmazást segíti, ha szükséges, kiküszöbölhető. A GASG-ben alkalmazott rangparaméterek hasonlítanak talán leginkább az optimalitáselmélet megközelítésére: egy-egy elemre több megszorítás érvényes, amelyeket meg lehet sérteni mindaddig, amíg a legerősebb rangú kielégül. Így fel lehet tételezni, hogy minden nyelvben ugyanazok a megszorítások érvényesek, csupán a rangsoruk más, így kapunk különböző nyelveken különböző kimeneteket.

3.2.1 LFG

3.2.1.1. Elméletben

Az LFG (Lexikai Funkcionális Grammatika, Kaplan–Bresnan 1982) egy megszorítás alapú nyelvtan, amely nem nyelvészeti értelemben vett szintaxiselmélet, hanem olyan formális keret, amelyen belül több különböző grammatika megfogalmazható (Komlósy 2001). Az univerzális grammatika modellje. Azt vallja, hogy a nyelvi jelek különböző szinteken írhatók le, és a nyelvtan feladata a szintek közötti összefüggések feltárása. Minimálisan két reprezentációs szintet feltételez, az összetevős szerkezet (c-struktúra) szintjét, amely a *variabilitást* biztosítja, és a funkcionális szerkezet (f-struktúra) szintjét, amely az *univerzalitást* teszi lehetővé. Az elmélet legtöbb változata azonban egyéb szinteket is használ. A szinteket leképezések (mappings) kötik össze, és fontos, hogy ezek a leképezések áttetszőek legyenek (*transzparencia*), amit a nyelvtan monotonitása garantál.

Az LFG megalkotásával olyan kompetenciaelmélet kidolgozására tettek kísérletet, amely számítógéppel hatékonyan kezelhető, összhangba hozható a pszicholingvisztikai kutatások eredményeivel, és figyelembe veszi a nyelvhasználatból származó adatokat is. Szintaktikai transzformációk helyett (amelyek létét pszicholingvisztikai kutatások megcáfolták) a lexikonban működő, lexikai tételeket átalakító szabályokat feltételez (az

argumentumokhoz a grammatikai funkciókat csak lexikai szabályok alapján rendeli – lexikai leképezés elmélete). A lexikon így jóval több tételt tartalmaz, de ez megtérül a szintaxisban, ahol így nincsenek transzformációs szabályok (Kálmán et al. 2002). További előnye a chomskyánus irányzatokhoz képest, hogy üres elemeket nem feltételez (null-elem megszorítás), sorrend-független, és betartja a lexikai integritás elvét (komplett szóalakok vesznek részt a folyamatokban, inflexiók és jegyek nem generálhatók, illetve a szintaxis szabályai nem láthatják a szavak belső szerkezetét). Nem konfigurációs, hanem relációs, vagyis a grammatikai viszonyokat nem (feltétlenül) a szórend kódolja, hanem esetleg mindenféle (pl. morfológiai) jegyek, vagyis semleges a lehetséges megjelenítési módok tekintetében (Komlósy 2001).

A szintek közül az a-struktúra (argumentum-szerkezet) tartalmazza a predikátumok argumentumait és azok thematikus szerepét (pl. *üt* <AG, PAT>). Az argumentumok említési sorrendjét a thematikus szerepek feltételezett univerzális hierarchiája határozza meg:

(3) AG > BEN > RECIP/EXP > INST > TH/PAT > LOC[15]

A c-struktúrát (összetevős szerkezetet) környezetfüggetlen frázisstruktúra-fák alkotják, amelyek a felszíni viszonyokat (például a szórendet) rögzítik. A fák csomópontjai annotáltak: funkcionális egyenlőségeket tartalmaznak (attribútum–érték párokra vonatkozó megszorításokat), amelyeket feloldva megkapjuk a mondat f-struktúráját. A környezet-független apparátust tehát egy sajátos indexelési technika egészíti ki (Alberti 2006a):

(4) $S \rightarrow NP[\uparrow SUBJ=\downarrow]\ VP[\uparrow=\downarrow]$

A $\uparrow=\downarrow$ jelölés arra utal, hogy az annotált csomópont ($\downarrow$) meghatározó eleme a fölötte lévő csomópontnak. A (4)-ben olvasható újraírószabály szerint tehát a mondat (S) meghatározó összetevője az igei csoport, annak jegyei (például idő-jegye) öröklődik az egész mondatra. A főnévi csoporthoz tartozó annotáció pedig azt rögzíti, hogy az adott NP a fölöttes csomópont (tehát a teljes mondat) alanyául fog szolgálni. Minden c-struktúrabeli összetevő jelenléte opcionális.

A funkcionális szerkezetet rekurzív attribútum–érték mátrixok alkotják (attribute–value structure, AVS), amelyek az absztrakt szintaktikai (függőségi) viszonyokat tükrözik. Az f-struktúra a nyelvtan változatossága mögött rejlő univerzális jellemzők leírására szolgál. Matematikai objektum, amely a grammatikai viszonyokat, vagyis az a- és a c-struktúra közötti megfeleltetéseket rögzíti (a bennük található információkból unifikációs műveletek hozzák létre). Az egyediségi feltétel miatt függvény (egy attribútumnak csak egy értéke lehet). Az érték lehet egyszerű szimbólum (acc, +), szemantikai forma (PRED értékei), alárendelt f-struktúrák, fonológiai formák (*it*, *there*, idiómák jelentés nélküli, de kötelező részei), vagy ezek kombinációja. A *Láttalak* mondat f-struktúráját (5) mutatja.

[15] A rövidítések feloldása: ágens (cselekvő), beneficiens (haszonélvező), recipiens (kapó), experiens (átélő), instrumentum (eszköz), théma (jellemzett), patiens (elszenvedő) és localis (hely).

(5)
$$\begin{bmatrix} \text{PRED} & \text{'LÁT<(↑SUBJ), (↑OBJ)>'} \\ \text{TENSE} & \text{PAST} \\ \text{SUBJ} & \begin{bmatrix} \text{PRED} & \text{'pro'} \\ \text{NUM} & \text{SG} \\ \text{PER} & 1 \end{bmatrix} \\ \text{OBJ} & \begin{bmatrix} \text{PRED} & \text{'pro'} \\ (\text{NUM} & \text{SG}) \\ \text{PER} & 2 \end{bmatrix} \end{bmatrix}$$

A funkcionális struktúrák jólformáltságát három megszorítás biztosítja: *konzisztencia*, vagyis hogy minden grammatikai funkciónak és minden funkcionális jegynek csak egy értéke lehet; *teljesség*, vagyis hogy a régens vonzatigényeit az f-struktúrának ki kell elégítenie, illetve a szemantikai argumentumhelyekhez társított grammatikai funkciók mindegyikének szemantikai jeggyel (PRED attribútummal) is kell rendelkeznie; végül *koherencia*, amely azt mondja ki, hogy vonzatfunkció csak olyan f-struktúrában szerepelhet attribútumként, amelynek PRED értéke az adott funkciót (is) igénylő régens (Komlósy 2001).

A fentieken kívül két további szintet is feltételez az elmélet, a prozódiai szerkezetet (p-struktúra), amely az adott elem hangtani tulajdonságait kódolja, és a szemantikai szerkezetet (s-struktúra), amely pedig az elemhez tartozó jelentéstani reprezentációt tartalmazza.

Az LFG ismertetését a lexikai leképezés elméletének bemutatásával zárom (Komlósy (2001) alapján), amely a thematikus szerepek feltételezett univerzális hierarchiáján alapul. A legmagasabb rangú argumentum lesz a logikai alany, és a többi szereplő grammatikai viszonya is kiszámolható néhány egyszerű szabály segítségével, továbbá a passziválás is magyarázható vele. A példa az angol nyelvre vonatkozik. Két jegy bevezetését igényli: tematikusan korlátozott (restricted, r) és tárgyjellegű (objective, o). Az alany (subj): [-r -o], a tárgy (obj): [-r +o], a második (tematikus) tárgy: [+r +o] és az oblikvuszi bővítmény (minden más): [+r -o]. A szabályok a következők (mellettük egy példa is látható):

	put <AG, TH, LOC>		
1: ágens → [-o]	1.	[-o]	
2: théma/patient → [-r]	2.		[-r]
3: loc → [-o]	3.		[-o]
4: top-role → [-r]	4.	[-r]	
5: többi → [+r]	5.		[+r]

A *put* ige esetében tehát a szabályrendszer segítségével megkapjuk, hogy az ágens csak alany lehet, a théma lehetne alany is meg tárgy is, de mivel alany már van, csak tárgy lehet, végül a lokatív oblikvuszi bővítmény lesz. A passzivált ige grammatikai szerepeit akkor kapjuk meg, ha a top role-t elnyomjuk, vagyis a 4. szabály helyett a top role → ∅ szabályt alkalmazzuk. Az előbbi példánál maradva ekkor az ágensnek nem lesz hangalakja, és így a théma lesz az alany.

3.2.1.2. Gyakorlatban

Az LFG nem csupán elméletben, hanem gyakorlatban is magas szintű kidolgozottságot és nagy lefedettséget ért el több nyelvre is, melyek különböző nyelvtípusokba tartoznak (konfigurációs, nem konfigurációs); merthogy egyszerre dolgoznak egy nyelvspecifikus (c-) és egy univerzális (f-) struktúrával. Hatékony algoritmusok és implementációk születtek, és

több nagy lefedettségű LFG nyelvtant fejlesztenek folyamatosan, mára már több, mint egy tucat nyelvre. Azonban egyik elméleti alapú nyelvtant sem (akár LFG, HPSG, TAG, CG) alkalmazták még az elmúlt évekig praktikus NLP célokra, amelynek oka, hogy még mindig nem érték el a természetes nyelv teljes lefedettségét, és a rendszerek még nem elég hatékonyak a „piaci" szoftverekhez képest (Frank 2003). Az utóbbi egy-két évben azonban mind sebességben mind robusztusságban nagy előrelépés történt azáltal, hogy különböző sekélyelemző technikákat és statisztikai módszereket vontak be, és ezek a hibrid rendszerek, úgy tűnik, a nagy pontosság eléréséhez szükséges komplexebb elemzés ellenére is fel tudják venni a versenyt hatékonyságban a meglévő programokkal.

Kaplan–King (2003) azt a hipotézist vizsgálja, hogy a különféle sekély jelölő (mark-up) technikák (szófaji egyértelműsítés, fráziscsonkolás) csökkentik a többértelműséget és növelik a sebességet, miközben a pontosságot nem befolyásolják. Általában az „igazi" (mély) elemzéstől függetlenül működnek: bemenetként megkapják a normál ortográfiájú bemenő karaktersorozatot, és úgy adják vissza, hogy ellátják különféle diakritikus jegyekkel. A cikk három ilyen technikát tárgyal: szófaji elemzés, tulajdonnév-felismerés és címkézett zárójelezés. (Architektúra: *eredeti string* → *3 féle jelölő* → input string → tokenizáló → *tulajdonnév-konvertáló* → morfológia → *szófaji szűrő* → LFG nyelvtan, belőle két ág: c- és f-struktúra, közben a *címkézett zárójelezés szűrője, és a tulajdonnevekhez tartozó szublexikális szabályok*.) Eredményei (azon túl, hogy további vizsgálatok szükségesek) azt mutatták, hogy a tulajdonnév-felismerő sebességben és pontosságban is javított az eredeti elemzőn, a címkézett zárójelezés sebességben javított, pontosságban nem rontott sokat, a szófaji szűrő viszont sokat rontott a lefedettségen és a pontosságon is.

Schneider (2005) is azt vizsgálja, hogyan lehet nagy lefedettséget elérni mély-nyelvészeti elemzőkkel. Ezt két tényező teszi különösen nehézzé: a távoli függőségek, mert nem tartható a c-struktúrában még meglévő környezetfüggetlenség, és a többértelműség, mert óriási lesz a keresési tér. Ebben a cikkben alapvetően az első problémával foglalkozik:

Az eredeti LFG-ben (Kaplan–Bresnan 1982) a távoli függőségekre üres c-struktúrabeli elemeket (nyomokat) használtak. Később rájöttek, hogy nem összetevős, hanem inkább funkcionális szinten kellene őket kifejezni, ez az ún. funkcionális bizonytalanság[16]: távoli függőség az f-struktúrában kifejezve. Így nincs szükség nyomokra, a megközelítés adekvátabb és kevésbé redundáns.

Az LFG leginkább azért használ c-struktúrát, mert az környezetfüggetlen. Az f-struktúra magában csak egy függőségi nyelvtan, amely környezetfüggő, így nem hatékony. A Függőségi Nyelvtan (DG) az eredeti elképzelés szerint egy proto-szemantikai, egyszintű, nyelvfüggetlen elmélet, amelynek nincs szüksége c-struktúrára. A szórendnek nincs elsődleges szerepe, időnként segít az egyértelműsítésben. Nincs arra vonatkozó megszorítás, hogy a fej hol keresse a bővítményeket, azok lehetnek akármilyen távolságra. Később (alapvetően a chomskyánus irányzat hatására) többszintű Függőségi Nyelvtanok is születtek, illetve (részben) ezen alapuló egyéb nyelvtanok, mint az LFG vagy a HPSG. (Az LFG a válasz arra a kérdésre, hogy az összetevős szerkezet vagy a függőségi viszonyok számítanak,

---

[16] Komlósy (2001) alapján: „funkcionális bizonytalanság [functional uncertainty]: a kontrollálónál nem lehet egyértelműen megadni a grammatikai relációknak azt a láncolatát, amely a kontrollálthoz vezet, mert az elvileg a legkülönbözőbb mélységekben is be lehet ágyazva." A probléma az extrapozíciós szerkezetek kezelése kapcsán merül fel (angol: wh-típusú, „távolsági" viszonyok esete, ideértve a topikalizációt is). Megoldás: nem csupán egy grammatikai funkció, hanem egy azokból álló reguláris kifejezés is állhat a leírásban. Így például a topikalizációra használt formula: (↑TOPIC) = (↑COMP* GF), ahol a * a Kleene-művelet jele, GF pedig a primitív grammatikai funkciók halmaza. A kifejezés jelentése: a mondat topikja azonos a mondat *akármilyen mélységben beágyazott* mondatvonzatának valamelyik vonzatával (a mélység lehet 0 is, ekkor a mondat topikja a mondat valamelyik vonzatával azonos).

azáltal, hogy mindkettőt tartalmazó leírást használ: egyrészt összetevős szerkezeten alapuló környezetfüggetlen c-struktúrát, másrészt pedig nem-konfigurációs f-struktúrát, amely az összetevők közötti függőségi viszonyokat fejezi ki.)

Mivel gyakorlatilag nincs a szórendre megszorítás, ezért Függőségi Nyelvtannal nagyon nehéz hatékony elemzést végezni. Olyan algoritmus, amely megszorítatlan távoli függőségeket tud elemezni, NP-teljes[17]. Hogy a probléma kezelhető legyen, a környezetfüggőséget a minimumra kell szorítani. Egy teljesen környezetfüggetlen nyelvtan esetében az algoritmus komplexitása lehet akár $O(n^3)$ is (vagyis polinomiális), ezért nagy a c-struktúra vonzereje a nyelvtechnológiában. Az LFG úgy szorítja vissza a környezetfüggőséget, hogy a környezetfüggetlen c-struktúrát képezi le a nemkonfigurációs f-struktúrára. Bizonyított, hogy a funkcionális bizonytalanság az egyetlen környezetfüggő kiterjesztés, amely szükséges a természetes nyelv leírásához (LTAG-ben ilyen a csatolás (adjoining)).

A különféle formális nyelvtanok komplexitása a gyakorlatban magasabb, mint a korábban említett $O(n^3)$. Nem lehet pontosan tudni, a TAG-é $O(n^7)$ vagy $O(n^8)$, implementációtól függően, a HPSG-é a legrosszabb esetben exponenciális. Eddig az ilyen mélynyelvészeti elméletekre épülő elemzők komputációsan túl drágának bizonyultak, nem elég hatékonyak. Ezért koncentráltak sokan inkább véges állapotú eszközökkel megvalósítható feladatokra, mint a csonkokon alapuló elemzés vagy a sekélyelemzés.

A környezetfüggetlen elemzés tehát nagyon hatékonynak bizonyult a nagy lefedettségű elemzésben, különösen akkor, amikor megsegítették statisztikai alapú egyértelműsítéssel. Ezek az elemzők azonban általában környezetfüggetlen kimenetet adnak, vagyis nem kezelnek távoli függőségeket. A treebankok esetleg tartalmaznak rájuk vonatkozó információkat (nyomokat), de a rajtuk tanított elemzők ezeket általában nem veszik figyelembe. Schneider (2005) megmutatja, hogy a legtöbb távoli függőség kifejezhető környezetfüggetlen módon, amelyik nem, az pedig funkcionális bizonytalansággal. A treebankokban lévő nyomok legnagyobb része átalakítható lokális függőséggé véges állapotú mintázatokat vagy lexikalizált utóelemző szabályokat használva. Némelyik távoli függőség a konfigurációs grammatikai reprezentációból adódik, a többi pedig modellezhető az enyhén környezetfüggő LFG-beli funkcionális bizonytalanság segítségével.

Mára – hála a statisztikai eszközök bevonásának – az első mélynyelvészeti formális nyelvtanon alapuló elemzők elérték azt a szintet lefedettségben és robusztusságban, hogy nagy korpuszokat lehet velük elemezni. Ennek egyik oka, hogy a távoli függőségeket jól közelítik determinisztikus vagy környezetfüggetlen eszközökkel (pl. a Frank 2003 által bemutatott rendszer).

Az angol nyelvben kétféle távoli függőség található. Az egyik, amikor egy vonzatot elmozgatunk, az eredeti helyén csak a nyoma marad, és nemvonzat pozícióba kerül. Ilyen a wh-mozgatás, a topikalizáció vagy a vonatkozó mellékmondatok esete. A másik, amikor egy összetevő, ami a felszínen csak egyszer jelenik meg, a funkcionális szerkezetben (mint szemantikai argumentum) többször. Ilyen a kontroll, az emelés és az it-cleft. Az első típus legnagyobb része (kivéve a nem-alanyi wh-mozgatás) kezelhető lokálisan a DG-ben. A második típusnál a környezetfüggetlen elemzés elegendő, a koreferencia megállapítását az utóelemző végzi statisztikai eszközökkel (kontroll és emelés: ha nincs alany és az ige infinitívuszban van, akkor alanyi vagy tárgyi kontrollal állunk szemben, a fölérendelt ige illetve melléknév lexikai valószínűségétől függően).

Schneider (2005) javaslata azon alapul, hogy ha az f-struktúrát környezetfüggetlenül lehetne előállítani, nem lenne szükség a teljes c-struktúrára, elegendő lenne a fontosabb

[17] NP: nondeterminisztikus polinomiális. Az NP-teljes problémákat valószínűleg nem lehet polinomiális időben megoldani (ez a sejtés, de még nem bizonyított), vagyis egyáltalán nem tekinthetők számítástechnikai szempontból hatékonynak.

elemeinek (csonkjainak) megtalálása, és statisztikai módszerek alkalmazása. Az általa javasolt elemző (amelynek kimenete egyszerű f-struktúra) reprezentációsan minimális, mélynyelvészeti, lexikalista, robusztus és gyors; kiértékelték (felveszi a versenyt a többi elemzővel) és ki is próbálták. A környezetfüggő konstrukciókat úgy kezeli, hogy (1) a legtöbb távoli függőséget véges állapotú mintázatokkal közelíti (1. típus), (2) utóelemzést végez (2. típus), (3) az LFG-ből ismert funkcionális bizonytalanságot alkalmazza. Az elemző tehát kombinálja a sekély és mélyelemzést (hogy pontos is és hatékony is legyen), és a távoli függőségeket legtöbbször környezetfüggetlen módon fejezi ki. Így a komplexitás olyan alacsony, mint egy valószínűségi elemzőnél, viszont mélyelemzést végez, tehát a kimeneten igazi nyelvészeti információ olvasható.

Végkövetkeztetése az, hogy nem feltétlenül vagy szükség a c-struktúrára vagy egyéb konfigurációs felszíni reprezentációs szintre a nyelv szintaktikai elemzéséhez, csonkok és függőségi viszonyok elegendőek. A GASG végsőkig viszi ezt a javaslatot, kipróbálja, mi történik akkor, ha semmivé redukáljuk a c-struktúrát. A hipotézisünk az, hogy a program nem lesz emiatt számítástechnikai szempontból olyan, mint a függőségi nyelvtanok (nem hatékony), mivel a GASG-ben vannak megszorítások a szórendre is, rangparaméterek formájában.

Az utóbbi években tehát nagy fejlődés tapasztalható a nyelvészeti alapú, kézzel írt nyelvelemzők készítésében, egyre nagyobb lefedettséget tudnak velük elérni (a pontosság feladása nélkül). A kezdeti nagyobb energiabefektetés megtérül később, amikor új nyelvekre dolgozzák ki a rendszert. Úgy tűnik tehát, a mélyelemzés tényleg a megfelelő megoldás, ha magasabb célokra írunk nyelvelemző programokat. A többértelműség kezelésére ezek a rendszerek is leginkább korpuszt használnak, mert a legtöbb esetben nyelvtanon kívüli információra van szükség, mint például a kontextus. A kis szabályok és ritka alakzatok problémájára egyes alkalmazások (pl. Frost et al. 2005) optimalitáselméleti megközelítést alkalmaznak. Kiszámolják az összes lehetőséget, ezeket rangsorolják, végül a rangok segítségével megkapják, mely alakzat(ok) az optimális(ak), de a többi lehetséges elemzés sem veszik el. A rangok hozzárendelését korpusz alapján végzik, mert megbízhatóbb, gyorsabb és hatékonyabb.

Készülnek olyan programok is, amelyek speciális célokat tűznek maguk elé, egy ilyen például a GETARUN rendszer (Cahill et al. 2002), amelyet szövegértésre fejlesztettek ki. Három fő modult tartalmaz: egy alacsony szintűt, amely csupán szintaktikai elemzést végez, egy közepes szintűt, amely szemantikai interpretációt rendel a szöveghez és diskurzus-modellt épít (szituációs szemantikát használ), végül egy magas szintűt, amely érvelésre és generálásra képes. A legfontosabb feladatok között található az egyértelműsítés, a vonzatok és a szabad bővítmények elkülönítése, és a visszalépések (backtracking) visszaszorítása. LFG-n alapul, felülről lefelé elemez, és az Univerzális Grammatika szellemében íródott, vagyis feltételez egy közös magot, amely minden nyelvre érvényes, a különbség pedig parametrizálható szabályokban ragadható meg. Prolog programnyelven írták. Lexikonában ragozott szóalakok találhatók összes tulajdonságukkal együtt (AVS-ekben). Ha egy szintaktikai összetevő sikeresen felépül, a program ellenőrzi, hogy az f-struktúrára vonatkozó elvek (konzisztencia, egyediségi feltétel, koherencia) teljesülnek-e. A program kimenete egy f-struktúra, ami bemenetül szolgál a kötéseket (binding) vizsgáló és betöltő modul, illetve a szemantikai komponens számára, ahol a logikai viszonyok számolódnak ki. A rendszer extrapozíciót is tartalmaz, hogy a távoli függőségeket is kezelni tudja. Topik és fókusz felismerésére és elemzésére is képes.

Nem csupán angol, hanem más nyelvekre is fejlesztenek LFG-alapú elemzőket. Az XLFG például (Clément, Kinyon 2001) egy nyílt forráskódú elemző francia nyelvre. Operációs rendszerek között hordozható, felhasználóbarát felülettel rendelkezik pedagógiai alkalmazásokra is, továbbá elég rugalmas ahhoz, hogy új elveket is tesztelni lehessen benne.

C programnyelven írták. Kimenetként c- és f-struktúrát ad. A cél nem az volt, hogy robusztus és gyors elemzőt fejlesszenek, ennek ellenére viszonylag gyorsnak mondható (egy hosszúnak tekinthető mondatot, amelyhez 73 elemzést ad, 2 mp alatt elemez). Mint pedagógiai alkalmazás a célja tanulni és tanítani az LFG-t. Agrammatikus mondatokat is elemez, ha az összetevős szerkezet jól formált, de az f-struktúra nem. Megmutatja, miért nem grammatikus, melyik elv sérült a háromból, vagy hogy hol nem működött az unifikáció. A ragozott alakokat morfológiai táblázatokból és a lemmák lexikai tulajdonságaiból (szubkategorizáció, lexikai transzformációs szabályok) számolja ki. Súlyozza a lexikai egységeket, hogy melyik fordul elő nagyobb valószínűséggel a mondatban. Egyértelműsítésre nem OT-LFG-t használ, mint a legtöbb alkalmazás, hanem TAG derivációs fákat. Emellett gyakorlati és elméleti érveket is sorakoztat. Gyakorlati érv, hogy a valószínűségen alapuló módszerek nem nyelv- és nem domainfüggetlenek, továbbá drágák, mert óriási tanítókorpusz szükséges hozzájuk (pl. treebank), ami leginkább csak az angol nyelvre létezik. Elméleti érv, hogy a valószínűségen alapuló módszereknek nincs magyarázó erejük, legtöbbször működnek ugyan, de nem mondanak semmit arról, hogy miért. Továbbá sok többértelműség nem igazán ragadható meg a c-struktúrában, inkább valamiféle függőségi struktúrában (lexikai szinten), például hogy egy lexikai egység inkább idiomatikus, mint szó szerinti jelentésben szerepel, inkább vonzat, mint szabad bővítmény, illetve hogy egy vonzat a hozzá közelebbi lehetséges régenséhez tartozik inkább.

Készítenek treebankokat is LFG-alapokon, például Rosén et al. (2005). A treebank mondatszerkezettel annotált korpusz (nem csak szófajcímkékkel), amely egyéb szintaktikai vagy akár szemantikai információt is tartalmazhat. Azért van rájuk szükség, mert ha a korpuszt csak szószintű annotálással látjuk el, az nem mindig elégséges, nem mindig találja meg például a frázisok határait vagy a különírt szóösszetételeket. Sokszor hibás is az annotálás, mert leginkább automatikus módszerekkel készül. A megoldás tehát egy alaposabban annotált korpusz (treebank), amelyet fél-automatikus módszerrel hoznak létre (kézzel lassú és nem konzisztens, teljesen automatikusan nem elég pontos), vagyis a kimenetet átnézi egy ember. Az egyértelműsítés is kézzel történik, mert egy gép számára több kifejezés tűnik többértelműnek, mint egy ember számára. Sok treebank használ LFG formalizmust, amelyek úgy készülnek, hogy LFG-elemzővel elemeznek egy korpuszt, az elemzéseket átnézik, kiválogatják a helyeseket (kézzel), esetleg módosítják, ha szükséges, majd eltárolják. Az így kapott treebank kutatási célokra is alkalmas, hiszen nagy mennyiségű empirikus adatot tárol a nyelvről, így könnyedén lehet benne keresni, milyen alakzatok léteznek, azoknak milyen a gyakorisága stb. Gyakorlati alkalmazások és elméleti feltevések is tesztelhetők velük, léteznek továbbá párhuzamos treebankok is, amelyek több nyelven tartalmazzák ugyanannak a mondatnak az elemzését.

Rosén et al. (2005) a TREPIL norvég treebank fejlesztéséhez a ParGram Projekt (lásd lejjebb) eredményeit vette alapul. Háromféle kimenetet ad: c-, f és mrs-struktúrát (MRS: Minimal Recursion Semantics, mélyebb szintű szemantikai reprezentációt nyújt, mint az egyszerű predikátum–argumentum viszonyok, a 3.2.2.2 pontban lesz róla szó részletesebben). A TREPIL treebank és nyelvtan egyben, továbbá gépi fordításra is alkalmas, különösen, amióta szemantikai reprezentációt is rendel a mondatokhoz. Weben megjeleníthető, mindhárom kimenetet és az összes megoldást kiírja (hacsak nem kéri a felhasználó, hogy csak egy megoldást adjon). Lisp fejlesztőkörnyezetben készült, XML-kompatibilis, és Java nyelvű elemeket is használ. Optimalitásjelölő rangokat is tartalmaz, de ez a komponens kikapcsolható.

Az LFG-t mint elméleti keretet használó elemzők közül a legígéretesebbek azok, amelyeket a Parallel Grammar projekt (ParGram, Butt et al. 2002) keretében fejlesztenek. A projekt lényege, hogy a hasonló szerkezetek elemzései a különféle nyelvekben, amennyire csak lehet, párhuzamosak. Ezzel az erős standardizációval elérik, hogy a nyelvtanok hasonló

számítógépes alkalmazásokban használhatók, és a gépi fordítás is egyszerűsíthető. Eredeti célja az LFG-formalizmust tesztelni: univerzalitását, lefedettségének határait, és hogy mennyire tartható a párhuzamosság a különféle nyelvek között. Az eredmények biztatóak, azt bizonyítják, hogy nagyfokú párhuzamosság érhető el, így új nyelvtanok építése is könnyebben, gyorsabban lehetséges. Az elemzőt szándékosan nagyon különböző nyelvekre dolgozták ki először: angolra, németre, japánra, urdura és norvégra, így megmutatható, hogy a rendszer tudja kezelni a szabad szórendet is vagy a névmáselhagyást. Némelyik nyelvtan nagy lefedettségű, ipari alkalmazás (angol, német, japán), és némelyiknél az s-struktúrát (szemantikát) is kidolgozták (pl. norvég).

A párhuzamosságot az f-struktúra szintjén érik el. Néha szándékosan nem párhuzamosak az f-struktúrák, például a névszói állítmány esetében (az *It is red* mondat esetében az angolban a kopula a fej, a japánban a melléknév), vagy amikor egy nyelvben megtalálható egy bizonyos jegy (pl. a németben a nem), egy másikban pedig nem. Ez utóbbira példa az angol progresszív jegy is: *He cried* vs. *He was crying.* A TNS-ASP jegy felelős az időért és az aspektusért, mindkét mondat esetben TENSE: past, de míg az első mondatnál PROG: -, a másodiknál PROG: +. Álláspontjuk szerint azokban a nyelvekben, ahol a progresszivitás nem jelenik meg a felszínen, ténylegesen nincs is ilyen megkülönböztetés (például a németben), ilyenkor az f-struktúrából a PROG jegy hiányzik. Az az érv emellett, hogy ha lenne ilyen megkülönböztetés ezekben a nyelvekben is, akkor rengeteg többértelműséget kapnánk. Ehelyett azt vallják, hogy majd a szemantika eldönti, hogy az adott mondat progresszív-e. Folyamatosan dolgoznak azon, hogy a párhuzamot minél jobban megtartsák, a formalizmust úgy módosítják, hogy ezt a célt elérjék. Mert minél kisebb a különbség a különböző nyelvek elemzései között, annál könnyebb soknyelvű alkalmazásokat készíteni (pl. jó minőségű gépi fordítást). Jelenleg a projekt keretében az alábbi nyelvekre fejlesztenek nyelvtanokat[18]: kínai, angol, francia, német, grúz, norvég, japán, török, urdu, walesi, arab, magyar[19], vietnámi, madagaszkári.

Az elméletet az XLE (Xerox Linguistic Environment) elemző és nyelvtan-fejlesztő környezetben implementálták. A rendszer lelke egy hatékony unifikációs alapú elemző és generáló. Olyan jelenségeket tartalmazó mondatok elemzésére is képes, mint például a funkcionális bizonytalanság vagy a mellérendelés. Újabb verziói a többértelműséget is hatékonyan kezelik (Cahill et al. 2007), mégpedig úgy, hogy a mondat összes lehetséges elemzését egy „csomagolt” (packed) reprezentációban tárolja a program mindaddig, amíg nem tud közülük dönteni (valószínűségi alapon). Kimenetként c-struktúrát (morfológia, összetevős szerkezet, szórend) és f-struktúrát (predikátum–argumentum struktúra, módosítók, idő, mód stb.) is ad. Olyan technikákat is tartalmaz, amelyek segítségével növelhető a robusztusság (egészében nem grammatikus mondatok esetében képes azok kisebb részeihez reprezentációkat társítani), és be van építve egy olyan szűrő, amely segítségével a program bármely mondat elemzését polinomiális időn belül elvégzi.

Gépi fordítás céljára az XTE (Xerox Translation Environment) alkalmas, amely a ParGram projekt fordítórendszere (Frank 1999). A nyelvfüggetlennek tartott f-struktúra teszi lehetővé, hogy többnyelvű nyelvtechnológiai alkalmazásokra is használhassák a rendszert, mint amilyen a gépi fordítás. A hatékony elemzőt és generálót tartalmazó XLE platformot kiterjesztették egy transzfer-komponenssel. A transzfer az f-struktúrán keresztül történik, amelyről ugyan elismerik, hogy időnként nem kellő mértékben univerzális (pl. a fejváltó fordítás esetében), de a nyelvek közötti különbségek legnagyobb része már ezen a szinten eltűnik. A legmegfelelőbb az s-struktúrán (szemantikán) keresztüli fordítás lenne, viszont

[18] Forrás a ParGram honlapja: http://www2.parc.com/isl/groups/nltt/pargram/

[19] Az előkészületek megtörténtek már a magyar ParGram projektet illetően, de a tényleges fejlesztések 2008 nyarától indulnak.

ekkor ki kellene dolgozni a szintaxis–szemantika interfészt mind az elemzés, mind a generálás oldaláról (azóta ez irányban folytak a kutatások). A többértelműséget a döntés végsőkig való halasztásával kezeli.

Összefoglalva tehát az LFG mind elméletben mind a nyelvtechnológia területén jelentős eredményekkel rendelkezik. Számos nyelvre fejlesztenek nyelvtanokat, amelyek a szigorú LFG formalizmust követik, így az elemzések kellő mértékben párhuzamosak ahhoz, hogy akár egyetlen hatékony fejlesztőkörnyezet kezelni tudja őket. Ezáltal a jó minőségű gépi fordítás is elérhetővé válhat, de legalábbis lényegesen egyszerűbben megvalósítható. Az utóbbi években fellendültek az olyan kutatások is, amelyek azt vizsgálják, hogyan lehetne az amúgy mélynyelvészeti elemzést végző rendszerekbe statisztikai módszereket integrálni, így ezek a nyelvtanok is elérhetik a piacon jelenleg megtalálható rendszerek lefedettségét és hatékonyságát úgy, hogy pontosságban jelentősen túlszárnyalják őket.

3.2.2 HPSG

3.2.2.1. Elméletben

Az elméletet alapvetően Trón (2001) és Kálmán et. al (2002) alapján ismertetem. A Fejközpontú frázisstruktúra-nyelvtan (Head-Driven Phrase Structure Grammar, HPSG, Pollard–Sag 1994) is egy *deklaratív*, megszorítás alapú nyelvtan. Az explicit megfogalmazás (HPSG-formalizmus) miatt könnyen algoritmizálható, implementálható (emiatt akár számítógépes nyelvészeti irányzatnak is tekinthető). A nevéből következik, hogy szintén használ frázisstruktúrát (PSG), a *fejközpontúság* (H) pedig azt jelenti, hogy a függőségi viszonyok a fejbe vannak beépítve. Elutasítja a mag és periféria szétválasztását, minden jelenség feltárása a célja.

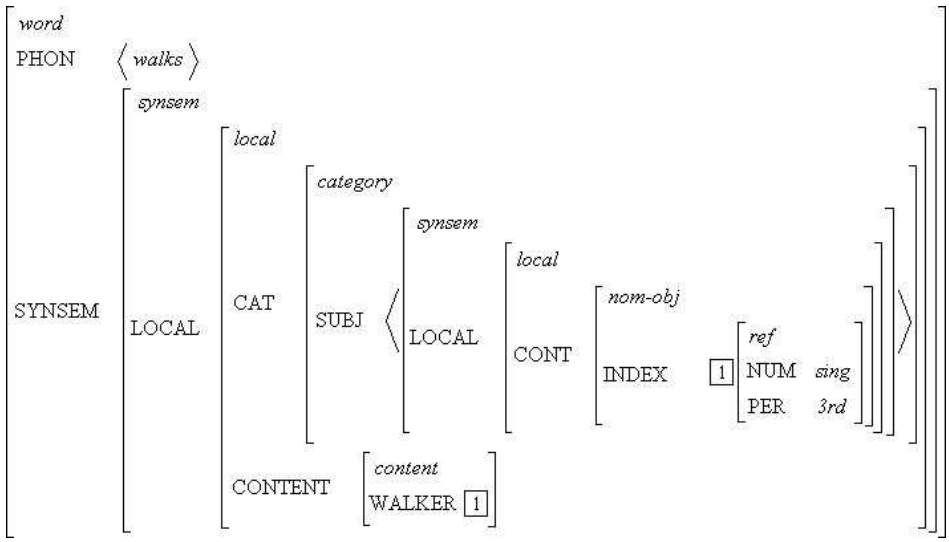
A HPSG nyelvelméletének objektumai a *lexikon*, az univerzális *elvek*, illetve a nyelvspecifikus elvek és konstrukciók. *Egyszintű*, vagyis nem tekinti a hagyományos nyelvi modulokat különálló komponenseknek, és nem tételez fel különálló reprezentációs formalizmusokat sem az egyes nyelvi szintek – fonológia, morfológia, szintaxis, szemantika – leírására. *Generatív* nyelvtan, vagyis célja egy olyan explicit formális rendszer megadása, amely alkalmas egy adott nyelv jól formált és teljes kifejezéseinek előállítására. Ezeket azonban nem az alapegységekből származtatja különféle műveletekkel, mint a levezetés alapú (derivációs) elméletek, hanem leírások (jellemzések) halmazát tárolja, a nyelv pedig azon elemek összessége, amelyek ezeknek a megszorításoknak megfelelnek. Nem szól tehát a jelek előállításának a folyamatáról, csak deklarálja, melyek jól formáltak, ezért nevezik deklaratív nyelvtannak. Megszorítás alapú, mert kiszűri a sok lehetséges nyelvi megnyilvánulásból a jól formáltakat, azokat, amelyek minden megszorítást kielégítenek. A nyelvi kompetencia modellje.

Akárcsak az LFG, a HPSG is attribútum–érték struktúrákat (AVS) tárol, amelyek különböző *típusokba* rendezhetők (az objektumok fajtákba sorolhatók, és a fajták feletti általánosítások a típusok). Egy nyelvelméletnek definiálni kell, melyek azok a típusok (nyelvileg releváns entitások), amelyek számítanak, ezeknek milyen a belső struktúrájuk, milyen részstruktúráik vannak, illetve hogy az egyes típusok hogyan ágyazódnak egymásba. Meg kell adni továbbá az adott típusra vonatkozó megszorításokat. A HPSG-ben egy típus lehet alulspecifikált (pl. *sheep* száma nem *sg* vagy *pl*, hanem csak annyi, hogy *number*, egy olyan típus, amelynek a fenti két érték az altípusa). A típusok tulajdonságai öröklődnek az altípusokra, de az altípusoknak lehetnek saját tulajdonságaik is.

Az AVS-ben kerülnek ábrázolásra a különféle *jelek* (sign), ahol egy jel lehet szó (word) vagy frázis (phrase). Egy AVS két értéket tartalmaz: PHON (a jel hangtani

reprezentációja) és SYNSEM (szintaktikai és szemantikai tulajdonságai). A SYNSEM jegyen belül találhatók a LOC és a NON-LOC jegyek, előbbi a lokális, utóbbi a távoli függőségekre vonatkozik. A LOC attribútumon belül rögzítésre kerül annak kategóriája (CAT), tartalma (CONTENT) és a kontextus, amelyben előfordul (CONTEXT). Végül a CAT jegyen belül rögzítjük, hogy mi a fej (HEAD), a SUBCAT jegy pedig a szubkategorizációt kódolja, vagyis hogy az egység milyen elemeket kíván meg a mondatba, amelyet oblikvuszi hierarchiába rendezve jelenítünk meg. Az angol *walks* szó ábrázolása (6)-ban olvasható[20] (a magyar *sétál* szónak ugyanez lenne az ábrázolása a fonetikai formát kivéve). Az attribútumokat nagybetűk jelölik, a zárójelek tetején található dőlt betűs kifejezések pedig a típusok.

(6) Az angol *walks* szó lexikai ábrázolása



A frázisokhoz tartozó AVS-ben a közvetlen összetevők típusai: HEAD-DTR (fej), COMP-DTR (vonzat) és ADJ-DTR (szabad bővítmény). „A közvetlen összetevős konstrukciók deklarációja, valamint a hozzájuk tartozó attribútumok és azok lehetséges értékeinek definiálása elégséges ahhoz, hogy az AVS-ek hagyományos szintaktikai fastruktúráknak megfelelő dominanciaviszonyokat tudjanak kódolni. A frázisokat így nem szükséges közvetlenül frázisstruktúra-építő szabályokkal felépíteni." (Kálmán et al. 2002: 171)

A HPSG négy univerzális frázisstruktúra-elvet fogalmaz meg: (1) Fejjegyelv: a frázis bizonyos tulajdonságai (pl. szófaj, eset, igemód) megegyeznek a fej ezen tulajdonságaival (öröklődés). (2) Szubkategorizációs elv: a vonzatok tulajdonságai megegyeznek a fej rájuk vonatkozó kívánalmaival, és a fej vonzatigényéből törölni kell, amit már kielégített valami összetevő. Itt ellenőrződik, hogy minden vonzat megtalálható-e, illetve hogy csak olyanok szerepelnek-e, amiket a fej megkíván. (3) Közvetlen összetevős elv: a fej és vonzatai bizonyos sémákra illeszkednek, a közvetlen összetevős sémákat (lehetséges fákat) meg kell adni. Ezek nyelvspecifikusak, és többé-kevésbé azt rögzítik, hogy milyen sorrendben kapcsolódhatnak a bővítmények a fejhez Az angol nyelvre olyan sémákat feltételeznek, mint a fej–alany séma, a fej–vonzat séma vagy a fej–szabad bővítmény séma. (4) Szemantikaelv: a szemantikai kompozíciót irányítja. Megmondja, hogy a frázis jelentése (a CONTENT attribútum értéke) azonos a fej jelentésével (a CONTENT attribútum értékével).

A HPSG is a lexikonban kezeli a kontrollt, az emelést és a passzivizációt is. A távoli függőségek kezelésére is bevezet egy olyan eszközt, amellyel el tudja kerülni a nyomok feltételezését.

[20] Forrás: http://www.emsah.uq.edu.au/linguistics/Working%20Papers/ananda_ling/HPSG_Summary.htm

Az egyszintű nyelvtanok szellemében a jelentéstani komponenst nem tekinti különálló modulnak: a jelentéstani ábrázolás a megszorítások együttes kielégítésével a mondattani, morfológiai struktúrával együtt áll elő. A jelentéstani komponens felé a legfontosabb elvárás, hogy kompozicionális legyen. A HPSG a jelentést leginkább MRS segítségével ábrázolja, amely egy lapos (flat) szemantikai reprezentáció, amely alulspecifikált ábrázolásra is képes, így könnyedén kezeli a többértelműséget elméletben és implementációsan egyaránt. A következő pontban röviden bemutatom ezt az elméletet.

3.2.2.2. MRS

Az elméletet Copestake et al. (2005) alapján ismertetem. Az MRS (Minimal Recursion Semantics) megalkotói olyan szemantikai reprezentációt akartak létrehozni, amely nyelvészeti alapú számítógépes alkalmazásokban használható elemzésre és generálásra egyaránt. Egy szemantikai elméletnek négy kritérium teljesítését kell megcéloznia: (1) leíró adekvátság, hogy jól fejezze ki a kívánt jelentést, (2) grammatikai kompatibilitás, hogy egyéb grammatikai információhoz – leginkább szintaxishoz – jól köthető legyen, (3) számítógépes megvalósíthatóság, hogy létre tudja hozni a jelentésreprezentációkat, hatékonyan megtalálja a jelentéstanilag ekvivalens kifejezéseket, és egyszerűen ki tudja fejezni a relációkat a különböző szemantikai reprezentációk között, végül (4) alulspecifikáltság, hogy engedjen meg alulspecifikált formulákat, amelyekből aztán rugalmasan, monoton építkezéssel lehessen előállítani a már konkrétabb jelentéseket. Korábban nemigen született olyan elmélet, amely mindegyik kritériumnak megfelelt volna, ezért például az elsőt gyakran feláldozzák a harmadikért. Egy másik probléma, hogy a szemantikai reprezentációhoz gyakran olyan információkra van szükség, amelyek egyetlen mondat elemzésével nem nyerhetők ki, több mondatos szöveg elemzésére lenne tehát szükség. A negyedik kritérium melletti érv pedig az, hogy gyakran nem kell feloldani egy (például hatóköri) többértelműséget, mert nincs rá szükség például a fordításhoz, ezért szerencsés, ha a reprezentáció lehet (és maradhat) alulspecifikált.

Mindezekre megoldás az MRS, amely önmagában még nem egy szemantikai elmélet, inkább valamiféle metanyelv szemantikai struktúrák leírására a predikátumlogika nyelvén, kiegészítve általánosított kvantorokkal. Alapelemei: elementary predications (EPS), amelyek egyszerű relációk argumentumok között (pl. beyond(x,y)), ahol alapesetben a reláció neve egy lexémának felel meg. Az MRS „lapos” (flat) reprezentáció, mert az EP-k soha nincsenek egymásba beágyazva. Azonban – szemben a korábbi lapos reprezentációkkal – mégis képes a hatóköri viszonyokat kifejezni. A szintaktikája úgy lett tervezve, hogy könnyen hozzárendelhető legyen jegyalapú grammatikákhoz, mint például a HPSG. Az MRS nem azért született, hogy elméletileg újat nyújtson a különféle szemantikai elméletek között, hanem azért, hogy legyen egy olyan eszköz, amellyel hatékonyan kezelhető a számítógépes szemantika. Magába foglal több olyan módszert, amelyek bizonyítottan alkalmasak arra, hogy nagy lefedettségű, általános célú nyelvtanok elemző, illetve generáló feladatát elvégezzék.

Hogy mik a problémák a hagyományos szemantikákkal számítógépes szempontból, azt a szemantikai transzfer alapú gépi fordításon mutatja meg (forrásnyelvi mondat elemzésével forrásnyelvi szemantikai reprezentáció előállítása, majd abból transzferszabályok segítségével célnyelvi szemantikai reprezentáció készül, amelyből generálással előáll a célnyelvi mondat). Az egyik probléma, hogy a hagyományos szemantikai reprezentáció tartalmaz információt a szintaxisról (frázisstruktúra), pedig az irreleváns szemantikailag, ezért hamis többértelműséget kaphatunk, illetve olyan olvasatok is előállnak, amelyek nem léteznek, és amelyekre ezért akár nem is alkalmazhatóak a transzferszabályok. A lehető legegyszerűbb transzferszabályokra lenne szükség és arra, hogy a program a logikai

formából generálni tudja a célnyelvi mondatot. Ez utóbbit hagyományos szemantikai reprezentációval nem tudták hatékonyan megvalósítani, ezért kezdtek lapos reprezentációban gondolkodni, ami úgy tűnik, megoldja a problémát. Például a *fierce black cat* kifejezés logikai formája hagyományos szemantikai reprezentációval (7) alatt olvasható, míg ugyanez egy lapos elméletben (8)-at eredményezi.

(7) λx[fierce(x)$\wedge$(black(x)$\wedge$cat(x))]
(8) fierce(x), black(x), cat(x)

Lapos szemantikai reprezentációval tehát egyszerűbb transzferszabályok írhatók, és nem jelentenek problémát a nyelvek közötti sorrendbeli különbségek sem (a spanyolban például a főnév van elöl). A hatóköri viszonyokra vonatkozó információkra azonban szükség lehet. Ha (9)-et a hagyományos (10) szerinti reprezentáció helyett a (11)-ben olvasható formulával írjuk le, elveszik az a tudás, hogy melyik predikátum vesz fel nagyobb hatókört, így a fordítás ugyanúgy lehetne egy (12)-nek megfelelő célnyelvi mondat is.

(9) Every white horse is old.
(10) every(x,white(x)$\wedge$horse(x), old(x))
(11) every(x), horse(x), old(x),white(x)
(12) Every old horse is white.

Tehát olyan reprezentációs formalizmus megalkotása a cél, amely figyelmen kívül tudja hagyni a hatóköri viszonyokat, ha azok irrelevánsak, de meg tudja jeleníteni, amikor szükség van rájuk. Ezt az MRS úgy tudja megvalósítani, hogy minden EP elé egy címkét tesz, amely a fastruktúra releváns részére utal. (13) reprezentációja (14) alatt olvasható.

(13) Every big white horse sleeps.
(14) h0: every(x, h1,h2), h1: big(x), h1: white(x), h1: horse(x), h2: sleep(x)

Három címke található a leírásban; h0 a fa csúcsát jelzi, ami onnan látszik, hogy a predikátum (*every*) után zárójelben nem csak a változó (x), hanem két címke is szerepel (h1 és h2), vagyis a fa itt két csomópontra ágazik. A big(x), white(x) és horse(x) EP-khez ugyanaz a címke tartozik, ami arra utal, hogy szemantikailag a három predikátum egyenrangú, így generálásnál a sorrendjük a célnyelv jellegzetességeiből számolható. Ha lapos reprezentációt szeretnénk kapni, egyszerűen elhagyjuk a címkéket, ha viszont szükség van az eredeti hatóköri viszonyokra, a címkék alapján a fastruktúra visszanyerhető.

Az MRS legnagyobb előnye, hogy a reprezentáció lehet alulspecifikált. A (15) alatti mondat két lehetséges olvasata: (i) minden kutyára található néhány olyan fehér macska, amit ő üldöz, illetve (ii) néhány fehér macskára az igaz, hogy minden kutya őket üldözi. Az első esetben a *minden*-nek van nagyobb hatóköre, míg a másodikban a *néhány*-nak. A két olvasatot egyetlen reprezentációban ábrázolva (16)-ot kapjuk.

(15) Every dog chases some white cat.
(16) h1: every(x, h3,h8), h3: dog(x), h7: white(y), h7: cat(y),
h5: some(y, h7,h9), h4: chase(x, y)

Attól függően, hogy h8-nak és h9-nek milyen értéket adunk, kapjuk az egyik, illetve a másik olvasatot. Ha h8=h5 és h9=h4, akkor nyerjük az (i) értelmezést (az *every*-nek lesz nagy hatóköre), viszont ha a h8=h4 és h9=h1 egyenlőségek teljesülnek, a (ii) értelmezéshez jutunk (a *some* hatóköre lesz a nagyobb). A h8= h9=h4 egyenlőség nem állhat fenn, mert ebben az esetben nem kapnánk fát.

Egy MRS struktúra tehát egy rendezett hármas, ahol az első komponens megmondja, melyik címke tartozik a fa csúcsához (top handle), a második komponens az EP-k halmaza, a harmadik pedig a címkékre (handle) vonatkozó megszorítások halmaza, amely lehet üres is (a pontos definíciók Copestake et al. (2005) 4. pontjában olvashatók). A (17) alatti mondat MRS-beli reprezentációja (18), amely a kitöltött harmadik komponens ellenére is alulspecifikált, mert a megszorítások halmazában csak azok az egyenlőségek szerepelnek, amelyeknek mindenképpen teljesülni kell; jelen esetben, hogy a határozó a fa csúcsán található (h0=h5) és az ige felett operál (h6=h7), illetve hogy a *minden* első „argumentuma" mindenképpen a *kutya* (h2=h4), a *néhány*-é pedig a *fehér* és a *macska* (h9=h11).

(17) Every dog probably chases some white cat.

(18) <*h*0, {*h*1:every(*x*, *h*2,*h*3), *h*4:dog(*x*), *h*5:probably(*h*6), *h*7:chase(*x*, *y*), *h*8: some(*y*, *h*9,*h*10), *h*11: white(*y*), *h*11: cat(*y*)}, {*h*0 =$_q$ *h*5, *h*2 =$_q$ *h*4, *h*6 =$_q$ *h*7, *h*9 =$_q$ *h*11}>

Ez a reprezentáció hat lehetséges hatóköri olvasatot enged meg, amelyek mind értelmesek, mivel azonban a legtöbb esetben (például fordítás) nem szükséges ezek között dönteni, felesleges lenne ezeket felsorolni. A legtöbb nyelvtechnológiai alkalmazás számára tehát ez a formalizmus az ideális választás, mert egyszerű (könnyen implementálható), mert pontosan azt engedi meg, ami a nyelvben ténylegesen létezik, és mert különböző szintjei vannak, amelyekből annyira hivatkozunk, amennyire éppen szükség van.

Fontos tulajdonsága a reprezentációnak, hogy a hagyományos leírás könnyedén átalakítható HPSG-szerű jegystruktúrává, ahol a jegyek – HOOK, RELS és HCONS – a rendezett hármas elemeinek felelnek meg, vagyis az első tartalmazza a (globális és lokális) csúcsát a fának, a második a különféle relációkat, a harmadik pedig a megszorításokat. Így az elmélet egyszerűen beágyazható jegystruktúrát használó nyelvtanokba. Leginkább a HPSG használja az MRS-t: az ábrázolásban a CONT (content, tartalom) jegy értékeként jeleníti meg az ilyen típusú struktúrákat (a beágyazás részletei Copestake et al. (2005) 6. pontjában olvashatók).

Mivel a formalizmust nagy lefedettségű nyelvtanok is használják, ezért képesnek kell lennie arra is, hogy nem tökéletesen formált, vagy nem teljesen elemzett bemenethez is reprezentációt tudjon rendelni. Ehhez többek között magasabb szintű alulspecifikáltságra van szükség, illetve arra, hogy az egyes relációk argumentumait ne kelljen minden esetben kimerítően felsorolni. Mindezeket az elmélet egy fejlesztés alatt álló változata az RMRS (Robust MRS) igyekszik megvalósítani (Copestake 2007). A két típusú reprezentáció kompatibilis egymással (egyik átírható a másikba), a különbség csupán annyi, hogy az RMRS a reprezentációban külön állításként jeleníti meg az argumentumokat, nem a reláció részeként. Például az MRS-beli l1:chase(e,x,y) állítás helyett azt adja, hogy l1:a1:chase(e), l1:a1:ARG1(x), l1:a1:ARG2(y), ahol az 'a1' a horgony (anchor) jele. Mindez például azért hasznos, mert ha nem található meg a szövegben egy régens valamelyik kötelező argumentuma (vagy azt valami miatt nem tudta a program elemezni), akkor is tudunk valamiféle reprezentációt társítani a mondathoz, minden megtalált információt meg tudunk jeleníteni, és nem azt a választ kapjuk, hogy a mondat nem grammatikus, és nem rendelhető hozzá szemantikai reprezentáció, mivel egy fontos szereplő hiányzik.

3.2.2.3. Gyakorlatban

A legkomplexebb HPSG-formalizmust használó alkalmazás a DELPH-IN (Deep Linguistic Processing with HPSG Initiative), amelyet többféle nyelvtechnológiai célra fejlesztnek. Bond et al. (2005) egy nyílt forráskódú, szemantikai transzfer alapú gépi fordító rendszerként mutatja be a DELPH-IN-t, amely létező forrásokat használ: elemzőket, generálókat, kétirányú nyelvtanokat és transzfer-motort. Első lépésként a japánról angolra

való fordítást valósították meg a segítségével. A cél először a mechanizmus működésének megmutatása volt, ezért kezdetben száznál kevesebb transzfer-szabályt, és csupán párezres transzfer-lexikont tartalmazott. A rendszer a forrásnyelvet (japán) elemzi egy szabály alapú forrásnyelvi nyelvtannal (JACY), a legjobb elemzést valószínűségi rangok alapján választja ki; kimenete pedig egy precíz, alulspecifikált (nyelvspecifikus) forrásnyelvi szemantikai reprezentáció (MRS_S). Ebből a transzfer-motor előállítja az alulspecifikált (nyelvspecifikus) célnyelvi (angol) szemantikai reprezentációt (MRS_T) újraíró szabályokkal úgy, hogy ha több megoldás is lehetséges a szabályalkalmazásnál, akkor több fordítást kapunk. Végül a célnyelvi generátor (ERG) angol nyelvű szöveget készít belőle.

A projekt általánosabb célja egy olyan mélynyelvészeti elemző megalkotása, amely szemantikai elemzésre (és szemantikai reprezentáció hozzárendelésére) is képes[21]. Nyelvészeti és statisztikai módszerek kombinációját alkalmazza, és minden nyelvre egységes formalizmust használ, amelyet a HPSG és az MRS implementációjával valósít meg. Korábban nem volt olyan mélynyelvészeti elemző, amely elég hatékony lett volna, ezért ez a projekt az alapformalizmus megalkotásán túl azt is kutatja, hogyan lehet statisztikai módszerek bevonásával a lefedettséget, a robusztusságot és a hatékonyságot növelni.

A számítógépes és a nyelvi erőforrásokat megosztják egymás között, így azok többféle alkalmazásban is használhatók, továbbá kevesebb időt és energiát igényel a rendszer új nyelvekre történő kidolgozása. A HPSG alapú (MRS-t is tartalmazó) formalizmust többféle fejlesztői környezetben is implementálták, hogy eltérő célok megvalósítására legyenek képesek. Az így kapott nyelvtanok és lexikonok azonban hordozható formátumban készülnek, vagyis könnyedén használhatók más projektekben. A leginkább használt erőforrások a következők:

(1) Linguistic Knowledge Builder (LKB): interaktív nyelvtanfejlesztő környezet típusos jegystruktúrával rendelkező nyelvtanok számára. Tartalmaz elemzőt, generálót és több specializált hibakeresőt (debuggert); illetve képes a különféle adatok (fák, jegystruktúrák, MRS struktúrák stb.) megjelenítésére és a nyelvtan, illetve a lexikon jólformáltságának ellenőrzésére. Lisp fejlesztőkörnyezetben implementálták, és mindenféle platform (Windows, Linux) alatt működik.

(2) A PET rendszer: nagy hatékonyságú elemző típusos jegystruktúrával rendelkező nyelvtanok számára. Ugyanazt a formalizmust használja, mint az LKB (annak egy komponensének tekinthető), viszont sokkal kevésbé erőforrás-igényes, robusztusabb, hordozható és könnyedén beágyazható más nyelvtechnológiai alkalmazásokba. A hatékonyság növelésének az (egyik) ára, hogy lényegesen korlátozottabb hibakeresővel rendelkezik. C++ fejlesztőkörnyezetben implementálták (illetve bizonyos részeit C-ben, hogy még hatékonyabb legyen), és alapjain már kereskedelmi forgalomban kapható termék is született.

(3) A nyelvi források között található több nagy lefedettségű nyelvtan különféle nyelvekre: elsődlegesen angol, német és japán, de már dolgoznak a francia, koreai, görög, norvég, portugál és spanyol nyelvtanokon is. Az angol nyelvre kidolgozott ERG (LinGO English Resource Grammar) fejlesztése 1993-ban kezdődött, és azóta is tart. Körülbelül tízezer lexémából álló kézzel írt lexikont tartalmaz, és külső szótárak is beolvashatók. A British National Corpus-on (BNC) mért fedése 57% volt, és bár azóta javult, nagyon messze van még a teljestől (Zhang et al. 2007). Más típusú szövegeken még ez az arány sem érhető el, tesztelték például telefonbeszélgetéseken, ahol 22%-os fedést tudtak vele elérni (mivel a spontán beszédben nagyon sok a hiba). Ezért hangsúlyozza Zhang et al. (2007) annak fontosságát, hogy egy elemző a mondatnál kisebb koherens nyelvi egységek elemzésére is

[21] A legfrissebb információk a projekt honlapján olvashatók: http://www.delph-in.net/

képes kell, hogy legyen, mert csak így lehetséges a nemfolytonos spontán beszéd kezelése. Ezért alkalmaz napjainkban egyre több nagy lefedettségre törekvő mélynyelvészeti rendszer részleges elemzést is.

Az egyéb nyelvekre fejlesztett elemzők közül mindenképpen érdemes még kiemelni a japánra működő JACY nyelvtant (Siegel–Bender 2004). Kidolgozása 1996-ban kezdődött, méretét tekintve az ERG-hez hasonlítható. A nyelv sajátosságai miatt ki kellett egészíteni szegmentálóval, morfológiai elemzővel és ismeretlenszó-kezelővel. Pusztán 2005 januárja óta tart a spanyol nyelvtan fejlesztése (Spanish Resource Grammar, SRG, Marimon et al. 2007), viszont mivel a nyelvtan írását segítő rendszerek köre folyamatosan bővül, és az elemzőkörnyezet is egyre hatékonyabban működik, a spanyol nyelvtan már jelenleg is viszonylag nagyméretűnek mondható (kb. 50 000 nyílt szóosztálybeli lexikai egység, amelyek 400 különböző típusba rendezhetők, 40 lexikai szabály a grammatikai viszonyokat változtató műveletek leírására, és 84 olyan szabály, amely az egységek kombinálódásáért és a kompozicionális szemantika (MRS) felépítéséért felelős). Olyan nyelvi jelenségeket képesek már kezelni, mint emelés és kontroll, alárendelő szerkezetek, hiányzó alany, passzivizáció, összehasonlító szerkezetek, klitikumok, kérdő mondatok, mondathatározók stb. A nyelvtan fejlesztését részben kézzel, részben korpuszból végzik.

(4) LinGO Grammar Matrix: egy olyan programcsomag, amely elősegíti, hogy minél egységesebbek legyenek az ebben a formalizmusban írt nyelvtanok. Minél homogénebb a már megírt grammatikák halmaza, annál hatékonyabb elemző írható, és csökken az új nyelvtanok megalkotásának költsége (idő és energia) is. A Matrix olyan konstrukciókat tartalmaz (magnyelvtan), amelyek elég univerzálisak ahhoz, hogy bármely nyelv grammatikájának a megírását segíthessék. Tartalmaz típusdefiníciókat, MRS-beli reprezentációkat és az összerakás módját, általános szabályokat, mint például derivációs és inflexiós (lexikai) szabályok és frázisstruktúra szabályok. Tartalmazza továbbá az alapvető konstrukciók típusait (fej–vonzat, fej–alany, fej–szabad bővítmény, mellérendelés stb.) és a specifikusabbakét is (vonatkozó mellékmondat, főnévi összetétel stb.), illetve a HPSG általános elveinek implementációját (fejjegy-elv, távoli függőségekre vonatkozó megszorítások stb.).

A rendszert a norvég nyelvre próbálták ki először (Bender et al. 2002), utána kezdték más nyelvek tanulmányozásával (angol, német, japán) kiépíteni a mai (univerzálisnak gondolt) rendszert. Az elmúlt években (a) a lexikai reprezentáción, (b) a szintaktikai általánosításokon kidolgozásán és (c) a szociolingvisztikai variánsok kezelhetőségén dolgoztak. (a) A lexikon építésének könnyebbé tételéhez nyelvfüggetlen típushierarchiát kellett kidolgozni, arra kellett megtalálni a választ, hogyan lehet a szintaktikai szubkategorizációt leképezni a szemantikai predikátum–argumentum struktúrára. Olyan lexikai szabályok megfogalmazására volt szükség, amelyek az általánosításokat ki tudják fejezni a lexikonon belül, illetve a nyelvek közötti szabályosságok is megragadhatók velük, mind inflexiós mind derivációs szinten. A lexikai egységeket és tulajdonságaikat relációs adatbázisban tárolják. (b) Szintaktikai általánosítások kidolgozására leginkább azért volt szükség, hogy a rendszer nagyobb szórendi szabadságot tudjon kezelni, mind a vonzatokra mind a szabad bővítményekre vonatkozóan. A lehető legabsztraktabban kellett megragadni például a kontroll, a klitikumok vagy az inkorporáció jelenségét, hogy mindenféle nyelvre alkalmazni lehessen. (c) A különféle szociolingvisztikai variánsok kezelésére pedig azért van szükség, hogy a rendszernek ne okozzanak problémát a grammatikalizálódott pragmatikai jelenségek, mint például udvariasság vagy empátia (a japán nyelv esetében ezeknek különösen nagy jelentősége van), illetve többféle dialektusban vagy regiszterben írt szövegek elemzésére is képes legyen.

Az MRS Matrixba ágyazásáról számol be például Flickinger–Bender (2003). Egy lexikai egység ábrázolásánál a SYNSEM attribútumon belül két jegy található: a CAT

(kategória) és a CONT (tartalom), ahol az első a szintaxis, a második a szemantika leírására szolgál. A CONT értéke MRS típusú jegystruktúra, amely három attribútumot tartalmaz: RELS (atomi állítások, melyeket lexikai egységek vagy szintaktikai szerkezetek vezetnek be, hatóköri relációkkal, szerepekkel), HCONS (megszorítások a hatóköri viszonyokra) és HOOK (atomi predikátumok attribútumai, amik alapján a jel más jelekkel kombinálódik). A RELS és a HCONS egy-egy lista, a HOOK pedig egy négy attribútummal rendelkező jegystruktúra, ahol mindegyik látható szemantikai elemre, amiket a jel behozhat további kompozíció céljára, található egy jegy: LTOP (legnagyobb hatókörű reláció), INDEX (a lexikai szemantikai fej által bevezetett változó, mint a lambda-változó), E-INDEX (egy további eseményváltozó, amely például predikatív frázisok esetében szükséges, például a kontroll jelenségének kezelésére), végül XARG (a külső látható argumentum szemantikai indexe (ha van), tipikusan az igei csoportnál az alany). A cikk részletesen ismerteti a szemantikai reprezentációra vonatkozó elveket, azok működését elméletben és az implementáció során.

Az elmúlt évekig tehát alapvetően a magnyelvtan fejlesztésére koncentráltak, amely olyannyira univerzális, hogy minden nyelv grammatikája használni tudja. A magnyelvtan azonban önmagában még nem képes mondatokat elemezni és generálni, előbb fel kell tölteni nyelvspecifikus információval, mint például szórend, és létre kell hozni az adott nyelv lexikonát. Hogy ezt a munkát segítsék, a korábban csak a magnyelvtant tartalmazó Matrixot kiegészítették különféle könyvtárakkal (libraries), amelyek különféle nyelvi jelenségek (nyelvenként) különböző lehetséges megvalósulásainak implementációját tartalmazzák (Bender et al. 2007). Így a nyelvtant fejlesztő felhasználóknak lehetőségük van arra, hogy egy webes felületen keresztül beállítsák a nyelvük paramétereit azáltal, hogy több nyelvtipológiai kérdést megválaszolnak; ami alapján aztán a rendszer összeállít egy kezdeti (immár az adott nyelvre jellemző) nyelvtant úgy, hogy az említett könyvtárakból a megfelelőt kiválasztja. Az így előállított nyelvtanok ugyan csak viszonylag kis fragmentumát fedik le az adott nyelvnek, de még így is nagy segítséget tudnak nyújtani az első lépéseknél. Ezek a könyvtárak olyan információkat tartalmaznak, mint az alapszórend (pl. SOV), hogy kötelezők vagy választhatók a különféle determinánsok, hogy NP vagy PP kategóriájú argumentumok találhatók-e a nyelvben, hogy hogyan fejezik ki a tagadást, vagy milyen az eldöntendő kérdések formája, mik a mellérendelési stratégiák stb. Ezeknek a jegyeknek a kombinálásával többszázezer lehetséges nyelvtan alkotható.

(5) LinGO Redwoods: hibrid rendszer, a mély és a sekély elemzést együtt valósítja meg, többféle statisztikai módszert használ, amellyel kiegészíti a nyelvészeti elemzést. Erre azért van szükség, mert amikor ipari alkalmazássá kezdték fejleszteni a rendszert, rájöttek, hogy egyes részfeladatokat hatékonyabban kellene tudni végrehajtani, mint például az egyértelműsítés vagy az elemzések közül a legoptimálisabb kiválasztása; illetve hogy a robusztusság növelése érdekében szükség van sztochasztikus modellek alkalmazására. A rendszer treebankok létrehozását is segíti, kettőt már készítettek is, amelyek körülbelül hétezer (teljesen elemzett) mondatot tartalmaznak.

(6) A Heart of Gold környezet: XML alapú rendszer, amely szintén ötvözi a mély és a sekély nyelvelemző technikákat, de ezeken belül a robusztus, többnyelvű és alkalmazás-központú (de természetesen továbbra is HPSG alapú) elemzésre koncentrál. Használ sekély szófaji egyértelműsítőt, csonkolásos eljárást, továbbá tulajdonnév-felismerőt is tartalmaz. Egységes infrastruktúrát nyújt olyan alkalmazások számára, amelyek RMRS és/vagy XML alapú természetesnyelv-feldolgozó komponenst tartalmaznak. Mélyelemzőnek a PET rendszert használja, és a fent említett különböző nyelvre megírt nyelvtanokon operál. További előnye, hogy egyéb (mély és sekély) nyelvtechnológiai komponensek könnyedén hozzáadhatók.

A fent leírt rendszer tehát alapvetően mélynyelvészeti elemzésre képes, de a gépi fordítás is megvalósítható a segítségével, akárcsak az LFG formalizmust használó ParGram

projekt esetében. Mindkét megközelítés azt az utat követi, hogy egy szigorúan formalizált keretet használva kidolgozzák a különböző nyelvek nyelvtanait, készítenek olyan elemzőket, amelyek az ilyen típusú bemenetet hatékonyan tudják kezelni, ezután már csak a transzferszabályokat kell kidolgozni, és rendelkezésre áll a gépi fordító. Érdekesség, hogy olyan rendszer is létezik, amely LFG és HPSG alapú elemzőket is használ: a norvég–angol gépi fordítást megvalósító szemantikai transzfer alapú LOGON (Lønning et al. 2004). Olyan célokra kezdték fejleszteni, ahol a fordítás minősége fontosabb a nagy lefedettségnél. A fordítás három lépésben történik. Az első a norvég mondat LFG alapú grammatikai és szemantikai mélyelemzése, amihez az XLE platformon fejlesztett NorGram-ot használják (amit a ParGram projekten belül fejlesztettek). Kimenete egy nyelvspecifikus logikai szemantikai reprezentáció (MRS). Majd ezeknek a reprezentációknak a transzfere történik nyelvspecifikus angol reprezentációkba (MRS). Végül a szemantikai reprezentációból angol nyelvű mondatot generálnak HPSG alapú elemző (generáló) segítségével (célnyelvtan: ERG, generátor: LKB). A projekt hosszú távú, bár célja egyelőre csak egy demonstráció fejlesztése, amely megmutatja, hogy a mechanizmus működik. A többértelműség kezelésére valószínűségi rangokat használ. Fontos jellemzője a modularitás, így mindig a legfrissebb elemzőt rakhatják a rendszer mögé.

3.2.3 LTAG és szemantika

Az elmélet rövid bemutatását Dang et al. (2000) alapján végzem. Az LTAG (Lexicalized Tree Adjoining Grammar) az *enyhén környezetfüggő* TAG *lexikalizált* változata. A nyelvtan nem más, mint *kezdő*- és *segédfák* véges halmaza (initial and auxilary elementary trees), illetve két művelet, amelyek segítségével ezek a fák kombinálhatók. A kezdőfák a nyelv minimális, nem rekurzív nyelvészeti struktúrái, mint például egy ige és a vonzatai. A segédfák a nyelv rekurzív struktúrái, mint például a prepozíciós módosítók, amik szintaktikailag beágyazott VP-ket eredményeznek. A két művelet a *behelyettesítés* (substitution) és a *csatolás* (adjoining). Az első azt jelenti, hogy a fa egy levelét egyszerűen helyettesítjük egy új fával, a második pedig azt, hogy egy fa valamely belső csomópontját helyettesítjük egy segédfával. Mindegyik fa hozzá van rendelve a nyelv egy lexikai egységéhez (ennyiben több, mint az eredeti TAG), ez a fa *horgonya* (anchor). A fa reprezentálja azt a tartományt, amely felett a lexikai egység közvetlenül specifikálhat szintaktikai és szemantikai megszorításokat, mint például alany–ige egyeztetés, illetve szelekciós megszorítások.

Az LTAG legnagyobb előnye, hogy valójában nem tartalmaz távoli függőségeket, mert minden viszony lokális, csak közbeékelődhetnek elemek (részfák) a csatolás művelete által; vagyis *kiterjesztett lokális tartománnyal* (extended domain of locality) rendelkezik. A nyelvtan tehát enyhén környezetfüggő, mert korlátlan távolságú függőségeket is megenged (de nincs szükség külön eszközre a kezelésükhöz). A lexikalizált fák kiterjesztett lokalitása miatt különösen alkalmas lexikális szemantikai ábrázolásra. Minden lexikai egység megfelel egy fának. Az igéhez tartozó fa lesz a mondat váza, amely az igét mint terminális szimbólumot máris tartalmazza. A mondat többi része helyettesítéssel vagy csatolással adódik a vázhoz (a megfelelő helyekre a deriváció során). Hogy hogyan rakódnak össze a különféle fák, az eltárolódik az ún. derivációs fában. Csomópontjai megfelelnek a lexikálisan lehorgonyzott fáknak, a nyilak pedig címkézve vannak azzal az információval, hogy hogyan kombinálódtak a fák. Mivel mindegyik lexikálisan lehorgonyzott kezdőfa megfelel egy szemantikai egységnek, a derivációs fa nagyon hasonlít egy szemantikai (függőségi) reprezentációra.

A szemantika LTAG-be integrálására többféle megközelítés született. Van, amely párhuzamosan hozza létre a mondat szemantikáját és szintaxisát; egy másik szerint a

derivációs fa szemantikája úgy írható le, mint fák összekapcsolásának (attachment) a halmaza, ahol egy ilyen összekapcsolás nem más, mint lapos (flat) szemantikai reprezentációban írt formulák konjunkciója, amellyel a hatóköri többértelműségek is kezelhetők.

Adekvát kézzel írt szemantikai reprezentáció készítésének nehézsége az egyik ok, ami miatt csak jól körülhatárolt aldomaineken működnek az NLP alkalmazások. Többféle lexikonmegközelítés létezik, de felfedezhető némi egységesség. A legellentmondásosabb terület a poliszémia kezelése: hogyan lehet a különféle értelmeket megkülönböztetni és megragadni. Számos alkalmazás használ valamiféle igei lexikont (VerbNet), amely explicit szintaktikai és szemantikai információt tartalmaz az egyes lexikai egységekről. Több közülük (Dang et al. (2000) is) Levin igei osztályait használja, hogy szisztematikusan megalkossa a lexikai bemeneteket. Levin és követői azt vallják, hogy a szintaktikai keretek direkt leképezései a mögöttes szemantikának, ezért hasznos az igéket osztályozni. Így általánosításokat lehet megragadni, mint például thematikus szerepek, lehetséges vonzat-keretek, szelekciós megkötések. Ha az osztályokat hierarchiába rendezik (akár az eredeti osztályozás kis módosítása árán), azzal elérhető, hogy az osztályok koherensek legyenek (ugyanazokkal a szintaktikai és szemantikai jegyekkel rendelkezzenek).

Létezik már olyan elemző, amely képes előállítani a derivációs fát, Dang et al. (2000) pedig azt a célt tűzte ki, hogy kidolgoznak hozzá egy minél pontosabb kompozicionális szemantikát. Olyan szemantikai predikátumokkal dolgozik, amelyek elég primitívek ahhoz, hogy sokukat több fánál is használhassák. Minden *E* eseményt szétbontanak egy három részből álló eseménystruktúrára: előkészületi (preparatory) fázis (*during(E)*), tetőpont (culmination, *end(E)*) és eredmény (consequent, *result(E)*); és az adott igénél azt szerepeltetik, ami releváns. Ezzel bonyolultabb eseménystruktúrájú igeosztályok leírására is képesek, például kezelni tudják, hogy a *break* típusú igék tárgyára más jellemzők igazak az esemény bekövetkezése előtt és után. A kezdőfák az ige alapjelentését hordozzák mindegyik típusnál. Például a *run* csoporthoz intranzitív és tranzitív fák tartoznak, amelyek szemantikai predikátumai a következők: $motion(during(E), X_{arg1})$ az intranzitív esetben, és $cause(during(E2), X_{arg0}, E1) \wedge motion(during(E1), X_{arg1}))$ a tranzitív változatnál. A segédfákhoz is tartozik predikátum, például a *to* prepozícióhoz, a következő szemantikai leírással: $goal(end(E), X_{arg0,arg1}, X_{arg1})$. Amikor több fa összekapcsolódik, betöltődnek a megfelelő értékek a megfelelő változókba. A *John ran to the park* mondathoz így a következő szemantikai reprezentációt kapjuk: $motion(during(E), john) \wedge goal(end(E), john, park)$, a *Bill ran the horse to the park* mondathoz pedig a $cause(during(E2), bill, El) \wedge motion(during(El), horse) \wedge goal(end(El), horse, park)$ leírást. Az igékhez kiterjesztett jelentés is tud társulni, ha szabad bővítményekkel látjuk el őket. Az elmélet ezt a csatolás és a szemantikai predikátumok konjunkciója segítségével tudja kezelni. Például a *hit* osztály alapvetően nem tartalmazza azt a jelentésmozzanatot, hogy a tárgy mozog, de ha egy PP csatolódik (pl. *across NP*), az ige kiterjesztett jelentést nyer, és így egy másik Levin osztály (*throw*) tagjává válik.

Az LTAG tehát megfelelő választásnak tűnik, ha egy alapos kompozicionális szemantikai reprezentáció megalkotása a cél, amely a távoli függőségeket is ugyanazzal a mechanizmussal oldja meg, mint a lokális viszonyokat. A csatolás művelete pedig lehetővé teszi a szabályos poliszémia kezelését, amely kulcskérdése bármely szemantikai elméletnek.

Romero and Kallmeyer (2005) is a derivációs fán számolja a szemantikát, mert a fák szemantikailag minimálisak, a derivációs lépések pedig predikátum–argumentum viszonyok betöltésének felelnek meg. Azonban ez néha problematikus, mert a derivációs fa nem mindig nyújt elégséges információt ahhoz, hogy a kívánt szemantikai függőségeket tökéletesen megalkossuk. A hiányzó információ (hogy mely lexikai egységek tartoznak össze) kinyerhető szemantikai jegyek unifikációjából. Romero and Kallmeyer (2005) a derivációs

fa részfáihoz nem csupán egy-egy szemantikai reprezentációt, hanem egy-egy szemantikai jegystruktúrát is hozzárendel, és a jegyek unifikációja alapján megkapjuk a szemantikai reprezentációban lévő változók megfelelő hozzárendeléseit.

A szemantikai reprezentáció formulák halmazából (típusos lambda-kifejezések) és hatóköri megszorításokból áll. Mindegyik szemantikai reprezentáció hozzá van kötve egy szemantikai jegystruktúrához. Ezek típusos jegystruktúrák, ahol az egész *sem* típusú, jegyei pedig a fa csomópontjai (0 a fa gyökere, 1, 2, …, 11, 12 stb.), melyek értékei *tb* (top-bottom) típusúak, ahol a jegyek *T* (top) és *B* (bottom), amelyek értékei *bindings* típusú jegystruktúrák. Egy *bindings* típusú jegystruktúra három jegyből áll: *I*, értékei individuális változók, *P*, értékei propozíciós címkék és *S*, értékei szituációs változók. A szemantikai kompozíció csak jegy-unifikációból áll. Az unifikáció során a szemantikai reprezentáció bizonyos változói értéket kapnak, majd végül a szemantikai reprezentációk uniójával előáll az alulspecifikált reprezentáció, amely a hatóköri többértelműségeket is kezelni tudja.

Implementációja ugyanazokat a mechanizmusokat használja, mint az egyszerű LTAG elemző (FTAG, feature-structure based TAG), néhány apró különbséggel. A gyakorlatban (némi megszorítással) az FTAG $O(n^6)$ komplexitása csak konstansokkal módosul, ha a bemutatott szemantikával kiegészítjük. Az egyértelműsítési feladat elvileg NP-teljes, de kidolgoztak rá (szintén némi egyszerűsítéssel) polinomiális algoritmust.

3.2.4 Kategoriális Nyelvtan a nyelvtechnológiában

A különféle kategoriális nyelvtanokat a 2.1.2 pontban már részletesebben ismertettem, itt csak újra megemlítem a legfontosabb jellemzőiket, majd röviden bemutatom, milyen eredményeket értek el a számítógépes nyelvészet területén. A nyelvtantípus matematikai alapokra épül, klasszikus változata *környezetfüggetlen*, egyes kiterjesztései *enyhén környezetfüggő* grammatikát határoznak meg. A fa leveleihez és csomópontjaihoz is *kategóriákat* rendel, amelyek megmutatják, milyen más egységekkel kombinálódhat az adott egység (nincs különbség tehát lexikai tétel és frázis között). Így a *valencia* (egy kategória aktuális kombinációs képessége) akárcsak a HPSG-ben, ebben az elméletben is központi szereppel bír. Az összes ismert lexikalista elmélet közül a kategoriális nyelvtan a leginkább lexikonközpontú, hiszen a lexikai tételek kombinálódására pusztán egy-két egyszerű szabályt fogalmaz meg. A legfontosabb közülük a *függvényalkalmazás* (function application), amely a nyelvtan mindegyik változatában szerepel. Ha pusztán ezt a szabályt alkalmazzuk, az környezetfüggetlen nyelvtant eredményez, de ha kiegészítjük *unifikációval*, enyhén környezetfüggő grammatikát kapunk (radikális lexikalizmus, Karttunen 1986). A GASG még a függvényalkalmazás műveletét sem alkalmazza, így éri el a totális lexikalizmust. A Kategoriális Nyelvtannak nem célja a redundanciamentesség (szóosztályokra vonatkozó általánosítások kifejezése), és nem foglalkozik a lexikon strukturáltságának (lexikai tételek gazdaságos leírásának) problémájával.

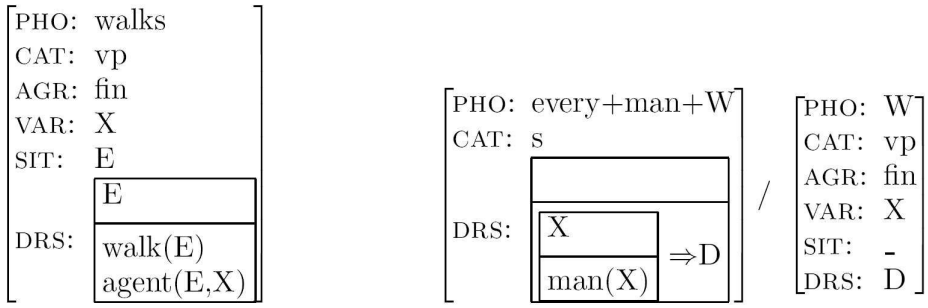
Számítógépes alkalmazásaik statisztikai módszereket használnak. A legtöbb ebben a formalizmusban írt elemző CCG alapú (Combinatory Categorial Grammar, pl. Steedman–Baldridge 2007), amely egy enyhén környezetfüggő formalizmus, ahol a szintaxis és a szemantika közötti interfész teljesen transzparens. Legfontosabb tulajdonsága, hogy könnyedén kezeli a távoli függőségeket is. A CCG úgy éri el az enyhén környezetfüggő szintet, hogy (a függvényalkalmazás mellé) további műveleteket (combinatory rules) von be, melyek közül a két legfontosabb a *típusemelés* és *függvénykompozíció*. Mivel minden szintaktikai művelet, amit használ, univerzális, és minden (nyelvspecifikus) egyéb jegyet a lexikonban tárol, ezért az elmélet teljesen (fully) lexikalistának tekinthető. A kategóriák morfológiai információt is tartalmazhatnak, például szám vagy eset.

Létezik CCG alapú treebank is (CCGbank), amely normálformában lévő CCG derivációkat tartalmaz. A PENN treebankból – ami a legnagyobb kézzel készített korpusz angol nyelvre – automatikusan állították elő, és alkalmas arra, hogy statisztikai elemzők (vagy akár egyéb alkalmazások is) az elemzéskor használhassák.

CCG-re épülő elemzőt mutat be például Clark et al. (2002). Abból indul ki, hogy a derivációs struktúrára valójában nincs is szükség, hiszen az elemzés végén rendelkezésünkre áll egy interpretálható predikátum–argumentum szerkezet, illetve a hozzá kapcsolódó függőségi viszonyrendszer (amelyek minden szükséges információt rögzítenek). Az elemzőjük így közvetlenül a végleges (derived) struktúrát állítja elő (amely a távoli függőségeket is tartalmazza). Először előelemzés történik az ún. supertagger segítségével, majd alulról felfelé haladó elemző algoritmust alkalmaznak. A következő műveleteket használja: függvényalkalmazás (előre és hátrafelé), általánosított függvénykompozíció (előre és hátrafelé) és típusemelés. Alkalmaznak továbbá egy mellérendelő szabályt, amely azonos kategóriájú kifejezések koordinálására képes. Az elemző pontos, hatékony és nagy lefedettségű. A CCGbank ismeretlen mondatainak közel 98%-át képes elemezni, és annyiban többet nyújt, hogy távoli függőségek kezelésére is képes.

Traat–Bos (2004) kombinálja a CCG-t jegystruktúrák unifikációjával, és információs szerkezetet is beépít a szintaktikai és szemantikai reprezentációkba. A fő hangsúlyt a téma, réma és a fókusz kezelésére helyezi, a szemantikai reprezentációhoz pedig a DRT-t alkalmazza. Az így létrejött kategoriális nyelvtantípus, az UCCG (Unifikációs CCG, amely tehát a CCG, az UCG és a DRT kombinációja), így alkalmassá válik prozódiailag annotált szövegek elemzésére és generálására (így dialógusrendszerek pontosságának javítására). Korábban az információs struktúra implementálásával nem igazán foglalkoztak a számítógépes nyelvészetben. A CCG-től a többféle művelet használatát és az intonáció kezelését veszi át; az UCG-től pedig a jegystruktúrákat, így a szintaxis és a szemantika együtt épül fel az unifikáció során. DRT-formalizmust használ, de kiegészíti, hogy prozódiai információ is kódolható legyen benne. A lexikonja kéttípusú jelet tartalmaz: alap (basic) és összetett (komplex) jeleket. Egy jelnek, kategóriájától függően, többféle attribútuma lehet. A kötelező jegyek: PHO (fonológiai forma), CAT (kategória) és DRS (szemantikai reprezentáció). Az opcionális jegyek pedig: AGR (inflexiós tulajdonságok), VAR (entitásokra utaló diskurzus-referensek) és SIT (eseményekre utaló diskurzusreferensek), ez utóbbi kettő mindig változó. A következő példa egy alap jelet (VP-t) és egy összetett jelet (NP-t) mutat be (nagybetűvel a változókat, kis betűvel a konstansokat jelöli).

(19) Az angol *walks* VP, illetve *every man* NP UCCG-beli ábrázolása.



Jelenleg négy műveletet vettek át a CCG-ből, amelyek a következők:

(1) előre mutató függvényalkalmazás (forward application: X/Y Y ⇒ X)
(2) visszafele mutató függvényalkalmazás (backward application: Y X\Y ⇒ X)
(3) előre mutató függvénykompozíció (forward composition: X/Y Y/Z ⇒ X/Z)
(4) visszafele mutató függvénykompozíció (backward composition: Y\Z X\Y ⇒ X\Z)

Az információs szerkezetet leginkább a prozódia mutatja, de egyes nyelvekben egyéb tényezők (szórend, specifikus lexikai egységek) is közrejátszanak. A CCG-ben kétféle dallamhangsúllyal (pitch accent) lehet megjelölni a különböző szavakat (L és H), így (információs szerkezet tekintetében) különféle olvasatokat kaphatunk ugyanazon szósor elemzésekor. Egy szó lehet jelöletlen, illetve jelölt téma vagy réma hangsúllyal, ami három lexikai bemenetet eredményez minden szóra. Az UCCG-ben ezzel szemben a hangsúly egy külön lexikai egység, amely valamely szóval kombinálódva hangsúlyjelölt alakot eredményez. Az információs szerkezet kezeléséhez két további jegy bevezetése szükséges: INF (információs szerkezet) és FOC (fókusz). Mindkettő értéke lehet változó vagy konstans, az INF esetében téma-, réma- vagy frázishangsúly a lehetséges értékek, míg a FOC esetében az, hogy hordoz-e erős hangsúlyt az adott lexikai egység. A jegyek értékei a DRS-be is bekerülnek. A rendszert az angol nyelv egy kis fragmentumára implementálták is, a program prozódiailag annotált szövegeket elemez, és kimenetül olyan DRS-eket ad, amelyek az információs szerkezetről is hordoznak információt.

A CCG egy másik unifikációs alapú implementálásáról számol be Beavers (2003). Érdekessége, hogy a rendszert az LKB (Linguistic Knowledge Building) nyelvtanfejlesztő környezetben készítették, amelyet korábban csupán HPSG alapú nyelvtanok implementálására használtak. Az LKB-t elvileg úgy fejlesztették, hogy alkalmas legyen bármilyen típusos, jegy alapú, unifikációs nyelvtan számára, elméleti háttértől függetlenül. Az említett CCG implementáció egyik célja éppen az, hogy megvizsgálja, hogy gyakorlatilag is alkalmas-e az LKB kategoriális nyelvtanok fejlesztésére, illetve esetleg milyen kiterjesztésekkel az, hogy a későbbiekben más CCG alapú alkalmazások is használhassák a bizonyítottan hatékony fejlesztőkörnyezetet.

3.2.5 Konstrukciós nyelvtan

A Konstrukciós nyelvtan egy *egyszintű* grammatika, ahol a nyelvi szabályszerűségek leírásának egyetlen formája a *konstrukció*, amely „egy általánosítás formai és jelentéstani tulajdonságok konvencionális együttjárásáról”. „A nyelvtan nem más, mint rengeteg tartalmi és formai általánosítás együttjárásának és kölcsönös gátlásának bonyolult szövedéke.” (Kálmán et al. 2003: 112) A nyelvi objektumokat konstrukciók engedélyezik. Egy konstrukció közvetlen általánosítása azon objektumoknak, amelyekben előfordul, és további konstrukciók jelenlétét (teljesülését) követelheti meg. Egy nyelv konstrukciós nyelvtana: „konstrukciók gyűjteménye, strukturált adatbázisa, melyek megfelelő módon kombinálva az adott nyelv jólformált kifejezéseit adják” (Trón 2001). Ez az elmélet is elutasítja a mag és periféria szétválasztását, azt vallja, hogy az általános elveknek az egyedi esetek rögzítése alapján kell körvonalazódniuk. Ez ugyan utópisztikus kicsit, mindenesetre fontos, hogy az általános elvek és az egyedi esetek kezelése egyformán jól működjön egy rendszerben (Trón 2001).

A konstrukciók *öröklődési hálózatba* rendeződnek, az általánosabb konstrukciók a hierarchiában magasabban helyezkednek el, mint a speciálisabbak. Ez azért is hasznos, mert az öröklődési hierarchiákat a matematikusok részletesen tanulmányozták, és kialakultak a legfontosabb módszerek formális kezelésükre. Hatékony mondatelemző és mondatlétrehozó rendszereket készítettek ilyen hierarchiákra támaszkodva (Kálmán 2001). A konstrukciós nyelvtannak nem központi problémája a grammatikalitás, meg tudja magyarázni, hogy egyes (nem grammatikus) formákat miért használunk mégis bizonyos jelentésben.

Kálmán (2001) a *kompozicionalitás* problémáját is tárgyalja, illetve hogy hogyan érvényesül az elv az unifikációs grammatikákban, különös tekintettel a konstrukciós nyelvtanra. A kompozicionalitás elve kimondja, hogy egy összetett szerkezet jelentése az alkotóelemeinek a jelentésétől és az összerakás módjától függ (csak). Ez ebben a formában

egy gyenge követelmény (hiszen mi mástól is függhetne). Az erős kompozicionalitás az, amikor minden alkotórész és összerakási mód hozzáad valamit a jelentéshez, ami formálisan a jelentéskomponáló függvények additivitását jelenti. Az összetett kifejezés jelentése minden részénél informatívabb (vagy legalább annyira informatív). A Konstrukciós nyelvtan ezt is kielégíti, hiszen a konstrukciók által előírtak együttes teljesülése a jelentésekre vonatkozóan éppen a jelentésbeli előírások együttes érvényesítését jelenti. A követelmények egyidejű érvényesítésének matematikai modellje az *unifikáció* (additív) művelete, amely (egyik) definíciója: akkor és csak akkor igaz, hogy *a* és *b* unifikációja *c*, ha *c* az *a* és *b* legnagyobb alsó korlátja, vagyis $c \leq a$, $c \leq b$, és minden olyan *d*-re, amelyre $d \leq a$ és $d \leq b$, $d \leq c$. Vagyis az unifikáció eredménye definíció szerint informatívabb a két unifikált komponensnél (és az ilyenek közül ő a legkevésbé informatív). Ezek szerint tehát „a konstrukciós nyelvtan lényegéből következően minden eddig ismertnél megszorítottabb, függetlenül is motivált kompozicionalitás-fogalmat foglal magába, amelynek lényege, hogy a jelentések szigorúan a formai szerkezettel párhuzamosan, unifikáció segítségével kombinálódnak" (Kálmán 2001: 19).

A konstrukciós nyelvtan alapjain is készülnek nyelvtechnológiai alkalmazások, a magyar nyelvre például a Kálmán et al. (2003) által körvonalazott rendszer.

A konstrukciós nyelvtan és a totális lexikalizmus (a GASG) bizonyos szempontból nagyon hasonlóak, a különbség, hogy a GASG mindig lexikai egységekhez (szavakhoz, illetve morfémákhoz) rendeli azokat a jegyeket, amelyeket a konstrukciós nyelvtan különféle konstrukciókban kódol. Például az alany–állítmány konstrukció helyett a GASG a predikátumoknál veszi fel azt a követelményt, hogy szükség van egy alanyra (vagy a pro-drop nyelvek esetében időnként csupán valamiféle egyeztető morfémára), azt pedig, hogy állítmányt minden (grammatikus és teljes) mondat tartalmaz, az időjel keresésével ragadja meg. A GASG-ben tehát a metaszabályok is lexikaiak (lokálisak), az időjel tulajdonságaiból adódik, hogy keresni kell egy igét, az ige tulajdonságaiból, hogy (minimálisan szemantikailag) keresni kell egy alanyt, stb. A totális lexikalizmus felfogható úgy is, mint egy speciális konstrukciós nyelvtan, ahol csak lexikai konstrukciók vannak.

Ebben a fejezetben ismertettem a lexikalizmus elvét és röviden bemutattam a legfontosabb lexikalista elméleteket, kitérve arra is, milyen eredményeket értek el ezidáig a nyelvtechnológia területén. Az irányzat legfontosabb jellemzői, hogy a szótári komponenst helyezi előtérbe, elutasítja a transzformációkat, deklaratív, vagyis jólformáltsági megszorításokkal dolgozik, és egyszintű. Fontos a pszichológiai realitás és a számítógépes implementálhatóság. Alapjain minden eddiginél pontosabb nyelvelemzők fejlesztése lehetséges, amelyek (számítástechnikai szempontból könnyen kezelhető) szemantikai reprezentációt is képesek a mondatokhoz rendelni. További előnye a lexikalista elméletek alkalmazásának, hogy nagyobb fokú univerzalitás érhető el velük, mint a szintaxis-központú elméletekkel, aminek gyakorlati haszna, hogy ha a különböző nyelvekre ugyanazt a formalizmust használjuk, nagyfokú párhuzamosságot lehet megvalósítani, amivel a gépi fordítás is könnyebben elérhetőnek tűnik. A hatékonyság és a lefedettség növelése érdekében a lexikalista alapú rendszereket is kiegészítik statisztikai módszerekkel, és hogy a rendszerek kellően robusztusak legyenek, a mélyelemzés mellett gyakran némi sekélyelemzésre is szükség van. A magyar nyelvre egyelőre nem készült (nagy lefedettségű) szigorú elméleti alapokra épülő lexikalista elemző, amely jelentéstani reprezentációt is képes lenne társítani a mondatokhoz. A következő fejezetben ismertetem a totálisan lexikalista GASG-t, és részletesen bemutatom azt a kis adatbázison működő elemző és gépi fordító programot, amelyet az elmélet implementálásaként készítettünk.

4 GeLexi-projekt

A GeLexi kutatócsoport 2000-ben alakult a PTE BTK Nyelvtudományi Tanszékén (a GeLexi a Generatív Lexikon rövidítése), azzal a céllal, hogy „kipróbálja” a totálisan lexikalista megközelítést a számítógépes nyelvészet területén[22]. Kezdetben Alberti Gábor, Balogh Kata és Kleiber Judit voltak a csoport tagjai, majd 2002-ben csatlakozott Viszket Anita, végül 2003-ban Balogh Kata kivált a csoportból[23]. Alapvető célunk elméleti: egy minden eddiginél szótárközpontúbb nyelvtan (a GASG) *legitimálása* azáltal, hogy működő implementációt készítünk hozzá. Ezzel párhuzamosan célunk *elméleti nyelvészeti ötletek implementálhatóságának vizsgálata*, kipróbálni, hogy egy újfajta, *„totálisan” lexikalista* megközelítés mennyire tud eredményes (és hatékony) lenni a számítógépes nyelv-feldolgozásban. (Nem törekszünk tehát (egyelőre) piaci termék létrehozására.)

Az elmúlt évtizedekben megfigyelhető erőteljes lexikalista fordulat „végső” állomása a totális lexikalizmus, ahol már frázisstruktúra sem épül, és nincsenek szintaktikai műveletek, a mondatok pusztán *unifikáció* segítségével épülnek össze. Nincsenek „nagy szabályok”, minden követelmény *lokális*, amelyek együttes kielégülése eredményezi a grammatikus mondatokat (*deklarativitás, egyszintűség*). A *szemantika* a központi komponens, a szintaxis elsődlegesen azt a célt szolgálja, hogy jelentéstani reprezentációt tudjunk a mondatokhoz társítani. Kiindulópontja a nem hierarchikus felépítésű diskurzusreprezentációs struktúra (DRS), amihez a totálisan lexikalista GASG egy kompatibilis (szintén nem hierarchikus) szintaxist kínál.

Nem állítjuk, hogy az így nyert teljesen homogén (csak lexikonból álló) és hierarchiamentes megközelítés feltétlenül eredményesebb, mint az egyéb (nagyon sikeres) lexikalista elméletek; azonban több okból is érdemes kipróbálni. Először is elméletileg érdekes kérdés, hogy hasznos-e a bizonyítottan jól működő lexikalizmust a végsőkig fokozni; illetve hogy megfelelően hatékony tud-e lenni a gyakorlatban egy elméleti szempontból feltétlenül elegánsabb homogén nyelvtan, vagy a nyelv olyan, hogy szótárra és szintaktikai szabályokra egyaránt hivatkozik (Pinker 1999). Gyakorlati szempontból pedig azért tűnik ígéretesnek a megközelítés, mert minden eddiginél univerzálisabb keretet biztosít a nyelvek leírásához – ebből a szempontból az optimalitáselmélettel rokonítható – azáltal, hogy minden nyelvben ugyanazokat a megszorításokat tételezi fel, csak a hozzájuk rendelt rangparaméterek térhetnek el, ami a nyelvek közötti változatosságot eredményezi; illetve hogy minden eddiginél közvetlenebb módon képes szemantikai reprezentációt társítani a mondatokhoz, hiszen nem kell feloldania egy hierarchikus szintaxis és egy „lapos” szemantika közötti ellentmondást.

Az implementációt *Prolog* programnyelven készítettük, amelyet a mesterséges intelligencia programnyelvének is szoktak tekinteni, mert beépítve tartalmazza azokat a mechanizmusokat, amelyekre ilyen típusú feladatok elvégzésekor szükség van. A Prolog lelke is az unifikáció, akárcsak a lexikalista elméleteké, illetve a keresést hatékony visszalépő (backtracking) algoritmussal segíti. Azonban ha nagy adatbázist tartalmazó piaci termék létrehozása a cél, a Prolog már nem bizonyul kellően hatékonynak. Mivel a GeLexi projekt célja a totális lexikalizmus gyakorlatban való kipróbálása kis adatbázison, számunkra

---

[22] A kutatásokat nagyban segítette a T 38386 számú OTKA pályázat, amely segítségével az eredményeinket számos konferencián be tudtuk mutatni.

[23] Alberti Gábor az elméleti oldal minél részletesebb kidolgozásáért (és az elméleti tisztaság betartatásáért) volt felelős, Viszket Anita szintén elméleti szakemberként volt jelen, alapvetően a kritikusi szerepet vállalta; az implementációs munkát Balogh Katával végeztük (miután ő kivált a csoportból, egyedül folytattam).

a Prolog volt a tökéletes választás. (Következő lépésként, amikor már a hatékonyság is szempont lett és az adatbázis méretét is növelni akartuk, mi is áttértünk hatékonyabb programnyelv alkalmazására.)

Az így elkészült program bemenete egy magyar (illetve még kisebb adatbázison angol) nyelvű szósor, amelyről az elemző el tudja dönteni, hogy grammatikus-e, és amennyiben az, különféle reprezentációkat tud hozzá társítani: morfológiai, szintaktikai és szemantikai szerkezetet. A szintaxis (mivel frázisstruktúrát nem építünk) gyakorlatilag a függőségi viszonyokat tartalmazza, a szemantika pedig egy DRS-szerű reprezentáció. A program képes továbbá a két nyelv közötti gépi fordításra is, amelyet az elemző kétirányú használatával (elemzés és generálás) valósítunk meg, a szemantikai reprezentáción, illetve az ún. kopredikációs hálózaton keresztül. A programon kiértékelést egyelőre nem végeztünk, mert túlságosan kevés lexikai egységet tartalmaz, és a nyelvi jelenségek kis körét képes csak kezelni. Amit ezen a „játékgrammatikán" meg akartunk mutatni, az sikerült, hogy a totális lexikalizmus alapjain készülhet elemző, a mechanizmusok működnek, és érdemes az elmélet további implementálásán dolgozni, immár hatékonyabb környezetben. Az így elkészülő elemzőn már fogunk kiértékelő eljárásokat futtatni.

4.1 GASG

A GASG (Generative/Generalized Argument Structure Grammar, Alberti 1999) tehát egy egyszintű deklaratív nyelvtan, amely „totálisan" lexikalista, mivel semmiféle szintaktikai műveletet nem alkalmaz, minden információt a lexikonban tárol, és az elemek összeépülését kizárólag az unifikáció vezérli. Egy módosított unifikációs kategoriális nyelvtannak is tekinthető, amelyből még az egyetlen művelet – a függvényalkalmazás – is törölve lett. A Karttunen (1986) által javasolt radikális lexikalizmust viszi még tovább, amelyben megfogalmazódik, hogy ha megfelelően gazdag a lexikon, akkor az unifikáció segítségével úgy tudnak összeépülni a mondatok, hogy a frázisstruktúra már nem hordoz további információt, így az gyakorlatilag feleslegessé válik, továbbá hamis többértelműségekhez is vezet. Számítógépes nyelvészeti munkákban is (pl. Schneider 2005) találkozhatunk azzal a nézettel, hogy a frázisstruktúra redukálása hasznos lehet, sok alkalmazás pusztán a hatékonyság növelése miatt támaszkodik rá, mert nélküle függőségi nyelvtant kapunk, amely – mivel nem tartalmaz a szórendre vonatkozó megszorításokat – számítástechnikai szempontból nem hatékony. A GASG azonban számot ad a szórendről is – rangparaméterek segítségével –, így a frázisstruktúra feladása nem feltétlenül eredményez exponenciális futásidejű elemző algoritmust.

4.1.1 Előzmények

A GASG megalkotását az az igény motiválta, hogy találjunk egy olyan kompozicionális szintaxis–szemantika párt, ahol a szemantikai oldalon nem a hagyományos (montagoviánus) szemantikai reprezentáció található, hanem egy annál modernebb, dinamikus szemantikai elmélet. Mivel a legalkalmasabbnak erre a célra a DRT (van Eijck–Kamp 1997) tűnt, amellyel a hierarchikus felépítésű szintaxiselméletek csak a lambda-absztrakció bevonásával tehetők kompozicionálissá, létre kellett hozni egy olyan elméletet, amely nem hierarchikus szintaxison alapul, ez lett a GASG. Ebben a pontban röviden ismertetem, miért alkalmasabb a jelentéstan leírására a DRT, mint a montagoviánus szemantika, majd (szintén röviden) bemutatom a DRT-t, végül az LDRT-t, amely a DRT továbbfejlesztésének tekinthető, és amelyet a GASG (és az implementációja) is használ.

Montague a chomskyánus szintaxissal kompozicionális szemantikát dolgozott ki a '70-es években, aminek óriási volt a jelentősége: született végre egy formális szemantika-

elmélet, amelyet a Frege-féle kompozicionalitás szellemében hozzá lehetett rendelni az akkori legmodernebb szintaxishoz (algebrai homomorfizmust lehetett szintaxis és szemantika között létesíteni). Azóta azonban több kritika érte Montague szemantikáját, leginkább annak statikus volta miatt. A mondatokat nem lehet önmagukban, kontextus nélkül értelmezni, továbbá minden kimondott szó megváltoztathatja magát a kontextust, a világról való tudásunkat; a mondat jelentése tehát inkább annak ún. *kontextusváltoztató képességében* ragadható meg. Nem alkalmas továbbá a Montague-féle szemantika több mondatból álló szövegek elemzésére. A következő példa ezt érzékelteti: (20) logikai átírása (szemantikai reprezentációja) (21) kellene, hogy legyen, de a hagyományos módszerekkel ezt nem kaphatjuk meg, (22)-t kapjuk, mert a mondat végén az egzisztenciális kvantor hatóköre lezárul, így az *elmosolyodott* változóját már nem köti semmi.

(20) Belépett egy férfi. Elmosolyodott.
(21) $\exists x$ (férfi(x) $\wedge$ belépett(x) $\wedge$ elmosolyodott(x))
(22) $\exists x$ (férfi(x) $\wedge$ belépett(x)) $\wedge$ elmosolyodott(x)

A probléma nem pusztán technikai jellegű, hanem egy lényegi hibára (hiányosságra) mutat rá. (20) első mondata ugyanis nem csupán a létezését állítja egy belépő férfinak, hanem azt is legitimálja, hogy a továbbiakban róla beszéljünk. Ennek a megragadására a statikus szemantikaelméletek nem képesek, már csak azért sem, mert az egzisztenciális kvantor jelentése 'létezik egy vagy több…', tehát nem csak egy belépő férfiről van szó, így nem is lehet a mondatot egyetlen konkrét férfira utalva folytatni. Persze (20) első mondata valóban nem állítja, hogy csak egyetlen férfi lépett be, mégis vissza tudunk rá utalni.

A logika (Montague-féle szemantika) tehát nem tudja megragadni a nyelv több fontos tulajdonságát: hogy bizonyos főnévi csoportokra vissza lehet utalni, hogy azok régi vagy új entitásokat jelölnek (*egy fiú* vs. *a fiú*), és legfőképpen, hogy a kimondott szó megváltoztatja az interpretáció alapjául szolgáló világot, a kontextust. Ezért a korábbi statikus elméletek mellet különféle dinamikus szemantikai elméletek születtek (Kálmán–Rádai 2001), amelyek minimálisan a különböző formulák információállapot-változtató képességével egészítették ki a hagyományos logikai ábrázolást (pl. DPL, Dynamic Predicate Logic). A dinamikus elméletek másik ágát a különféle reprezentacionalista elméletek képviselik, amelyek közül a DRT a legismertebb.

A reprezentacionalista elméletek azt feltételezik, hogy a nyelv és a világ között egy harmadik szint is található, ahol a mondatok jelentését reprezentáljuk. Emellett érv a következő példapár:

(23) Valaki nem mosolygott. (Ő) Mérges volt.
(24) Nem mindenki mosolygott. *(Ő) Mérges volt.

(23) és (24) első mondata logikailag ekvivalens egymással, így egy predikátumlogikán alapuló rendszer nem tud számot adni arról, hogy ugyanaz a folytatás miért jó az első esetben, és rossz a másodikban. Megoldást jelent a problémára a nyelv és a világ(-modell) közé beépített diskurzusreprezentáció alkalmazása: a (23) példában a *valaki* bevezet egy szereplőt, akiről aztán tehetünk újabb állításokat (függetlenül attól, hogy talán többen is vannak azok, akik nem mosolyognak), a (24) példában viszont a negált első mondat nem vezet be „egy bizonyos" szereplőt, ezért nem is lehet rá visszautalni.

A DRT esetében ezek a reprezentációk egy rendezett párként írhatók fel, ahol az első komponens a bevezetett referensek listája (r: entitás, e: eseményszerűség), amelyekről a második komponensben állításokat tehetünk. Ezek az ún. DRS-ek (diskurzusreprezentációs struktúrák) egy könnyen átlátható „dobozos" formában is megjeleníthetők.

(25) Egy magyar fiú megköszöntött egy holland lányt.

r1 r2 e3
magyar (r1) fiú (r1) holland (r2) lány (r2) e3: megköszönt (r1, r2)

A DRS-ek rekurzívan beágyazhatók egymásba, így fejezhető ki például a tagadás, vagy a feltételes mondatok. Visszatérve az előző példapárra: a (23) első mondatához tartozó DRS: ⟨{x}, {személy x, ¬⟨∅, {mosolyog x}⟩⟩, vagyis bevezetünk az alap DRS-be egy x referenst, amire a következő mondat vissza tud utalni; a (24) első mondatához tartozó DRS azonban: ⟨∅, ¬ ⟨∅, ⟨⟨{x}, {személy x}⟩ ⇒ ⟨∅, {mosolyog x}⟩⟩⟩, vagyis nincsen a legkülső DRS-ben referens (az csak kétszeresen beágyazva jelenik meg), ezért hát nem is lehetséges a visszautalás.

A DRT-nek többféle kiterjesztése is született, amelyek segítségével számot tud adni az igeidőről, a prozódiai információról stb., azonban számos jelenség kezelésére még így sem képes. A tulajdonneveket például mindig ismertnek tételezi fel, amivel nem ad számot arról, hogy nem mindig tudjuk, kire referálnak. Nem tudja továbbá megmagyarázni, hogy miért lehetséges folytatása (26)-nak (27), míg (28) csak nagyon specifikus kontextusban lenne elfogadható.

(26) Jóska tegnap megnősült.
(27) ... A pap nagyon harsányan beszélt.
(28) ... *A kutya nagyon harsányan ugatott.

Sem a pap, sem a kutya nem kötelező eleme az esküvőnek, így csupán valamiféle világismereti tudás alapján lehetünk képesek a kettő között különbséget tenni, és ilyen típusú információ megragadására a DRT nem képes. Megoldást jelent a problémára, ha azt feltételezzük, hogy ezek a reprezentációk belső, mentális struktúrák, amelyek beszélőkhöz (hallgatókhoz) vannak rendelve; továbbá hogy mindenkihez tartozik egy óriási (mentális) DRS, amit a születése pillanatától épít, és amelyben így nyilvánvalóan minden világról való tudása szerepel. Egy ilyen rendszer az LDRT (Lifelong DRT, Alberti 2000).

Az LDRS-t formálisan egy olyan ⟨{R, P, W}, {ext, ref, prc, wrl, cur, mea, PHF}⟩ rendezett párként definiálhatjuk[24], amelyben az első komponens három megszámlálhatóan végtelen halmazból áll, amelyeket rendre a *referensek,* a *predikátumnevek* és a *világok* halmazaként nevezhetünk meg. A világok adják meg a normál DRS-ek referenseinek azt a hierarchiáját, amely abból fakad, hogy mely referens milyen mélyen beágyazott DRS-ben jelenik meg. A második komponens totális/parciális függvényekből és relációkból áll, amelyek *P*, *R* és *W* elemeit hozzák kapcsolatba egymással. Az LDRS-ek rekurzív definíciójának bázisát egy olyan $LDRS_0$ „üres LDRS" alkotja, amelynek második komponensében (majdnem) mindegyik függvény és reláció (még) üres. Maga a {P,R,W} hármas alaphalmaz nem változik a rekurziós lépések során, csupán relációk épülnek ki az elemeik között. Ez az $LDRS_0$ esetében úgy interpretálható, hogy az ideális hallgató a

[24] Ez az ismertetés Alberti–Kleiber (2001) alapján mutatja be az LDRT-t, a rendszer pontos matematikai definíciója, annak részletes magyarázata és sok példa Alberti (2000)-ben, vagy magyarul Alberti (2004)-ben olvasható.

születés pillanatában végtelen sok üres „fogassal” születik, amelyekre majd referenseket, predikátumokat és világokat akaszthat, melyeket különféle relációkkal össze is köthet.

Az *ext* egy (totális) függvényt jelöl, amely minden egyes P-beli predikátumhoz hozzárendeli annak extenzióját (rendezett n-esek halmazát, amikre a predikátum igaz). A *ref* parciális függvény másképpen teremt kapcsolatot referensek és predikátumok között: bizonyos referensekhez egy állításhalmazt – tulajdonképpen egy DRS-szerűséget – rendel, így válik lehetővé a nyelv önreferáló jellegének megragadása. A *prc* („precedencia”, azaz „megelőzés”) reláció egy szigorú részbenrendezés a világok körében egy kitüntetett *v* alapvilággal, amelyet nem előz meg más világ, a *wrl* parciális függvény predikátumokhoz és referensekhez rendel egy-egy világot, a *cur* kurzor-függvény pedig kitüntet egy világot a világok köréből, és kitüntet néhány referenst a referensek köréből (pl. az aktuális idő- és térreferenst, valamint az aktuális topik referensét). A *mea* jelentés-függvény predikátum-nevekhez rendel DRS-szerű állításhalmazokat (az aktuális tudásunkat reprezentálja az adott predikátum jelentését illetően), végül a PHF relációhalmaz a predikátumok olyan viszonyait írja le, amelyek a hozzájuk tartozó hangalakok azonosságán, képzési vagy összetételes viszonyán alapul.

A DRT alapú szemantikaelméletek tehát alkalmasabbnak látszanak a nyelv jelentéstani komponensének a leírására, mint a predikátumlogikán alapuló statikus elméletek. Kérdés azonban a szintaxissal való kompozicionalitásuk. Ha bevonjuk a λ-absztrakciót, a DRT is ugyanúgy alkalmas lesz arra, hogy a chomskyánus szintaxissal kompozicionális legyen, mint a Montague-féle szemantika, csak a deriváció végén egy nyelvközelibb reprezentációt nyerünk. A λ-absztrakcióról azonban könnyedén belátható, hogy nagyon erős eszköz, olyan egységek is létrehozhatók a segítségével, amelyek nem léteznek egy nyelvben sem, így egy λ-absztrakciót használó rendszer gyakorlatilag bármivel kompozicionálissá tehető. A DRT és a frázisstruktúrán alapuló nyelvtanok között pedig lényeges strukturális különbségek vannak, hiszen az utóbbi hierarchikus, míg az előbbi nem, így ténylegesen egy nagyon erős eszközre van szükség ahhoz, hogy a kettő közötti kompozicionalitást biztosítsuk. Ennek elkerülésére született meg a GASG, egy olyan szintaxiselmélet, amely izomorf a DRT-vel, de úgy, hogy nincs szükség „trükkökre”, hiszen ugyanaz a hierarchiamentesség jellemző mindkét rendszerre; így tehát egy szigorúbb értelemben vett kompozicionalitás tud megvalósulni szintaxis és szemantika között.

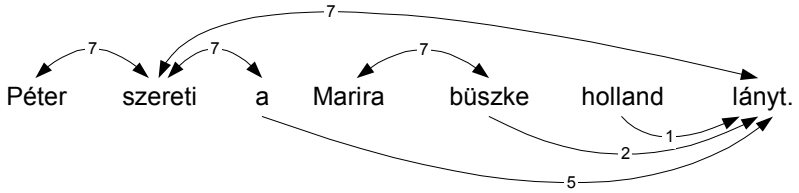
4.1.2 A modell

A GASG tehát egy egyszintű deklaratív nyelvtan, vagyis akkor tekint grammatikusnak egy mondatot, ha az őt felépítő lexikai egységekre vonatkozó minden megszorítás egyszerre kielégül. Totálisan lexikalista, vagyis minden információt a lexikonban tárol, ezért nincs is szükség frázisstruktúra szabályokra (és bonyolult szintaktikai reprezentációkra), a mondatok összeépüléséért pusztán az unifikáció felelős.

Mivel a nyelvtannak csak lexikai komponense van, egy-egy tétel leírása minden eddigi szótári leírásnál gazdagabb. A lexikai egységek jellemzése kéttípusú információból áll. Tartalmazza egyrészt az egység saját tulajdonságait (szófaja, morfológiai jegyei stb.), másrészt azokat a követelményeket, amelyeket azokkal az egységekkel szemben támaszt, amelyekkel relációba lép a mondatban (például egy ige hány és milyen vonzatot vár). Egy speciális követelmény az, hogy az adott elem milyen erősséggel akar ezekkel az egységekkel szomszédos lenni: ezek alapján kapjuk meg a mondat szórendjét.

A szórendről tehát szintaktikai fák építése nélkül is számot ad a modell, ún. *rangparaméterek* segítségével. A kiindulás az, hogy minden elem minden olyan elemmel szomszédos akar lenni, amelyikkel szintaktikai vagy szemantikai kapcsolatot létesít. Minden ilyen követelmény természetesen nem elégíthető ki, hiszen egy elemmel legfeljebb két másik

tud szomszédos lenni; ekkor van szükség a rangparaméterekre. Nyelvenként meghatározható, hogy mely egységek mely más egységekkel milyen erősséggel akarnak szomszédosak lenni, és az erősebb követelmény győz (a számok rangsort jelentenek, vagyis minél kisebb, annál erősebb). A 3. ábra érzékelteti mindezeket.



3. ábra: Rangparaméterek

A fenti példa a magyar nyelvre jellemző rangparamétereket mutatja. Az ige szomszédos akar lenni a vonzataival (alanya *Péter*, tárgyát két – egy determinánsi *a* és egy köznévi *lányt* – pilléren találja meg), de ez a követelmény viszonylag gyenge (hetes erősségű), ennél erősebb a névelő igénye, hogy a főneve mellett legyen (ötös erősségű), és még ennél is erősebb a különböző típusú melléknevek (különböző) igénye, hogy a főnevük mellé kerüljenek (egyes, illetve kettes erősségű). Még egy tényezőt kell figyelembe venni a helyes szórend meghatározásához: a magyar nyelvben az így a névelő és a főneve közé ékelődött melléknevek vihetik magukkal a bővítményeiket is (*a Marira büszke lány*), tehát a definíció a mondat szintjén *rekurzív*. Ez nem így van például az angolban, ahol bővítmény nélkül a melléknév beékelődhet a névelő és a főnév közé (*the proud girl*), de bővítményükkel együtt már nem, ebben az esetben a főnév mögé kényszerülnek (*the girl proud of Mary*). A modell tehát ezekkel a rangparaméterekkel nem csupán egy adott nyelven belül tud számot adni a szórendről, hanem a nyelvek közötti szórendi különbségek is elegánsan megragadhatók a segítségükkel.

A szórend rangparaméterekkel való kezelésének előnye, hogy homogén nyelvtant eredményez, vagyis minden követelmény egységesen kezelődik. Hiszen a szórendre vonatkozó megkötések is ugyanolyan követelmények, mint mondjuk az esetre vagy az egyeztetésre vonatkozók, egyes nyelvek inkább alkalmazzák, mint mások. A GASG-ben a szomszédossági követelmény tehát ugyanúgy egy tulajdonság, egy elvárás, amely a lexikonban az egység mellett rögzítve van, és amelynek teljesülnie kell.

A rangparamétereket nem csupán a szórend ellenőrzésére lehet alkalmazni, bármely tulajdonság vagy elvárás mellé rögzíthetünk rangokat, így egy optimalitáselmélethez hasonló megközelítést kapunk. Mondhatjuk például, hogy a *találkozik* ige a tárgyát két pilléren keresi, nem csupán főnévi, hanem determinánsi elemet is keres, vagyis nem jó a *Péter találkozott lánnyal* mondat. Viszont ez a követelmény gyengébb rangú, mert felülbírálhatja például a tárgy fókuszba helyezése: a *Péter LÁNNYAL találkozott* mondat grammatikus lesz. Ezzel az eszközzel egy minden eddiginél univerzálisabb szintaxiselmélet alkotható, ahol a nyelvek közötti különbségek rangparaméterek (számok) segítségével kifejezhetők.

A lexikai egységek gazdag belső tulajdonságai mellett egyetlen művelet található a nyelvtanban, az unifikáció, amely az egységek összeépülését vezérli. Csak azok az elemek tudnak kapcsolódni, amelyek jegyei kompatibilisek, egy szósor pedig akkor alkot grammatikus mondatot, ha az őt alkotó minden lexikai egység követelménye teljesül. Az így kapott szintaktikai reprezentáció a függőségi nyelvtanok reprezentációihoz hasonlít – azt tartalmazza, milyen grammatikai viszonyok létesültek az elemek között –, viszont számot ad a szórendről is, így a GASG nem tekinthető csupán egy dependencia grammatikának. A távoli függőségek problémája, amely sok elméletnek gondot okoz, – akárcsak az LTAG-ben – a GASG-ben sem létezik, mivel minden viszony lokális (és nem épül frázisstruktúra).

A modellben az unifikáció szerepe nem csupán annyi, hogy a jólformáltságot meg tudjuk állapítani. A lexikai egységek leírásában szemantikai információ is található, amely ugyanúgy tartalmaz változókat, mint a szintaktikai elvárások listája, az unifikáció tehát ezeken a formulákon is végrehajtódik, betöltődnek a változók, így előáll a mondat DRT alapú szemantikai reprezentációja.

A lexikai egységek részletes leírását *A magyar lány fut* mondaton mutatom be, amely négy szóból áll, melyek rendezett sorozata: <v1, v2, v3, v4>. Minden leírás négy komponensből áll: az első az ún. *sajáttest* (vagy *sajátszó*), vagyis az egység hangalakja; a második tartalmazza az egység tulajdonságait és elvárásait, vagyis hogy milyen jegyekkel rendelkező elemeket keres a mondatban; a harmadik komponens mutatja meg az egység jelentését; végül a negyedik kódolja a kapcsolatot a jelentés és az egység sajátteste között. Prolog-hagyomány szerint a kisbetű jelöli a konstansokat, a nagybetű pedig a változókat.

Λ1= <{v1=*a(z)*},
{ref.def(v1), n.common(V1.1), immprec(α=5, v1, V1.1), arg (CASE1, V1.1, V1.2)},
{Q1.1(X1), Q1.2.GRF1(X1)},
{corr(V1.1, Q1.1), corr(V1.2, Q1.2), corr(CASE1, GRF1)}>

Λ2=<{v2=*magyar*},
{adj(v2), n.common(V2), immprec(α=1, v2, V2)},
{Hungarian(X2), Q2(X2)},
{corr(v2, Hungarian), corr(V2, Q2)}>

Λ3=<{v3=*lány*},
{n.common(v3), 3.sg(v3), arg.nom(v3, V3)},
{girl(X3), Q3.subj(X3)},
{corr(v3, girl), corr(V3, Q3)}>

Λ4=<{v4=*fut*},
{fin.pres(v4), v.intr(v4), n(V4.11), 3.sg(V4.11), arg(nom, V4.11, v4), ref(V4.12),
immprec(α=5, V4.12, V4.11), immprec(α=7, V4.11, v4)},
{run(X4.1), Q4.1(X4.1)},
{corr(v4, run), corr(V4.11, Q4.1)}>

A második lexikai egység esetében például megtudjuk, hogy a *magyar* az egy melléknév (adj), ami keres egy főnevet, azon belül köznevet (n.common), és a melléknév α=1 rangban megelőzi a köznevet, vagyis a lehető legközelebb akar hozzá kerülni. Az egység jelentésre vonatkozó komponense azt mondja, hogy van egy X2 entitás, akiről tudjuk, hogy magyar, és még egy Q2 tulajdonságát is tudjuk, ami korrelál a keresett V2 köznévvel. A mondat összeépülésekor a V2 helyére a *lány* szó fog betöltődni, tehát ez az X2 entitás egy magyar lány lesz.

Az ige esetében sokkal több tulajdonság szerepel. Megtudjuk, hogy a *fut* finit, jelen idejű alak, intranzitív, vagyis nem keres tárgyat, egyetlen főnevet (n) keres (V4.11-et), amelynek egyes szám harmadik személyben és alanyesetben kell állnia (majd a *lány* szót fogja megtalálni), illetve keres még egy referenciális pillért (például egy névelőt, itt az *a*-t), ami ezt a főnevet α=5 rangban előzi meg. Az alanyesetű főnév pedig α=7 rangban előzi meg az igét. Végül megtudjuk, hogy van egy X4.1 entitás, aki fut, és még egy Q4.1 tulajdonság is igaz rá, ami korrelál azzal a főnévvel, ami V4.11-be töltődik be (ami a *lány* szó). Amikor az egész mondat összeépül, az is kiderül, hogy az X2 entitás azonos X4.1 továbbá X1 és X3 entitásokkal, hiszen a mondat egyetlen személyről tesz állítást.

Kezdetben a nyelvtan (és így a program) alapegységei (lexikai egységei) szavak voltak, külön szótári bemenet volt tehát például a *szeret* és a *szereti* (Alberti et al. 2002a-b), és a szavak rendszere valamiféle öröklődési hálózatban volt elképzelve. Később azonban egyértelművé vált, hogy a GASG teljes homogenitását az biztosítja, ha a morfológiát is „totálisan lexikalista" módon közelítjük meg (Alberti et al. 2005): nem a szavakhoz, hanem közvetlenül a morfémákhoz rendeljük a lexikai egységeket, amelyek oly módon „szerelik össze magukat" a mondatelemzés során, hogy az is eldől, mely elemek épülnek össze szóvá, és melyek alkotnak (egymás közelében maradó) külön szavakat (a morféma alapú GASG formális definíciója Alberti et al. (2003)-ban olvasható). Az univerzalitás szempontjából is szerencsés külön lexikai tételként felvenni minden állítást tevő elemet, hiszen míg a magyarban szuffixummal fejezzük ki például a műveltetést vagy a feltételes módot, addig az angolban külön szóval utalunk rájuk. (Morféma alapú szemantikát épít Gambäck (2005) is, amellyel a japán nyelv bizonyos jelenségeit tudja elegánsan megragadni.)

A GASG előnye tehát egyrészt az, hogy (akárcsak a legtöbb lexikalista elmélet) deklaratív és egyszintű, vagyis egyszerre életbe lépő megszorításokkal dolgozik, minden információt egy szinten kezel, így egyszerre képes látni a morfológia, a szintaxis vagy akár a szemantika szempontjából releváns jegyeket, ami lényegesen gyorsabbá és sikeresebbé teszi a mondat feldolgozását. A másik előnye pedig, hogy azáltal, hogy nem épít frázisstruktúrát, természetes módon izomorf egy modern dinamikus szemantikaelmélettel, a DRT-vel; illetve hogy a szemantikához való eljutás (ami az elemzés fő célja) – bonyolult szintaktikai reprezentációk (fák) építése nélkül – közvetlenül történik.

4.2 Az elemző

Az implementációt Prolog programnyelven készítettük, amely a számítógépes nyelvészek és mesterségesintelligencia-kutatók által gyakran használt magas szintű programnyelv (a működésére egy egyszerű példa az 1. sz. mellékletben olvasható). Azért is választottuk ezt a programnyelvet, mert a GASG által használt unifikációs eljárás eleve be van építve, így bizonyos szempontból úgy működik, mint a nyelv maga.

A program bemenete egy szósor, amelyről el kell dönteni, hogy grammatikus mondatot alkot-e, és amennyiben igen, a programnak ki kell írnia a releváns lexikai egységek listáját, az általuk alkotott szintaktikai viszonyokat és egy diskurzus-szemantikai reprezentációt, vagyis a mondat jelentését. A következő pontokban részletesen ismertetem a különböző nyelvi szintek implementálásának kérdéseit, és egy egyszerű példán keresztül bemutatom, hogyan néz ki a program által használt lexikon, milyen morfológiai, szintaktikai és szemantikai reprezentációt társít a mondatokhoz, és hogyan valósítja meg a fordítást. Az utolsó pontban pedig további példák segítségével megmutatom, milyen nyelvi jelenségek kezelésére képes.

4.2.1 A lexikon

A programhoz tartozó lexikon a Prologban íródott elemző-kód része, amely az adatbázis viszonylag kis mérete miatt (kb. 200 lexikai egység) egyelőre nem okoz problémát, de ha jelentősebb mértékben megnövelnénk a lexikai egységeink számát, külső adatbázisra lenne szükség. Technikailag úgy oldottuk meg a lexikai egységek négy komponensének a tárolását, hogy az egységhez tartozó sajátszó és az elem saját tulajdonságai a program adatbázis részében találhatók, az elvárásai és a szemantikájára vonatkozó információk pedig a program fő részét képezik. A Prolog jellege miatt nem lehetett minden információt egy helyen, az adatbázis részben tárolni, mert a programnyelv úgy működik, hogy az adatbázis részben csak konstansokat engedélyez, az elvárások és a szemantikai leírás pedig alapvetően

változókat tartalmaz. Azonban a lexikai egységek „kettévágása” semmilyen elméleti következménnyel nem jár, hiszen a program fő részében is csupán az elvárások felsorolása zajlik, a beépített unifikáción kívül semmilyen egyéb műveletet nem végez a program.

A program adatbázis részében a lexikai egységek felsorolása zajlik, amelyek (itt) két részből tevődnek össze. Az első rész az esetlegesen változókat is tartalmazó sajáttest[25], amely azt mutatja, milyen formában jelenhet meg az adott morféma. Például a *bokor* tő (29) sajátteste *bokOr*, ahol a nagybetű változót jelent, azaz egy olyan egységet, amely megvalósulása függ a környezettől, ebben a konkrét esetben lehet *o* vagy üres. Egy másik példa a tárgyeset ragja (30), amelynek a sajátteste *Vt*, ahol a *V* megvalósulása lehet *a*, *e*, *o*, *ö* vagy üres.

```
(29)    lexi(m("bok","O","r"), labstem("bush",  phonfst(2,1,2,2),1,[])).
(30)    lexi(m("V","t",""),    labsuff("ACC",   phonfsu(1,1,1,3), 1, 4)).
(31)    lexi(m("","vár",""),   labstem("wait",  phonfst(2,2,2,2),2,
                                        [["NOM","ACC"],["NOM","SUBLATIV"]])).
(32)    lexi(m("","magyar",""),labstem("Hungarian", phonfst(2,2,1,1), 4,[])).
(33)    lexi(m("","téged",""), labstem("yousg", phonfst(1,2,2,0),1,[["ACC"]])).
```

A lexikai egységek leírásának második része az úgynevezett címke (label), amely az egység tulajdonságait tartalmazza. A címke háromféle lehet: *labstem* tövek esetében, *labder* képzők esetében, és *labsuff* inflexiós toldalékok esetében, mivel e három morfématípusnak különböző releváns tulajdonságaik lehetnek.

A címke mindhárom esetben először a morféma „jelentését” tartalmazza egy közvetítő nyelven (amely jelenleg az angol, de bármely más nyelv megfelelne), majd a fonológiai, morfológiai, szintaktikai, szemantikai és esetleg egyéb tulajdonságait.

Toldalékoknál fonológiai tulajdonság lehet például, hogy a különféle, adott nyelvre jellemző, (írásban jelölt) fonológiai váltakozásokat kiváltja-e az adott toldalék. 1-es jelöli, ha igen, 2-es, ha nem, és 3-as, ha nem releváns. A magyar esetében például kérdés, hogy az illető toldalék (az arra érzékeny tövek esetében) nyújt-e (*kutya-kutyát*) rövidít-e (*kéz-kezek*) kiváltja-e a hangkivetést (*bokor-bokrom*) illetve nyit-e (*gázok-gázokat*). (30)-ban láthatjuk, hogy a tárgyeset fonológiai tulajdonságai (*phonfsu*, vagyis phonological features of a suffix): 1,1,1,3, vagyis nyújt, rövidít és kivet, a nyitás viszont nem releváns, hiszen rag, vagyis utána már nem következhet toldalék, aminek a magánhangzóját nyíltabbá tehetné.

Töveknél egészen más tulajdonságok lesznek fonológiailag relevánsak. Köznevek esetében a négy szám azt mutatja, hogy elöl képzett (magas hangrendű)-e a szó, kerek-e, a releváns magánhangzója mennyire nyílt, illetve, hogy nyitótő-e (pl. a *gáz* nem nyitótő, kötőhangként *-o* jelenik meg: *gázok*; *ház* nyitótő, kötőhangként *-a* jelenik meg: *házak*). Ezek az információk szükségesek ahhoz, hogy megfelelően tudjuk toldalékolni az adott szót. (29) esetében például láthatjuk, hogy a *bokor* szó mély toldalékokat vesz fel, utolsó magánhangzója kerek (bár ez mély magánhangzók esetében fonológiailag nem releváns, hiszen nincsen réses „párjuk”), és nem nyitótő (*bokrok* és nem **bokrak*). Főnévi névmásoknál megint mást jelentenek a számok: az adott névmás számát, személyét és határozottságát, illetve a 0 azt, hogy az adott lexikai egység egy névmás. (33)-ban láthatjuk, hogy a *téged* névmás egyes szám második személyű, és határozatlan ragozást vált ki.

Morfológiai tulajdonság például a szófaj, amit szintén számok jelölnek. 1-es a főnév (29) vagy főnévhez járuló toldalék (30), illetve főnévi névmás (33), 2-es az ige (31), vagy igéhez járuló toldalék, 3-as a melléknév (32) stb.

---

[25] A változó technikailag egy konstans (idézőjelben szerepel), hiszen mint említettem, ebben a részben változók nem szerepelhetnek. Funkcióját tekintve változó csak (ezt jelzi a nagybetű), és a program fő részében kerül feloldásra.

Végül szintaktikai tulajdonság például a lehetséges vonzatkeret(ek), ami csak a töveknél releváns. A *bokor* tőnek nem lehet vonzata, a lehetséges vonzatkeretek listája üres (ez nem minden főnévre igaz, például a *könyv* főnévnek gyakran van *-ról/-ről* ragos vonzata). A *vár* igének két különböző vonzatkerete is lehetséges. Az egyik [NOM, ACC], vagyis egy alany és egy tárgy (*Péter várja Marit*), a másik [NOM, SUBLAT], vagyis egy alany és egy *-ra/-re* ragos vonzat (*Péter vár Marira*). A *magyar* melléknévnek szintén nincs vonzata (vs. *büszke vkire*). Végül névmások esetében a jegy megint mást jelent, nem azt, hogy neki milyen vonzata lehet, hanem azt, hogy ő milyen esetben áll. Toldalékoknál az utolsó jegy vonzatkeretek listája helyett egy szám, ami azt mutatja, milyen erősséggel akar az illető toldalék a tővel szomszédos lenni. Mivel (30)-ban egy ragot látunk, ez a rangparaméter négyes, ami gyenge követelmény, hiszen a rag kerül mindig a szó végére.

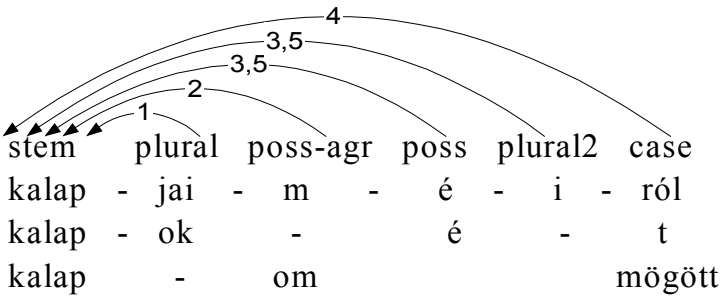
4.2.2 Morfológia

A GASG implementálását szószinten kezdtük (Alberti et al. 2002a-b), de mind elméleti, mind gyakorlati okokból hamarosan áttértünk morfémaszintre (Alberti et al. 2005). Az elméleti okokat korábban már kifejtettem: a nyelvtan teljes homogenitását és a még nagyobb fokú univerzalitást az biztosítja, ha minden morféma külön lexikai egység. Gyakorlati szempontból pedig azért hasznos, ha nem komplett szóalakokat, hanem morfémákat tárolunk, mert a magyarhoz hasonló ragozó nyelvek esetében óriási adatbázisra lenne szükség, ha a szavak összes lehetséges alakját tárolni akarnánk; „olcsóbb” tehát morfémákat tárolni, és morfológiai elemzést végezni.

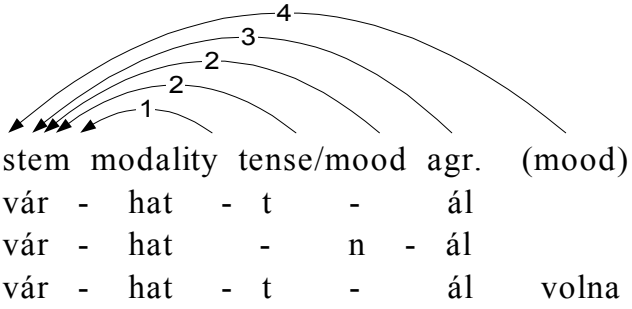
Kaptunk már olyan kritikát, hogy miért nem használunk már meglévő morfológiai elemzőt erre a feladatra, amikor nagyon hatékony rendszerek állnak már rendelkezésre a magyar nyelvre is. Ha csak a hatékonyság növelése (az adatbázis méretének csökkentése) miatt tárolnánk morfémákat szavak helyett, akkor talán jó választás lenne, viszont a fő érvünk minderre mégiscsak elméleti. Egy létező elemző használatával nem tudnánk a lexikai egységek ennyire gazdag leírását adni, továbbá nem tudnánk biztosítani a morfológiai alapú egyszintűséget (vagyis azt, hogy például a *-tat* képző morfológiai, szintaktikai és szemantikai jegyei egyszerre legyenek figyelembe véve). Mindenek előtt pedig fő célunk a totális lexikalizmus elvének gyakorlatban való kipróbálása, és ennek része a totálisan lexikalista morfológia tesztelése is, ami előbbre való, mint az elemzőnk hatékonyságának esetleges növelése egy bizonyítottan jól működő morfológiai elemző integrálásával.

Az elemzés első lépése a szegmentálás, vagyis a szavak morfémákra bontása a szótárban szereplő lexikai egységek alapján. Ezután a morfofonológiai ellenőrzés következik, annak megállapítása, hogy a mondat jól formált szavakból áll-e. Ez több tényezőtől függ, például hogy a szót alkotó morfémák helyes sorrendben vannak-e, és hogy a megfelelő allomorfok szerepelnek-e bennük (pl. *ház-ak* vs. **ház-ok*, vagy *bokr-ot* vs. **bokor-t*).

A morfémasorrendet rangparaméterek segítségével ellenőrzi a program. Az alapvető különbség a mondatszintű és a szószintű rangparaméterek között, hogy míg az előbbi rekurzív (4.1.2 pont, 3. ábra: Rangparaméterek), addig egy szón belül nem lehetséges, hogy a beékelődő elemek további elemeket „visznek magukkal”. A magyar főnévi morfológiára a 4. ábraán látható rangparamétereket alkalmazzuk, az igék rangparamétereit pedig az 5. ábra mutatja (a szuffixumok címkéjének utolsó eleme kódolja ezt az információt).



4. ábra: A főnévi (szón belüli) rangparaméterek



5. ábra: Az igei (szón belüli) rangparaméterek

Bármely elem után csak olyan elem állhat, amelynek szigorúan nagyobb (azaz gyengébb) a rangparamétere, így például egy igei tő után nem szerepelhet egyszerre időjel és módjel, mert mindkettőnek 2-es a rangparamétere (ennek a jelentésnek a kifejezésére alkalmazzuk a *volna* feltételes mód jelző szócskát, 4-es ranggal, vagyis a szóalak végén). Ez alól az egyetlen kivétel a főnevek esetében a birtoklást jelző *-é* és a birtokjel után előforduló *-i* többesszámjel, melyek állhatnak együtt is (és esetlegesen több is lehet belőlük). Az ilyen – ismételhető – rangok jelzésére szolgál a törtszám mint rangparaméter (itt 3,5).

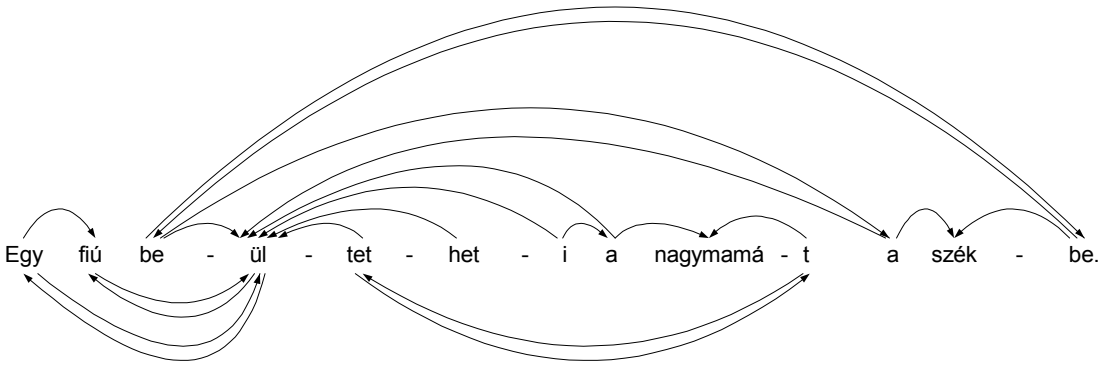
A morfofonológiai ellenőrzés másik komponense annak vizsgálata, hogy a szavakban a megfelelő allomorfok szerepelnek-e, vagyis hogy a változók megfelelő értékei töltődtek-e be az unifikáció során. A toldalékok keresik a tövet (és esetleg más morfémákat is), de a tő nem keres toldalékot (egyirányú a viszony). A változók unifikációja érdekében a keresés történhet előrefelé és visszafelé.

A tárgyeset ragja például olyan követelményeket tartalmaz, mint hogy bizonyos tövek utolsó magánhangzója hosszú legyen (nyúlás), például *kutya–kutyát*; más tövek utolsó magánhangzója rövid legyen (rövidülés), például *nyár–nyarat*; illetve bizonyos tövek utolsó magánhangzója eltűnjön (kiesés), például *bokor–bokrot*. Akkor sikeres az unifikáció, ha ezek az elvárások teljesülnek, vagyis az elemzett szó sajáttestében lévő változó aktuális értéke a fent említett követelményeknek megfelel.

Másik irányú keresést igényel, amikor a toldalék sajáttestében szerepel változó. Ilyen az elölségi harmónia (*ház-ban* vs. *szék-ben*), a kerekségi harmónia (*könyv-höz* vs. *eper-hez*), illetve a kötőhang megléte vagy hiánya (*folyó-t* vs. *ház-at*). Az unifikáció során a változó megfelelő értéke betöltődik, és akkor lesz a szóalak jól formált, ha az így nyert allomorf megegyezik a beírt változattal. Az allatív (*-hoz/-hez/-höz*) rag esetében például a sajáttest *h3z*, ahol a *3*-sal jelölt változó lehetséges értékei: *e*, *o* és *ö*; ha (viselkedés szempontjából) mély hangrendű tövet talál (*phonfst* első paramétere 2-es), betöltődik a [hátsó] tulajdonság, így a kötőhang *o* lesz, ha magas hangrendűt (*phonfst* első paramétere 1-es), akkor a kerekséget is ellenőrzi: ha a tő utolsó magánhangzója kerek (*phonfst* második paramétere

1-es), akkor a *V* értéke is [kerek] magánhangzó lesz, vagyis *ö*, ha pedig réses a tő (*phonfst* második paramétere 1-es), akkor kapjuk az *e* kötőhangot.

Mindezek ellenőrzésére természetesen bármilyen más morfológiai elemző rendszer is képes, és valószínűleg nagyobb hatékonysággal. A totálisan lexikalista morfológia előnye, hogy a különféle toldalékoknak lehetőségük van a szón *kívül* található morfémák keresésére is, tehát bármely másik morfémával grammatikai viszonyt tudnak létesíteni, függetlenül attól, hol található (ezért nem okoz gondot az elméletnek, hogy különböző nyelvekben különböző helyen lehet a szószint). A 6. ábra ezt érzékelteti.



6. ábra: Szón belüli és szón kívüli keresés

Az itt produktív *be* igekötő[26] keresi az illatív ragot (a ragnak persze kell majd egy tő), a műveltetés képzője keresi a tárgyragot (hiszen az *ül* ige intranzitív), és az *-i* egyeztető morféma keresi a tárgy határozottságát mutató morfémát (névelőt), amely őt (a tárgyas ragozást) legitimálni tudja. Illetve a „másik oldalról" az illatív vonzat is keresi az igekötőt mint régensét, és a tárgyrag is keres egy olyan elemet, amely őt behozhatja a mondatba, és ezt a műveltetés képzőjében tudja megtalálni.

Az elemző működését a *Péter szereti az okos magyar lányt* mondaton mutatom be. Ha a beírt szósor grammatikus, a program kiírja (a végén), hogy `yes`, és több reprezentációt is ad, melyek közül az első a lexikai egységek listája. A kérdés (a Prolog-terminust használva a *goal*) (34)-ben olvasható, míg az eredményt (először a morfológiát) (35) mutatja (minden a dolgozatban bemutatott programfutás megtalálható a CD-mellékleten).

```
(34)   gramm("Péter szereti az okos magyar lányt.").

(35)   LEXIKAI EGYSÉGEK:

       Péter: n(1,1,li(m("","Péter",""),labstem("Peter",phonfst(1,2,0,2),1,[])))

       szeret: n(2,1,li(m("","szeret",""),labstem("love",
                                          phonfst(1,2,2,2),2,[["NOM","ACC"]])))
       i: n(2,2,li(m("","i",""),labsuff("sg3obj+def",phonfsu(1,3,1,3),2,3)))

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

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

       magyar: n(5,1,li(m("","magyar",""),labstem("Hungarian",
                                          phonfst(2,2,1,1),4,[])))

       lány: n(6,1,li(m("","lány",""),labstem("girl",phonfst(2,2,3,2),1,[])))
       t: n(6,2,li(m("V","t",""),labsuff("ACC",phonfsu(1,1,1,3),1,4)))
```

[26] A *be* igekötő nem produktív használata esetén (pl. befejez) nem jelenik meg -ba/-be ragos vonzat; viszont gyakorlatilag bármely mozgást (esetleg helyzetet) jelentő igével alkothat egy egységet (produktív használat), és ekkor általában hoz magával egy illatív esetben álló bővítményt (bemegy, besétál, berobog, beül vmibe).

(35)-ben látható, melyik allomorfhoz melyik lexikai egységet találta meg a program. Először az adott egység sorszáma olvasható (hányadik szó hányadik morfémája), majd a sajátteste (változóval, ha több allomorfja van), végül a címkéje, amely a „jelentését” és a különböző tulajdonságait tartalmazza (róluk az előző pontban már részletesen szóltam). Következő lépésként a program a szintaktikai viszonyokat listázza ki, de ezt már a következő pontban fogom bemutatni.

4.2.3 Szintaxis

A szintaktikai ellenőrzés alapvetően lokális követelmények kielégítéséből áll, amelyek azt kódolják, hogy egy elem hány és milyen tulajdonságú elem jelenlétét követeli meg. Csak akkor lehet grammatikus a mondat, ha az őt alkotó mindegyik lexikai egység minden elvárása teljesül. Ezzel biztosítjuk, hogy a régensek minden vonzathelye kitöltődjön (teljesség), és hogy azok a megfelelő tulajdonságokkal (pl. eset, egyeztetés) rendelkezzenek (konzisztencia). Szükség van azonban némi „globális” ellenőrzésre is, amikor mintegy „felülről ránézünk” a szintaktikai viszonyok hálózatára, hogy a koherenciát ellenőrizzük. Semmi nem szűrné ki ugyanis azt, amikor például két alany van a mondatban: a *Péter Mari ül* szósort grammatikusnak ítélné a program, mert mindkét alany talál magának igét, az ige talál magának alanyt, tehát minden lokális követelmény teljesül.

Implementációs szinten bevezettünk ún. „meta” lexikai egységeket, amelyek összefogják az azonos elvárásokat tartalmazó elemeket. Nem soroljuk fel például külön a *szeret* és az *utál* szintaktikai és szemantikai elvárásait, hiszen azok csak a sajátszóban és a DRS-ben megjelenítendő predikátumban térnek el egymástól. Ez technikai egyszerűsítésnek tűnik, de elméletileg is vállalható. Hiszen így gyakorlatilag egy öröklődési hálózatszerű rendszert kapunk, ahol a speciális eseteket előbbre rangsoroljuk (ezt a Prologban úgy érjük el, hogy egyszerűen előbb szerepeltetjük a programkódban), az általánosabbakat pedig hátrébb. A programnyelv tartalmaz egy speciális szimbólumot, a ’!’-et (cut), amely azt biztosítja, hogy ha a program megtalálja a változók egy lehetséges illesztését, akkor nem keres továbbiakat. Ezzel elérhető, hogy ha egy lexikai egységet illeszteni tudott egy speciálisabb bemenetre, akkor az általánosabbal már ne próbálkozzon.

Másrészt pedig a GASG valahol egy kategoriális nyelvtan (kiindulópontja az UCG), és ezzel a módszerrel egy lexikai egységnek lehet egy nagyon alaposan kidolgozott kategóriája, amely csak rá jellemző, de lehet valami általánosabb is, így több lexikai egység osztozna ugyanazon a leíráson; ezt valósítja meg a rendszerünk a gyakorlatban. A lexikai egységeket bonyolult hierarchiába lehet rendezni az alapján, hogy milyen jellemzőik (saját tulajdonságaik, illetve elvárásaik) vannak. Például ha a vonzatkeretet nézzük, elvileg nagyon sok „kategóriacímke” lehetséges lenne: a magyar nyelvben, ha van mondjuk 20 eset, akkor elvileg 2^{20} lehetséges vonzatkeretet kaphatnánk. Ennek azonban csak töredéke fordul elő ténylegesen a nyelvben, hiszen a legtöbb ige mellett háromnál több vonzat nemigen szerepel (ötnél több meg már egyáltalán nincs), az egyik vonzat majdnem minden igénél nominatívuszban van (alany), stb. A lexikon egy nagyon magasan hierarchizált rendszer, lehetne minden eleme teljesen különböző, de mégsem ez a helyzet. A totális lexikalizmus nem tesz előre megszorításokat, megengedhetné a teljesen különböző lehetőségeket is (2^k variáció, ahol k a releváns jegyek száma), de megfogalmazhatók benne általánosítások is (például hogy milyen vonzatkeretek lehetségesek). A kategoriális nyelvtanoknál az egyik cél éppen a kategóriák rendszerének optimális megadása, az univerzális állítások és a nyelvspecifikus jellemzők megtalálása, és a nyelvek közötti különbségek megragadása.

A lexikai egységek elvárásai tehát a program fő részében találhatók, ott mondjuk meg, hogy az adott elem (morféma) hány és milyen tulajdonságokkal rendelkező másik elemet (morfémát) keres a mondatban. A keresés a régens–vonzat viszonyok esetében kétirányú,

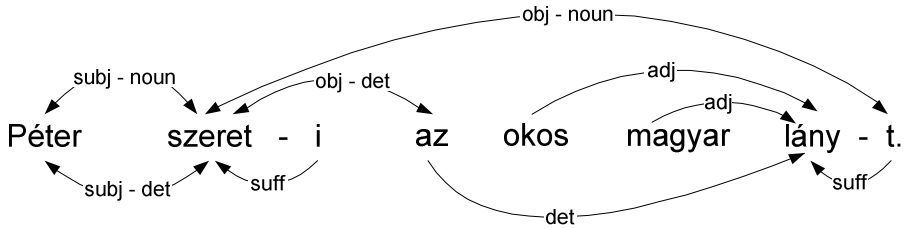
vagyis a régens is keresi a vonzatait – mégpedig két pilléren, egy főnévi (regent–noun–subj/obj/obl) és egy determinánsi (regent–det–subj/obj/obl) pilléren –; és a vonzatok is keresik azt az elemet, amelyik „behozhatja” őket a mondatba (noun–regent–subj/obj/obl, illetve det–regent–_, ahol az ’_’ jelöli Prolog-hagyomány szerint a „bármit”, mert (legalábbis a magyarban) a determináns nem tudja, milyen viszonyra keresi a régenst). A szabad bővítmények keresése egyirányú, egy főnévnek nem szükséges őt bővítő melléknév, de egy melléknévnek mindenképpen kell egy főnév, amelyet módosít (adj–noun–free viszony). Ugyanígy a tövek nem keresnek toldalékot, de a toldalékok keresnek tövet, tehát ez is egyirányú (szabad) viszony (suff–stem–free). Végül pedig a determináns kötődése a főnévhez szintén egyirányú (det–noun–free), hiszen a főnév előfordulhat a mondatban névelő nélkül is, de ez fordítva nem lehetséges. A globális ellenőrzést úgy végzi a program, hogy minden kétirányú viszonyhoz megkeresi a „párját”: amikor ugyanaz a viszony létesül ugyanazok között a lexikai egységek között, pusztán a sorrend más (pl. noun–regent–subj:11–21 viszony „párja”: regent–noun–subj:21–11). Mindezeket az előző pontban bevezetett (*Péter szereti az okos magyar lányt*) példán keresztül mutatom be. A mondathoz tartozó szintaktikai (függőségi) viszonyrendszer (36)-ban olvasható.

(36) SZINTAXIS:

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

regent-noun-subj: szereti-Péter
regent-det-subj: szereti-Péter
regent-noun-obj: szereti-lányt
regent-det-obj: szereti-az
det-noun: az-lányt
adj-noun: okos-lányt
adj-noun: magyar-lányt
```

A szintaktikai elemzés kimenetét két formában is olvashatjuk: egy részletesebb (így nehezebben áttekinthető) formában, ahol a szintaktikai viszony megnevezése után az őt alkotó lexikai egységek sorszámai találhatók (hányadik szó hányadik morfémája); és egy áttekinthetőbb, egyszerűsített formában, ahol a viszonyokat létesítő morfémák helyett az őket tartalmazó szavak vannak megjelölve. Ez utóbbin látható, hogy a *szereti* ige alanya (subj) *Péter* (kétszer), tárgya (obj) egyrészt a *lányt* (főnévi pillér), másrészt az *az* névelő (determinánsi pillér), mivel minden vonzatot két pilléren kell megtalálni, egy főnévi (noun) és egy determinánsi (det) pilléren. Ez tulajdonnevek esetében leggyakrabban (itt is) egybeesik (ezért szerepel a *Péter*-t tartalmazó sor kétszer), köznevek esetében általában egy névelő a determinánsi pillér, itt az *az*. Látható továbbá, hogy a névelő és a két melléknév is megtalálta a köznevet, amihez kapcsolódhat (det–noun, illetve adj–noun viszony). A morfémák közötti részletesebb viszonyhálózat könnyebben áttekinthető egy ábra segítségével (7. ábra).



7. ábra: Egy szintaktikai viszonyrendszer

A szintaktikai elemzés fontos lépése a szórendre vonatkozó megszorítások ellenőrzése, ami rangparaméterek segítségével történik. A programban *immprec*-nek (immediate precedence) neveztük el azt a követelményt, amely szószinten rekurzív, vagyis ha egy szó jogosan beékelődik két másik közé (tehát valamelyikhez erősebb rangban kapcsolódik, mint a másik), akkor vihet magával további elemeket is. Így az *immprec* definíciója (forrás: Alberti et al. 2002b) viszonylag bonyolult:

> **Definíció**: Egy *immprec*$_n$ *(n-ed rangú közvetlen megelőzési)* viszonyra vonatkozó követelmény *w1* és *w2* szavak között akkor tekintendő kielégítettnek, ha
>
> (1) *w1* a vizsgált szósorban (valóban) közvetlenül megelőzi a *w2* szót, vagy
>
> (2) van egy *w3* szó (*w1* és *w2* között)[27], amelyik kielégít egy immprec$_k$ viszonyra vonatkozó követelményt a *w1* vagy a *w2* szóval, ahol $n \geq k$, és a [w1,...,w3], valamint a [w3,...,w2] szósorok *legitimek.*
>
> Egy [x1,...x2] szósor akkor *legitim,* ha
>
> (1') csupán két tagból áll, vagy
>
> (2') akad egy olyan *x3* szó az [x1,...x2] sorozatban, amelyre igaz az, hogy valamilyen m szám mellett az *x3* szó és az *x1*, *x2* szavak valamelyike között van egy immprec$_m$ közvetlen megelőzési követelmény, továbbá az is teljesül, hogy az [x1,...x3] és az [x3,...x2] szósorok legitimek.[28]

A definíció alapján grammatikus például (37) (a program kiírja az elemzést, és azt, hogy `yes`); nem grammatikus viszont (38) (a program a `no` választ adja), mert az *angol* és a *fiú* szavak közé nem ékelődhetett be az *okos*, hiszen a nemzetiségnév erősebb (1-es) ranggal akar a főnévhez kapcsolódni, mint az egyéb melléknév (amely csak 2-es ranggal). Az angol nyelvben kissé máshogy működik az *immprec*, ott a jogosan beékelődött elemek nem vihetnek magukkal további elemeket: míg a magyarban grammatikus (39), addig az angolban nem: ha (40)-et tovább bővítjük, nem (41)-et, hanem (42)-t kapjuk.

(37) Az okos angol fiú szereti Marit.
(38) *Az angol okos fiú szereti Marit.
(39) A Marira büszke holland lány szereti Pétert.
(40) The proud Dutch girl loves Peter.

---

[27] A zárójeles változatot a svájci németben és a hollandban megfigyelhető keresztező-függőségi (infinitívuszi) konstrukciók esetében kell alkalmazni. Nem véletlen, hogy a különleges, a világ nyelvei körében talán szélsőségesnek tekinthető, nem-környezetfüggetlen apparátust igénylő keresztező függőség kezelése a GASG-ben is speciális közvetlen megelőzési definíciót kíván, ami megengedi azt a furcsaságot, hogy egymás szomszédságába vágyódó *w1* és *wi2* szavak közé olyan *w* szó ékelődjön, amelyet nem egy *w1* és *w2* közé legitim módon bekerülő *w3* szó „hoz magával", hanem egy *w1*-hez vagy *w2*-höz kötődő, de a [w1,…,w2] szózónán kívül eső *w3* szó (a *w* szó tehát „kívülről legitimálódik").

[28] Az *immprec viszony* és a *legitim szósor* definíciója egymásra hivatkozik, de ez nem okoz káros „körkörösséget", mert ún. *szimultán rekurziós* definiálási technikát alkalmaztunk, amit lényegében az immprec viszony rangparaméterezése tesz lehetővé.

(41) *The proud of Mary Dutch girl loves Peter.
(42) The Dutch girl proud of Mary loves Peter.

A szintaktikai ellenőrzés tehát több, mint amit egy függőségi nyelvtan végezne, hiszen az elemzés végére nem csupán a grammatikai viszonyok töltődnek be, hanem a szórend is ellenőrződik. A rangparaméteres megoldás elegánsan ad számot az egy nyelven belüli szórendi viszonyokról, és a nyelvek közötti szórendi különbségekről is. A következő reprezentációs szint a szemantika, amelyről a következő pont szól.

4.2.4 Szemantika

A program utolsó kimenete a szemantikai reprezentáció (Alberti–Kleiber 2003), amelyet az LDRT-ben (4.1.1) valósítottunk meg. Alapvető különbség a DRT-beli ábrázoláshoz képest, hogy a referenseket világokba ágyazzuk (megmondjuk róluk, hogy már korábban bevezetett entitásra utalnak, vagy most kerülnek bevezetésre), illetve hogy kitüntetünk egy fixpontot, amely arra a lexikai egységre mutat, amely az időjelet viseli. Ezzel tudunk különbséget tenni például az *Egy szép lány énekel* és az *Egy éneklő lány szép* mondatok között: az első esetében az ige, a második esetében a *szép* melléknév a mondat fixpontja.

A szemantikai reprezentációt kompozicionálisan állítja elő a program: mindegyik állítást tevő lexikai egységhez tartozik egy ún. proto-DRS, vagyis egy (vagy több) kondíciósor, amellyel az adott elem hozzájárul a végső DRS-hez. Ezek a sorok a lexikai egység leírásánál természetesen még változókat tartalmaznak az argumentumok helyén, melyek az unifikáció során töltődnek be.

Az ábrázolás lényege tehát, hogy a mondatok különféle referenseket vezetnek be (*provref*), amelyekről állításokat tesznek (*pred*). A referensek egyik típusa a személyekhez, dolgokhoz kötődő referensek, ezeket *r*-rel jelöljük, egy másik típusa az eseményekhez kötődő (eventuális) referensek, ezeket *e*-vel jelöljük. Ez utóbbiakra azért van szükség, mert eseményekről is tehetünk állításokat, például *Péter szereti Marit, és* ez *bosszantja Julit*. (43)-ban olvasható a *Péter szereti az okos magyar lányt* mondathoz rendelt szemantikai reprezentáció (a mondatot alkotó lexikai egységek listáját és a grammatikai viszonyok rendszerét a korábbi pontokban már bemutattam).

(43) SZEMANTIKA:

```
provref("fixpoint",[e(2,1,1)])
provref("old",[r(1,1,1)])
pred("Peter",1,[r(1,1,1)])
provref("new",[e(2,1,1)])
pred("love",2,[e(2,1,1),r(1,1,1),r(3,1,1)])
provref("old",[r(3,1,1)])
provref("<or=",[r(3,1,1),e(2,1,1)])
pred("clever",4,[r(3,1,1)])
pred("Hungarian",5,[r(3,1,1)])
pred("girl",6,[r(3,1,1)])
```

r1 r3 e21
Péter (r1)
okos (r3)
magyar (r3)
lány (r3)
e21: szeret (r1, r3)
…

A program által kiírt reprezentáció könnyebb áttekinthetősége kedvéért elhelyeztem egy DRS-ben a fontosabb állításokat, azaz, hogy van két szereplőnk: r1 és r3 (a számok arra utalnak, hogy a mondatban hányadik szó a determinánsi pillérük), ahol r1 Péter, r3 pedig a lány. Van továbbá egy eseményünk, e21 (a számok arra utalnak, hogy hányadik szó hányadik morfémája a predikátum), ahol e21 az, hogy r1 (Péter) szereti r3-at (a lányt). A „rajzolt” DRS-ben nem szerepelnek a mondat nagyobb szövegbe (diskurzusba) való ágyazásához szükséges predikátumok (fixpont, rendezési relációk), az eredeti (program által kiírt) reprezentációban azonban olvashatók. E szerint a fixpont az igére mutat, *Péter* egy már bevezetett („old”) referensre utal vissza (hiszen tulajdonnév), az eventuális referens

korábban még nem szerepelt („új” referens), a *lány* referense is ismert entitás („old”), mert határozott névelővel utalunk rá, végül pedig megtudjuk, hogy a *lány* referense egy korábbi világban „született”, mint az ige által bevezetett eventuális referens (vagy esetleg ugyanabban).

A program tehát képes a grammatikus mondatokhoz egy dinamikus diskurzus-szemantikai reprezentációt társítani, egy olyan angol nyelvű kimenetet (DRS-t) ad, amely tekinthető az angol nyelv egy megszorított, szigorúan formalizált változatának. Bárki, aki ért angolul, alig egy óra alatt elsajátíthatja a DRS-ek olvasásának képességét, míg megtanulni neki magyarul éveket venne igénybe. Ezért a programunk – a szemantikai komponense miatt – képes megvalósítani a *géppel segített fordítást*. Továbbá a használt szemantikai keret kellő mértékben univerzálisnak tekinthető ahhoz, hogy segítségével a *gépi fordítást* is meg lehessen valósítani (Alberti–Kleiber 2004). Ennek a jellemzőit mutatom be a következő pontban.

4.2.5 Gépi fordítás

A totálisan lexikalista megközelítés segítségével más nyelvek nyelvtanai is egységes keretben kezelhetők. A magyar mellett az angol nyelv néhány lexikai egységét is felvettük az adatbázisba annak igazolására, hogy nincsen szükség más mechanizmusokra egy egészen eltérő nyelv kezeléséhez sem, csupán kissé mások lesznek az egyes lexikai egységek követelményei (például a rangparaméterek), vagy máshol lesz a szószint. (44) a korábban már elemzett mondat angol nyelvű változatának a kimenetét mutatja, amely sok ponton hasonlít a magyar mondat elemzésére, a két szemantikai reprezentáció pedig teljesen megegyezik (pusztán a lexikai egységek számozásában lehetne eltérés, de ebben az esetben még abban sincs, mert az elemzett mondat szerkezete szinte teljesen párhuzamos a két nyelvben; később olyan példákat is bemutatok, ahol ez nincs így).

(44)
```
gramme("Peter loves the clever Hungarian girl.").

LEXICAL ITEMS:

Peter:  n(1,1,li(m("","Peter",""),labsteme("Peter",1,[["0"]])))

love:  n(2,1,li(m("","love",""),labsteme("love",2,[["NOM","ACC"]])))
s:  n(2,2,li(m("E","s",""),labsuffe("E/3",2,"infl")))

the:  n(3,1,li(m("","the",""),labsteme("the",3,[])))

clever:  n(4,1,li(m("","clever",""),labsteme("clever",4,[])))

Hungarian:  n(5,1,li(m("","Hungarian",""),labsteme("Hungarian",4,[])))

girl:  n(6,1,li(m("","girl",""),labsteme("girl",1,[])))

SYNTAX:

gr("noun","regent","subj",1,1,2,1)
gr("det","regent","_",1,1,2,1)
gr("regent","noun","subj",2,1,1,1)
gr("regent","det","subj",2,1,1,1)
gr("regent","noun","obj",2,1,6,1)
gr("regent","det","obj",2,1,3,1)
gr("det","noun","free",3,1,6,1)
gr("det","regent","_",3,1,2,1)
gr("adj","noun","free",4,1,6,1)
gr("adj","noun","free",5,1,6,1)
gr("noun","regent","obj",6,1,2,1)

regent-noun-subj: loves-Peter
regent-det-subj: loves-Peter
regent-noun-obj: loves-girl
regent-det-obj: loves-the
```

```
det-noun: the-girl
adj-noun: clever-girl
adj-noun: Hungarian-girl

SEMANTICS:

provref("fixpoint",[e(2,1,1)])
provref("old",[r(1,1,1)])
pred("Peter",1,[r(1,1,1)])
provref("new",[e(2,1,1)])
pred("love",2,[e(2,1,1),r(1,1,1),r(3,1,1)])
provref("old",[r(3,1,1)])
provref("<or=",[r(3,1,1),e(2,1,1)])
pred("clever",4,[r(3,1,1)])
pred("Hungarian",5,[r(3,1,1)])
pred("girl",6,[r(3,1,1)])

yes
```

A gépi fordításhoz elemzőnk kétirányú használatával juthatunk el: a program először ellenőrzi a forrásnyelvi mondat grammatikalitását és előállítja a különböző kimeneteket, majd generálja a célnyelvi mondatot a kimenetek alapján. A Prolog programnyelv egyik leghasznosabb tulajdonsága, hogy amit elemezni képes, azt generálni is[29], így a generáláshoz nem kellett külön mechanizmusokat kidolgozni. A kérdés az, hogy melyik reprezentáció alapján készíthető el a legoptimálisabb fordítás, vagyis honnan kiindulva célszerű a generálást elindítani. A legnyilvánvalóbb válasz az lehetne, hogy a szintaktikai viszonyok alapján, azonban ezen a szinten bizonyos mondatok esetében túl kevés információ áll rendelkezésünkre. Például a *Péter dolgoztatja Marit* mondatnál csupán a szintaktikai viszonyok alapján nem derül ki, hogy valójában Mari végez cselekvést. Egy másik példa az idiómák fordítása, ha nem hivatkoznánk a jelentésre, teljesen rossz megoldást kapnánk.

A korábbi elképzelésünk az volt, hogy a szemantikai reprezentáció, azaz a (nyelvfüggetlen) DRS elég információt tartalmaz a szerkezethű fordításhoz, de valójában ezen a szinten már történik információvesztés a szintaktikai viszonyokhoz képest (például hogy valamit vonzatként vagy szabad bővítményként fejeztünk-e ki). A DRS továbbá több redundáns információt is tartalmaz, így az erről a szintről való fordítás nehézkes és néhol nem teljesen szerkezethű lenne.

Ha tehát egyik szint sem felel meg tökéletesen arra a célra, hogy a fordítás kiindulópontja lehessen, akkor egy új szintet kell létrehoznunk a szintaxis és a szemantika közé, amely még tartalmazza a releváns szintaktikai viszonyokat, de már kiegészül bizonyos szemantikai információkkal (és nem tartalmaz redundanciát). Így a fordítás könnyebbé, és ami még fontosabb, szerkezethűvé válik. Ez a szint a *kopredikációs viszonyok* szintje („kopredikálás”: ugyanarról állítás) (Alberti et al. 2007a).

Ez az absztrakt szint tehát csak a releváns információkat tartalmazza, szemben a DRS-sel, amely tartalmaz redundanciát. Például *A magas fiú ül* mondat esetében a DRS a következőképpen néz ki: provref(r1)[30], magas(r1), fiú(r1), ül(r1), tehát a *magas* és az *ül* szavak között egy hamis kopredikálás van, az információ, hogy ugyanarról a referensről tesznek állítást, valójában nem releváns.

A másik – már szintén említett – nagyon fontos tulajdonsága ennek a szintnek, hogy a szerkezetőrzést biztosítani tudja, ami lényeges szempont a fordításkor. Természetesen

---

[29] A `gramm("Péter szereti Marit.")` kérdésre a válasz: `yes`; ha azonban nem konkrét szavakat, hanem (szavakat helyettesítő) változókat adunk meg, a program generálja az összes lehetséges „bemenetet”. Például egy korábbi programverzióban, amikor a lexikai egységek még szavak voltak, a `gramm(X,Y,Z)` kérdésre válaszul a program kilistázta az összes három szóból álló grammatikus magyar mondatot.

[30] Azaz bevezetünk egy referenst (provide referent).

előfordul, hogy nem lehet csupán a kopredikációs viszonyok alapján fordítani (például nincs egy bizonyos kifejezésnek megfelelője a célnyelvben), ekkor van szükség a DRS-ben kódolt információkra; azaz a jelentése alapján próbáljuk lefordítani a mondatot. A kopredikációs hálózat tehát nem mutat különbséget nagyon különbözőnek tekintett nyelvek esetében sem. Ha megnézzük például a korábban elemzett *Péter szereti az okos magyar lányt* magyar mondat kopredikációs hálózatát, amely (45) alatt olvasható, és összehasonlítjuk a mondat angol nyelvű változatának (*Peter loves the clever Hungarian girl*) kopredikációs hálózatával, amely (46)-ban látható[31], azt tapasztaljuk, hogy a két reprezentáció teljesen megegyezik (később arra is hozok példát, hogy a hálózatok akkor is megegyeznek – csak a morfémák számozásában térnek el –, amikor a két mondat szerkezete nagyon különböző).

(45) KOPREDIKÁCIÓS VISZONYOK:

```
copr("love",2,1,"Peter",1,1,1,1,"arg")
copr("love",2,1,"Peter",1,1,1,0,"arg")
copr("love",2,1,"girl",6,1,2,1,"arg")
copr("love",2,1,"the",3,1,2,0,"arg")
copr("the",3,1,"girl",6,1,0,1,"free")
copr("clever",4,1,"girl",6,1,1,1,"free")
copr("Hungarian",5,1,"girl",6,1,1,1,"free")
```

(46) COPREDICATIVE NETWORK:

```
copr("love",2,1,"Peter",1,1,1,1,"arg")
copr("love",2,1,"Peter",1,1,1,0,"arg")
copr("love",2,1,"girl",6,1,2,1,"arg")
copr("love",2,1,"the",3,1,2,0,"arg")
copr("the",3,1,"girl",6,1,0,1,"free")
copr("clever",4,1,"girl",6,1,1,1,"free")
copr("Hungarian",5,1,"girl",6,1,1,1,"free")
```

Ha megnézzük például az első sort, az azt állítja, hogy kopredikálás van a ’love’ predikátum (2,1: a második szó első morfémája) és a ’Peter’ predikátum (1,1: az első szó első morfémája) között, mégpedig úgy, hogy az előbbi első argumentuma azonos az utóbbi első argumentumával (1,1), és a viszony vonzatviszony (arg), szemben a szabad bővítményi (free) viszonnyal. A második sor annyiban tér el, hogy az utolsó szám 0, vagyis a ’love’ predikátum a ’Peter’ predikátummal mint determinánsi pillérrel kopredikál (a 4. sorban is 0 szerepel, ahol a tárgy determinánsi pillérével kopredikál az ige). Ugyanígy 0 látható, ha egy predikátum eventuális referensével való kopredikálást állítunk (pl. *kerestet*: ’okoz’ és ’keres’ közötti kopredikálás, az előbbi második argumentuma (amit okoz) és az utóbbi nulladik argumentuma (a keresés maga) között áll fenn a kopredikáció).

Ha a forrásnyelvi kopredikációs háló már rendelkezésünkre áll, alapvetően generáltatással próbáljuk a célnyelvi mondatot előállíttatni a programmal, amelynek azonban van egy nagyon fontos feltétele: a célnyelvi mondatot alkotó lexikai egységek számát ismerni kell. A valódi információt hordozó predikátumok esetében ez nem probléma, hiszen azokat minden nyelven szerepeltetni kell (esetleg nem ugyanannyi szó fejezi ki, de mindenképpen ugyanannyi lexikai egység). A gondot az esetért és az egyeztetésért felelős morfémák (ragok) jelentik, amelyek általában nem hordoznak információt[32], ezért nem is

---

[31] A kopredikációs hálózat egyike a reprezentációknak, amiket a program kiír, ha a mondatot grammatikusnak ítéli; ebben az esetben tehát a `gramm("Péter szereti az okos magyar lányt.").` kérdésre kapjuk meg (45)-öt, és a `gramme("Peter loves the clever Hungarian girl.").` kérdésre kapjuk meg (46)-ot (mindkét esetben a szintaktikai és a szemantikai reprezentáció között listázza ki a program).

[32] Van, amikor a ragok nem csupán egyeztetési szerepet töltenek be, például a *Látlak* mondatban: mivel a névmások hiányoznak, csak a személyragokból tudunk következtetni az alany és a tárgy számára és személyére, tehát ezt az állítást ők hordozzák.

jelennek meg a szemantikai reprezentációban; a célnyelvi mondatban azonban szerepelniük kell, különben nem lesz grammatikus a kimenet. Ezek az elemek nem csupán nyelvenként különböznek (mi az, amit megjelenít egy nyelv, és mi az, amit nem), hanem egy-egy nyelven belül is. Például a magyarban az igéken mindig jelöljük az első személyt, a harmadikat azonban gyakran nem, vagy tranzitív igék esetében egyedül a második személyű tárgyat jelöljük az igén, ahogy az angol nyelv is csak az egyes szám harmadik személyű alanyra alkalmaz egyeztető morfémát.

Ezért a generálást úgy végezzük, hogy a ténylegesen információt hordozó lexikai egységek mellett különböző változókat is feltételezünk, amelyek helyére esetért, illetve egyeztetésért felelős morfémákat kell a programnak találni. Ezek száma, típusa és helye univerzálisan kötött, ezért véges sok ilyen elemet kell keresni, amit úgy végzünk, hogy minden nyelvben feltételezünk minden olyan egyeztető morfémát, amely univerzálisan előfordulhat. Azt mondjuk például, hogy a magyarban mindig jelölve van tranzitív igéken a tárgy személye: második személyt jelent, ha *'l'* alakban jelenik meg, bármely más személyt jelent, ha üres alakban jelenik meg. Ugyanígy az angolban is elvileg mindig keresünk az alany és a tárgy számára, személyére, nemére és határozottságára utaló ragokat az igén, vagy akár esetragokat a főneveken. Ez a megoldás nem tűnik ugyan túl gazdaságosnak, arra azonban van mód, hogy egy-egy nyelv esetében lerögzítsük, hogy melyek azok a lexikai egységek, amelyek nem csupán üres testtel tudnak megjelenni az adott nyelvben, azaz valóban érdemes a programnak keresnie őket az igei vagy névszói tövek után (tehát az ilyen elemek köre nyelvenként tovább szűkíthető). A magyarban ezek a főnéven található esetragok és az igén található – alany számára és személyére, illetve a tárgy határozottságára, esetleg személyére vonatkozó információkat kódoló – ragok. A generálás során a változók helyére többféle érték betöltődhet, a célnyelvi elemző (grammatikalitás-ellenőrző) feladata kiszűrni a hibás alakokat és csak a helyeset hagyni meg[33].

Ha a szükséges célnyelvi lexikai egységeket megtaláltuk, a generálás utolsó lépése következik, vagyis a morfémafüzér, illetve szófüzér (a mondat) előállítása. Ehhez az egyik fontos információ a fixpont, ami a DRS-ből kiolvasható. A legtöbb esetben ez a finit igére mutat, illetve a magyar nyelv esetében az utolsó képzőre, amely még állítást tesz (névszói állítmány esetében a névszóra). A szórend pedig a kopredikációs viszonyok és a szintaxisban tárolt morfémasorrendre vonatkozó információk (rangparaméterek) alapján adódik. Először a lexikai egységek minden lehetséges variációját előállítjuk, ezután kiszűrjük a triviálisan lehetetlen változatokat, végül a célnyelvi elemzőbe épített megelőzési ragparaméterek csak a helyes szórendű mondatot találják grammatikusnak, így az lesz a forrásnyelvi mondat fordítása.

Konkrétan az előbbi példánál maradva (*Péter szereti az okos magyar lányt* mondat fordítása) a következő egységek szeretnének egymással szomszédosak lenni, hiszen közöttük szintaktikai–szemantikai kapcsolat van a kopredikációs háló alapján: *Peter love, love girl, the girl, clever girl, Hungarian girl*. Mindegyik szomszédossági (közvetlen megelőzési) igény természetesen nem teljesülhet, hiszen egy szót csak *egy* másik előzhet meg közvetlenül. Ekkor kapnak szerepet az adott nyelvre jellemző rangparaméterek, amelyek rögzítik, hogy melyik igény milyen erősségű. Például a névelő követelménye, hogy szomszédos legyen a főnévvel, gyengébb, mint a melléknévé, továbbá melléknevek között a nemzetiségnév szomszédossági igénye erősebb, mint a többi melléknévé, így tehát a *the clever Hungarian girl* sorrend adódik. Ugyanilyen megfontolásokat véve figyelembe a többi lexikai egység esetében is a program végül generálni tudja a helyes szórendet: *Peter loves the clever Hungarian girl*. Ha a mondatot szeretnénk visszafordítani magyarra, ugyanezt a

[33] A processzálási idő csökkentése érdekében természetesen egyéb szűrők is beépíthetők a programba.

mechanizmust kell követnie a programnak, a különbség csak az lesz, hogy más változókat (egyeztetésért felelős morfémákat) kell majd keresnie. A mondat végleges fordítása (47)-ben olvasható, ahol tehát a program egy magyar–angol fordítást hajt végre, (48) pedig azt az eredményt mutatja, amit akkor kapunk, ha a mondatot visszafordítjuk angolról magyarra.[34]

```
(47)   translate_Hun_Eng("Péter szereti az okos magyar lányt.").
       In English:  Peter loves the clever Hungarian girl.
       yes

(48)   translate_Eng_Hun("Peter loves the clever Hungarian girl.").
       In Hungarian:  Péter szereti az okos magyar lányt.
       yes
```

Összefoglalva tehát a gépi fordításhoz két alapvető problémát kell megoldani: megtalálni a célnyelvi lexikai egységeket, és előállítani azok helyes sorrendjét. Az első esetében változókat is feltételezünk, hogy az információt nem hordozó esetet és egyeztetést kifejező morfémákat is megtaláljuk, amelyek közül majd a célnyelvi elemző választja ki a megfelelőket. A helyes szórend kialakításához pedig előállítjuk a lehetséges változatokat, amelyek közül szintén a célnyelvi elemző fogja majd meghagyni a grammatikusakat (a közvetlen megelőzési rangparaméterek segítségével).

4.2.6 Példák

Az eddigiekben egy egyszerű példán keresztül mutattam be az elemző működését, és hogy milyen reprezentációkat társít a mondatokhoz, illetve hogyan valósítja meg a gépi fordítást. Ebben a pontban további példákon (programfutásokon) mutatom be, hogy a program milyen nyelvi jelenségek kezelésére képes.

4.2.6.1. Morfofonológia

A morfológiáról szóló fejezetben részletesebben kifejtettem, milyen módokon történik annak ellenőrzése, hogy a mondatban a lexikai egységek megfelelő allomorfja szerepel-e. A program számot tud adni a különböző tőalternációkról (pl. nyúlás, rövidülés, hangkivetés), biztosítja, hogy a tő változójának a megfelelő értéke töltődjön be. Így tehát grammatikusnak ítéli (49)-et, vagyis kilistázza az adott szót alkotó lexikai egységeket, majd kiírja, hogy `yes`. (50)-nél viszont az unifikáció eredménye (üres magánhangzó az *O* változó helyére) nem egyezik meg a bevitt alakkal (az *o* magánhangzóval), amely így rosszul formáltnak minősül, ezért a program a `no` választ adja.

```
(49)   grammword("Bokrot.").
       bokr:  n(1,1,li(m("bok","O","r"),labstem("bush",phonfst(2,1,2,2),1,[])))
       ot:  n(1,2,li(m("V","t",""),labsuff("ACC",phonfsu(1,1,1,3),1,4))
       yes
```

[34] A fordításkor használt célpredikátumok: translate_Eng_Hun: fordítás angolról magyarra, translate_Hun_Eng: fordítás magyarról angolra, translate_Eng_Hun_print: fordítás angolról magyarra úgy, hogy mindkét nyelven kiírja az elemzést is, végül translate_Hun_Eng_print: fordítás magyarról angolra szintén mindkét nyelvű elemzés (nyelvenként négy reprezentációs szint) megjelenítésével.

```
(50)   grammword("Bokort.").

       no
```

A toldalékok esetében is ugyanez a helyzet: az unifikáció során nyert változóértéknek meg kell egyeznie a bevitt értékkel. Ez teljesül (51)-ben, így az grammatikus lesz, nem teljesül azonban (52)-ben, mert a *3* változó értéke *e* kellene, hogy legyen, amely nem tud unifikálódni az *o*-val, így a válasz `no`. A program az érték kiszámolásakor figyelembe veszi magánhangzóknál az elölségi és a kerekségi harmóniát, a kötőhangok meglétére vonatkozó megszorításokat és a nyitás jelenségét; mássalhangzóknál pedig az írásban jelölt hasonulásokat.

```
(51)   grammword("Könyveihez.").

       könyv:  n(1,1,li(m("","könyv",""),labstem("book",
                                          phonfst(1,1,2,1),1,[["DELATIV"]])))
       ei:  n(1,2,li(m("J","A","i"),labsuff("plposs",phonfsu(1,1,1,1),1,1)))
       hez:  n(1,3,li(m("h","3","z"),labsuff("ALLATIV",phonfsu(1,2,2,3),1,4)))

       yes

(52)   grammword("Könyveihoz.").

       no
```

A szón belüli morfémasorrendről is részletesen szóltam a morfológiát ismertető fejezetben, a főnévi közvetlen megelőzési viszonyokat a 4. ábra, míg az igeieket az 5. ábra ismertette. A program tehát (ennek alapján) grammatikusnak ítéli az (53)-ban látható főnevet, de az (54) kérdésre a válasza `no`, mert a *kalapjaimrólé* morfsorban a 4-es rangú rag megelőzi a 3,5-es rangú birtoklást jelölő morfémát (ami nem lehetséges). Ugyanígy jól formált lesz az (55)-ben olvasható összes szó, mert morfémasorrendjük megfelel az igei rangparamétereknek, viszont rosszul formált az (56)-ban lévő *vártálhat* morfsor, mert benne az 1-es rangú modalitást kifejező morféma a 2-es, illetve 3-as rangú elemek mögé került.

```
(53)   grammword("Kalapjaiméiról.").

       kalap:  n(1,1,li(m("","kalap",""),labstem("hat",phonfst(2,2,2,2),1,[])))
       jai:  n(1,2,li(m("J","A","i"),labsuff("plposs",phonfsu(1,1,1,1),1,1)))
       m:  n(1,3,li(m("V","m",""),labsuff("possI",phonfsu(1,1,1,1),1,2)))
       é:  n(1,4,li(m("","é",""),labsuff("poss",phonfsu(1,2,2,1),1,3)))
       i:  n(1,5,li(m("","i",""),labsuff("pl2",phonfsu(1,1,1,1),1,3)))
       ról:  n(1,6,li(m("r","Ó","l"),labsuff("DELATIV",phonfsu(1,2,2,3),1,4)))

       yes

(54)   grammword("Kalapjaimrólé.").

       no

(55)   grammword("Várhattál várhatnál várhattál volna.").

       vár:  n(1,1,li(m("","vár",""),labstem("wait",
                             phonfst(2,2,2,2),2,[["NOM","ACC"],["NOM","SUBLATIV"]])))
       hat:  n(1,2,li(m("h","A","t"),labsuff("may",phonfsu(1,1,1,2),2,1)))
       t:  n(1,3,li(m("V","t","T"),labsuff("past",phonfsu(3,1,1,1),2,2)))
       ál:  n(1,4,li(m("Á","l",""),labsuff("sg2",phonfsu(1,2,2,3),2,3)))

       vár:  n(2,1,li(m("","vár",""),labstem("wait",
                             phonfst(2,2,2,2),2,[["NOM","ACC"],["NOM","SUBLATIV"]])))
       hat:  n(2,2,li(m("h","A","t"),labsuff("may",phonfsu(1,1,1,2),2,1)))
       ná:  n(2,3,li(m("n","É",""),labsuff("condindef",phonfsu(1,1,1,2),2,2)))
       l:  n(2,4,li(m("Á","l",""),labsuff("sg2",phonfsu(1,2,2,3),2,3)))
```

```
vár:  n(3,1,li(m("","vár",""),labstem("wait",
                        phonfst(2,2,2,2),2,[["NOM","ACC"],["NOM","SUBLATIV"]])))
hat:  n(3,2,li(m("h","A","t"),labsuff("may",phonfsu(1,1,1,2),2,1)))
t:  n(3,3,li(m("V","t","T"),labsuff("past",phonfsu(3,1,1,1),2,2)))
ál:  n(3,4,li(m("Á","l",""),labsuff("sg2",phonfsu(1,2,2,3),2,3)))
volna:  n(3,5,li(m("","volna",""),labsuff("cond",phonfsu(3,3,3,3),2,4)))

yes
```

```
(56)  grammword("Vártálhat.").

      no
```

4.2.6.2. Többértelműség

A Prologban lehetőség van arra is, hogy minden lehetséges megoldást kiírassunk a programmal. Ehhez arra van szükség, hogy a célpredikátum után a `fail` parancsot is kiadjuk, ekkor a program – miután egy megoldást már adott – hamisra fogja értékelni a predikátumot, így újabb megoldást keres (amit szintén megad), egészen addig, amíg nem talál többet, és kiírja, hogy `no`. Például a *dobom* szónak morfológiailag több elemzése is lehetséges (egy igei és egy főnévi), ezek olvashatók (57)-ben.

```
(57)  grammword("Dobom."), fail.

      dob: n(1,1,li(m("","dob",""),labstem("throw",phonfst(2,1,2,2),2,[["NOM","ACC"]])))
      om:  n(1,2,li(m("V","m",""),labsuff("sg1obj+def",phonfsu(1,1,1,3),2,3)))

      dob:  n(1,1,li(m("","dob",""),labstem("drum",phonfst(2,1,2,2),1,[])))
      om:  n(1,2,li(m("V","m",""),labsuff("possI",phonfsu(1,1,1,1),1,2)))

      no
```

Szintaktikailag többértelmű a *Péter keresi a lányt a kutyával* mondat, attól függően, hogy a *kutyával* szabad bővítmény Péterhez, a lányhoz vagy a kereséshez kapcsolódik, háromféle értelmezést kapunk. (58)-ban olvasható a mondathoz rendelt szemantikai reprezentáció.

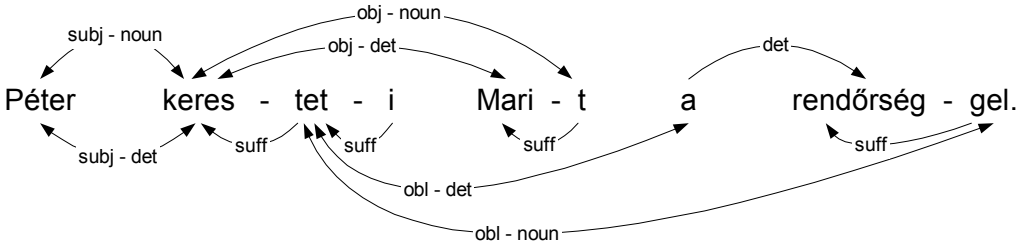
```
(58)  gramm("Péter keresi a lányt a kutyával.").

      SZEMANTIKA:
      provref("fixpoint",[e(2,1,1)])
      provref("old",[r(1,1,1)])
      pred("Peter",1,[r(1,1,1)])
      provref("new",[e(2,1,1)])
      pred("look-for",2,[e(2,1,1),r(1,1,1),r(3,1,1)])
      provref("old",[r(3,1,1)])
      provref("<or=",[r(3,1,1),e(2,1,1)])
      pred("girl",4,[r(3,1,1)])
      provref("old",[r(5,1,1)])
      provref("<or=",[r(5,1,1),e(2,1,1)])
      pred("dog",6,[r(5,1,1)])
      provref("new",[e(6,2,1)])
      pred("with",6,[e(6,2,1),or_e(2,1,1),or_r(1,1,1),or_r(3,1,1),r(5,1,1)])
```

A DRS utolsó sorában olvasható, hogy az e621 eventuális referens azt állítja, hogy *vagy* e211 (a keresés) *vagy* r111 (Péter) *vagy* r311 (Mari) referensekre igaz, hogy az r511 (kutya) referensével vannak („with”). Nem kell tehát a mondathoz három különböző elemzést társítani (ami a hatékonyságot jelentősen csökkenthetné), hiszen a lehetséges olvasatok a DRS-ből egyértelműen kiolvashatók.

4.2.6.3. Műveltetés

A műveltetés képzője ugyanolyan lexikai egység a programban, mint például egy ige: tároljuk a különféle (morfológiai, szintaktikai, szemantikai) tulajdonságait, illetve az elvárásait. Amit mindenképpen keres, az egy igei tő, de általában még egy elemet keres, hogy pontosan mit, az attól függ, hogy tranzitív vagy intranzitív igei tövet talál meg. Ha tárgyatlan igére tapad rá (pl. énekel), akkor (általában) egy tárgyat fog behozni a mondatba, ha pedig tárgyas igére, akkor egy -val/-vel ragos bővítményt fog keresni (pontosabban magát a ragot és egy determinánsi pillért). Ez utóbbira példa a *Péter keresteti Marit a rendőrséggel* mondat, amelynek a szintaktikai viszonyai a 8. ábraán láthatók.



8. ábra: Műveltetést tartalmazó szintaktikai viszonyrendszer

A program a mondathoz az (59)-ben látható reprezentációkat társítja (Kleiber 2007b).

(59)
```
gramm("Péter keresteti Marit a rendőrséggel.").

LEXIKAI EGYSÉGEK:

Péter:  n(1,1,li(m("","Péter",""),labstem("Peter",phonfst(1,2,0,2),1,[])))

keres:  n(2,1,li(m("","keres",""),labstem("look-for",
                                    phonfst(1,2,2,2),2,[["NOM","ACC"]])))
tet:  n(2,2,li(m("t","A","t"),labder("cause",phonfsu(2,2,0.2,2),2,ac(-1,0,1))))
i:  n(2,3,li(m("","i",""),labsuff("sg3obj+def",phonfsu(1,3,1,3),2,3)))

Mari:  n(3,1,li(m("","Mari",""),labstem("Mary",phonfst(2,2,0,2),1,[])))
t:  n(3,2,li(m("V","t",""),labsuff("ACC",phonfsu(1,1,1,3),1,4)))

a:  n(4,1,li(m("","a","Z"),labstem("the",phonfst(1,3,3,3),3,[])))

rendőrség: n(5,1,li(m("","rendőrség",""),labstem("police",phonfst(1,2,3,2),1,[])))
gel:  n(5,2,li(m("S","A","l"),labsuff("INSTR",phonfsu(1,2,2,3),1,4)))


SZINTAXIS:

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

regent-noun-subj: keresteti-Péter
regent-det-subj: keresteti-Péter
regent-noun-obj: keresteti-Marit
```

```
regent-det-obj: keresteti-Marit
regent-noun-obl: keresteti-rendőrséggel
regent-det-obl: keresteti-a
det-noun: a-rendőrséggel

KOPREDIKÁCIÓS VISZONYOK:

copr("look-for",2,1,"Peter",1,1,1,1,"arg")
copr("look-for",2,1,"Peter",1,1,1,0,"arg")
copr("look-for",2,1,"Mary",3,1,2,1,"arg")
copr("look-for",2,1,"Mary",3,1,2,0,"arg")
copr("cause",2,2,"Peter",1,1,1,1,"arg")
copr("cause",2,2,"Peter",1,1,1,0,"arg")
copr("cause",2,2,"look-for",2,1,2,0,"arg")
copr("the",4,1,"police",5,1,0,1,"free")

SZEMANTIKA:

provref("fixpoint",[e(2,2,1)])
provref("old",[r(1,1,1)])
pred("Peter",1,[r(1,1,1)])
provref("new",[e(2,1,1)])
pred("look-for",2,[e(2,1,1),r(4,1,1),r(3,1,1)])
provref("new",[e(2,2,1)])
provref("=",[e(2,2,1),e(2,1,1)])
pred("cause",2,[e(2,2,1),r(1,1,1),e(2,1,1)])
provref("old",[r(3,1,1)])
pred("Mary",3,[r(3,1,1)])
provref("old",[r(4,1,1)])
provref("<or=",[r(4,1,1),e(2,2,1)])
pred("police",5,[r(4,1,1)])

yes
```

r1	r3	r4	e21	e22
Péter(r1) Mari(r3) rendőrség(r4) e22: okoz(r1,e21) e21: keres(r4,r3) ...				

A szemantikai reprezentációból (a DRS-ből) kiolvasható, hogy van három szereplőnk: r1, r3 és r4 (a számok arra utalnak, hogy a mondatban hányadik szó a determinánsi pillérük), ahol r1 Péter, r3 Mari, r4 pedig a rendőrség. Van továbbá két eseményünk, e21 és e22 (a számok arra utalnak, hogy hányadik szó hányadik morfémája a predikátum), ahol e21 az, hogy r4 (a rendőrség) keresi r3-at (Marit), e22 pedig az, hogy r1 (Péter) okozza e21-et, vagyis azt, hogy a rendőrség keresi Marit. A mondat diskurzusba való ágyazásához szükséges predikátumok a program által kiírt reprezentációban olvashatók. E szerint a fixpont a műveltetés képzőjére mutat, *Péter* (akárcsak *Mari*) egy már bevezetett („old") referensre utal vissza (hiszen tulajdonnév), a két eventuális referens korábban még nem szerepelt („új" referensek), melyek azonos világba vezetődnek be („="), a *rendőrség* referense is ismert entitás („old"), mert határozott névelővel utalunk rá, végül pedig megtudjuk, hogy a *rendőrség* referense egy korábbi világban „született", mint a műveltetés eventuális referense (vagy esetleg ugyanabban).

A műveltetés egy olyan nyelvi jelenség, amelyet a különböző nyelvek nagyon különböző módokon tudnak kifejezni. A magyar nyelv képzőt használ erre a célra, míg az angol önálló szavakat (*make, have*). A morféma alapú totálisan lexikalista megközelítés egységesen tudja ezt kezelni, hiszen mindegyik esetben az adott morféma egy önálló lexikai egysége az adott nyelvnek, és irreleváns, hogy toldalékként vagy szóként jelenik-e meg. A nagyfokú különbözőség fordításkor sem okoz problémát, hiszen nem csupán a szemantika, hanem már a kopredikáció szintjén sem látható ez a felszíni különbség. A következő példa ezt érzékelteti. A műveltetésen túl tartalmaz modalitásért felelős elemet is, amelyet szintén eltérően fejez ki az angol és a magyar nyelv (önálló szó vs. képző). (60)-ban a *Péter énekeltetheti Marit* mondat fordítása látható, mindkét nyelvű elemzéssel (az összes reprezentációval) együtt (Kleiber 2007a).

(60) translate_Hun_Eng_print("Péter énekeltetheti Marit.").

LEXIKAI EGYSÉGEK:

```
Péter:  n(1,1,li(m("","Péter",""),labstem("Peter",phonfst(1,2,0,2),1,[])))

énekel:  n(2,1,li(m("","énekel",""),labstem("sing",phonfst(1,2,2,2),2,[["NOM"]])))
tet:  n(2,2,li(m("t","A","t"),labder("cause",phonfsu(2,2,0.2,2),2,ac(-1,0,1))))
het:  n(2,3,li(m("h","A","t"),labsuff("may",phonfsu(1,1,1,2),2,1)))
i:  n(2,4,li(m("","i",""),labsuff("sg3obj+def",phonfsu(1,3,1,3),2,3)))

Mari:  n(3,1,li(m("","Mari",""),labstem("Mary",phonfst(2,2,0,2),1,[])))
t:  n(3,2,li(m("V","t",""),labsuff("ACC",phonfsu(1,1,1,3),1,4)))
```

SZINTAXIS:

```
gr("noun","regent","subj",1,1,2,1)
gr("det","regent","_",1,1,2,1)
gr("regent","noun","subj",2,1,1,1)
gr("regent","det","subj",2,1,1,1)
gr("suff","stem","free",2,2,2,1)
gr("regent","noun","obj",2,2,3,2)
gr("regent","det","obj",2,2,3,1)
gr("suff","stem","free",2,3,2,1)
gr("suff","stem","free",2,4,2,1)
gr("det","regent","_",3,1,2,1)
gr("suff","stem","free",3,2,3,1)
gr("noun","regent","obj",3,2,2,1)

regent-noun-subj: énekeltetheti-Péter
regent-det-subj: énekeltetheti-Péter
regent-noun-obj: énekeltetheti-Marit
regent-det-obj: énekeltetheti-Marit
```

KOPREDIKÁCIÓS VISZONYOK:

```
copr("sing",2,1,"Mary",3,1,1,1,"arg")
copr("sing",2,1,"Mary",3,1,1,0,"arg")
copr("cause",2,2,"Peter",1,1,1,1,"arg")
copr("cause",2,2,"Peter",1,1,1,0,"arg")
copr("cause",2,2,"sing",2,1,2,0,"arg")
copr("may",2,3,"cause",2,2,1,0,"arg")
```

SZEMANTIKA:

```
provref("fixpoint",[e(2,3,1)])
provref("old",[r(1,1,1)])
pred("Peter",1,[r(1,1,1)])
provref("new",[e(2,1,1)])
pred("sing",2,[e(2,1,1),r(3,1,1)])
provref("new",[e(2,2,1)])
provref("=",[e(2,2,1),e(2,1,1)])
pred("cause",2,[e(2,2,1),r(1,1,1),e(2,1,1)])
provref("new",[e(2,3,1)])
provref("<",[e(2,3,1),e(2,2,1)])
pred("may",2,[e(2,3,1),e(2,2,1)])
provref("old",[r(3,1,1)])
pred("Mary",3,[r(3,1,1)])
```

In English: Peter may make Mary sing.

LEXICAL ITEMS:

```
li(m("","Peter",""),labsteme("Peter",1,[["0"]]))
li(m("","may",""),labsteme("may",2,[["VERB"]]))
li(m("","",""),labsuffe("not-E/3",2,"infl"))
li(m("","make",""),labsteme("cause",2,[["NOM","VERB"]]))
li(m("","",""),labsuffe("not-E/3",2,"infl"))
li(m("","Mary",""),labsteme("Mary",1,[["0"]]))
li(m("","sing",""),labsteme("sing",2,[["NOM"]]))
```

```
li(m("","",""),labsuffe("not-E/3",2,"infl"))

LEXICAL ITEMS without empty elements:

Peter:  n(1,1,li(m("","Peter",""),labsteme("Peter",1,[["0"]])))
may:  n(2,1,li(m("","may",""),labsteme("may",2,[["VERB"]])))
make:  n(3,1,li(m("","make",""),labsteme("cause",2,[["NOM","VERB"]])))
Mary:  n(4,1,li(m("","Mary",""),labsteme("Mary",1,[["0"]])))
sing:  n(5,1,li(m("","sing",""),labsteme("sing",2,[["NOM"]])))

SYNTAX:

gr("noun","regent","subj",1,1,3,1)
gr("det","regent","_",1,1,3,1)
gr("regent","verb","arg",2,1,3,1)
gr("regent","noun","subj",3,1,1,1)
gr("regent","det","subj",3,1,1,1)
gr("regent","verb","arg",3,1,5,1)
gr("noun","regent","subj",4,1,5,1)
gr("det","regent","_",4,1,5,1)
gr("regent","noun","subj",5,1,4,1)
gr("regent","det","subj",5,1,4,1)

regent-verb-arg: may-make
regent-noun-subj: make-Peter
regent-det-subj: make-Peter
regent-verb-arg: make-sing
regent-noun-subj: sing-Mary
regent-det-subj: sing-Mary

COPREDICATIVE NETWORK:

copr("may",2,1,"cause",3,1,1,0,"arg")
copr("cause",3,1,"Peter",1,1,1,1,"arg")
copr("cause",3,1,"Peter",1,1,1,0,"arg")
copr("cause",3,1,"sing",5,1,2,0,"arg")
copr("sing",5,1,"Mary",4,1,1,1,"arg")
copr("sing",5,1,"Mary",4,1,1,0,"arg")

SEMANTICS:

provref("fixpoint",[e(2,1,1)])
provref("old",[r(1,1,1)])
pred("Peter",1,[r(1,1,1)])
provref("new",[e(2,1,1)])
provref("<",[e(2,1,1),e(3,1,1)])
pred("may",2,[e(2,1,1),e(3,1,1)])
provref("new",[e(3,1,1)])
provref("=",[e(3,1,1),e(5,1,1)])
pred("cause",3,[e(3,1,1),r(1,1,1),e(5,1,1)])
provref("old",[r(4,1,1)])
pred("Mary",4,[r(4,1,1)])
provref("new",[e(5,1,1)])
pred("sing",5,[e(5,1,1),r(4,1,1)])

yes
```

Megfigyelhető, hogy a két nyelv kopredikációs hálózata a mondatok közötti szerkezetbeli eltérések ellenére is nagyon hasonló: pusztán az elemek sorszámában, illetve a felsorolás sorrendjében térnek el egymástól.

4.2.6.4. Zéró névmások

A program hasznosságát a géppel segített fordításban talán a zéró névmásokat tartalmazó mondatok elemzésén lehet leginkább bemutatni. Hiszen ha a magyar mondat minden állítást egy-egy (önálló) szóval fejez ki (mint a *Péter szereti az okos magyar lányt* mondat

esetében), bárki, aki rendelkezik egy magyar–angol szótárral, képes lefordítani a mondatot angolra, nincs tehát szüksége a program által kiírt szemantikai reprezentációra, illetve az nem nyújt neki sokkal többet, mint egy kétnyelvű szótár. Ha azonban egy magyarul nem tudó külföldi azt a mondatot látja, hogy *Szeretlek*, arra talán rájön egy szótár segítségével, hogy a 'love' predikátum szerepel benne, azt viszont nem fogja tudni kikövetkeztetni, hogy ki szeret kit. Ennek oka, hogy a magyar ún. pro-drop nyelv, azaz a névmások sokszor elhagyhatók, és csak a ragozásból tudunk a szereplők kilétére következtetni. Ha viszont beírja ezt a mondatot a programba, megkapja a szemantikai reprezentációt, ahol olvashatja, hogy aki szeret, az r011 (0: beépített, mindig jelenlévő referens, 11: egyes szám első személyű, vagyis 'én'), akit szeret, az pedig r012 (egyes szám második személyű névmás, vagyis 'te'). (61) az említett mondat fordítását mutatja, ahol először az elemzését olvashatjuk, amelynek legfontosabb sora a szemantikai reprezentációban olvasható (félkövérrel szedve), majd az angol fordítást, végül annak az elemzését.

(61) `translate_Hun_Eng_print("Szeretlek").`

```
LEXIKAI EGYSÉGEK:

szeret:  n(1,1,li(m("","szeret",""),labstem("love",
                                        phonfst(1,2,2,2),2,[["NOM","ACC"]])))
l: n(1,2,li(m("","l",""),labsuff("objperson2",phonfsu(3,2,1,1),2,2.5)))
ek: n(1,3,li(m("V","k",""),labsuff("sg1",phonfsu(1,1,2,3),2,3)))

SZINTAXIS:

gr("suff","stem","free",1,2,1,1)
gr("suff","stem","free",1,3,1,1)

KOPREDIKÁCIÓS VISZONYOK:

copr("love",1,1,"sg1",1,3,1,1,"arg")
copr("love",1,1,"objperson2",1,2,2,1,"arg")

SZEMANTIKA:

provref("fixpoint",[e(1,1,1)])
provref("new",[e(1,1,1)])
pred("love",1,[e(1,1,1),r(0,1,1),r(0,1,2)])

In English:  I love you.

LEXICAL ITEMS:

li(m("","I",""),labsteme("I",1,[["0","sg","1","NOM"]]))
li(m("","love",""),labsteme("love",2,[["NOM","ACC"]]))
li(m("","",""),labsuffe("not-E/3",2,"infl"))
li(m("","you",""),labsteme("you",1,[["0","_","2","_"]]))

LEXICAL ITEMS without empty elements:

I:  n(1,1,li(m("","I",""),labsteme("I",1,[["0","sg","1","NOM"]])))

love:  n(2,1,li(m("","love",""),labsteme("love",2,[["NOM","ACC"]])))

you:  n(3,1,li(m("","you",""),labsteme("you",1,[["0","_","2","_"]])))

SYNTAX:

gr("noun","regent","subj",1,1,2,1)
gr("det","regent","_",1,1,2,1)
gr("regent","noun","subj",2,1,1,1)
gr("regent","det","subj",2,1,1,1)
gr("regent","noun","obj",2,1,3,1)
gr("regent","det","obj",2,1,3,1)
gr("noun","regent","obj",3,1,2,1)
gr("det","regent","_",3,1,2,1)
```

```
regent-noun-subj: love-I
regent-det-subj: love-I
regent-noun-obj: love-you
regent-det-obj: love-you

COPREDICATIVE NETWORK:

copr("love",2,1,"I",1,1,1,1,"arg")
copr("love",2,1,"I",1,1,1,0,"arg")
copr("love",2,1,"you",3,1,2,1,"arg")
copr("love",2,1,"you",3,1,2,0,"arg")

SEMANTICS:

provref("fixpoint",[e(2,1,1)])
provref("old",[r(1,1,1)])
pred("=",1,[r(1,1,1),r(0,1,1)])
provref("new",[e(2,1,1)])
pred("love",2,[e(2,1,1),r(1,1,1),r(3,1,1)])
provref("old",[r(3,1,1)])
pred("you",3,[r(3,1,1)])

yes
```

Ha egy mondatnak több célnyelvi megfelelője is van, a program ki tudja írni az összes lehetséges megoldást (a korábban már említett `fail` parancs segítségével). (62)-ben az látható, hogy az *I love you* angol mondatnak elvileg hat magyarra fordítása lehetséges, amelyek közül az első (ami azt is jelenti, hogy elsődleges, vagyis ha nincs `fail` parancs, az egyetlen) a *Szeretlek* mondat, vagyis amikor a névmásokat nem tesszük ki. Ez a legsemlegesebb, a többi esetben valami (minimálisan pragmatikai) pluszjelentést hordoz a magyar mondat (topik, fókusz, amelyeket egyelőre nem kezel a program).

```
(62)   translate_Eng_Hun("I love you."),fail.

       In Hungarian: Szeretlek.
       In Hungarian: Szeretlek téged.
       In Hungarian: Szeretlek titeket.
       In Hungarian: Én szeretlek.
       In Hungarian: Én szeretlek téged.
       In Hungarian: Én szeretlek titeket.
       no
```

(63) egy olyan programfutást mutat, ahol a magyar mondat zéró névmást és műveltetést is tartalmaz (az elemzéseket is kiírja), majd (64)-ben az így nyert angol mondat visszafordítását láthatjuk magyarra, ahol szintén alkalmaztuk a `fail` parancsot, így mindkét fordítást megkaptuk (az elsődleges a névmás nélküli változat).

```
(63)   translate_Hun_Eng_print("Énekeltethetjük Pétert.").

       LEXIKAI EGYSÉGEK:

       énekel:  n(1,1,li(m("","énekel",""),labstem("sing",phonfst(1,2,2,2),2,[["NOM"]])))
       tet:  n(1,2,li(m("t","A","t"),labder("cause",phonfsu(2,2,0.2,2),2,ac(-1,0,1))))
       het:  n(1,3,li(m("h","A","t"),labsuff("may",phonfsu(1,1,1,2),2,1)))
       jük:  n(1,4,li(m("j","U","k"),labsuff("pl1def",phonfsu(2,1,1,3),2,3)))

       Péter:  n(2,1,li(m("","Péter",""),labstem("Peter",phonfst(1,2,0,2),1,[])))
       t:  n(2,2,li(m("V","t",""),labsuff("ACC",phonfsu(1,1,1,3),1,4)))

       SZINTAXIS:

       gr("suff","stem","free",1,2,1,1)
       gr("regent","noun","obj",1,2,2,2)
       gr("regent","det","obj",1,2,2,1)
       gr("suff","stem","free",1,3,1,1)
```

```
gr("suff","stem","free",1,4,1,1)
gr("det","regent","_",2,1,1,1)
gr("suff","stem","free",2,2,2,1)
gr("noun","regent","obj",2,2,1,1)
```

```
regent-noun-obj: énekeltethetjük-Pétert
regent-det-obj: énekeltethetjük-Pétert
```

KOPREDIKÁCIÓS VISZONYOK:

```
copr("sing",1,1,"Peter",2,1,1,1,"arg")
copr("sing",1,1,"Peter",2,1,1,0,"arg")
copr("cause",1,2,"pl1def",1,4,1,1,"arg")
copr("cause",1,2,"sing",1,1,2,0,"arg")
copr("may",1,3,"cause",1,2,1,0,"arg")
```

SZEMANTIKA:

```
provref("fixpoint",[e(1,3,1)])
provref("new",[e(1,1,1)])
pred("sing",1,[e(1,1,1),r(2,1,1)])
provref("new",[e(1,2,1)])
provref("=",[e(1,2,1),e(1,1,1)])
pred("cause",1,[e(1,2,1),r(0,2,1),e(1,1,1)])
provref("new",[e(1,3,1)])
provref("<",[e(1,3,1),e(1,2,1)])
pred("may",1,[e(1,3,1),e(1,2,1)])
provref("old",[r(2,1,1)])
pred("Peter",2,[r(2,1,1)])
```

In English: We may make Peter sing.

LEXICAL ITEMS:

```
li(m("","we",""),labsteme("we",1,[["0","pl","1","NOM"]]))
li(m("","may",""),labsteme("may",2,[["VERB"]]))
li(m("","",""),labsuffe("not-E/3",2,"infl"))
li(m("","make",""),labsteme("cause",2,[["NOM","VERB"]]))
li(m("","",""),labsuffe("not-E/3",2,"infl"))
li(m("","Peter",""),labsteme("Peter",1,[["0"]]))
li(m("","sing",""),labsteme("sing",2,[["NOM"]]))
li(m("","",""),labsuffe("not-E/3",2,"infl"))
```

LEXICAL ITEMS without empty elements:

```
we:  n(1,1,li(m("","we",""),labsteme("we",1,[["0","pl","1","NOM"]])))
```

```
may:  n(2,1,li(m("","may",""),labsteme("may",2,[["VERB"]])))
```

```
make:  n(3,1,li(m("","make",""),labsteme("cause",2,[["NOM","VERB"]])))
```

```
Peter:  n(4,1,li(m("","Peter",""),labsteme("Peter",1,[["0"]])))
```

```
sing:  n(5,1,li(m("","sing",""),labsteme("sing",2,[["NOM"]])))
```

SYNTAX:

```
gr("noun","regent","subj",1,1,3,1)
gr("det","regent","_",1,1,3,1)
gr("regent","verb","arg",2,1,3,1)
gr("regent","noun","subj",3,1,1,1)
gr("regent","det","subj",3,1,1,1)
gr("regent","verb","arg",3,1,5,1)
gr("noun","regent","subj",4,1,5,1)
gr("det","regent","_",4,1,5,1)
gr("regent","noun","subj",5,1,4,1)
gr("regent","det","subj",5,1,4,1)
```

```
regent-verb-arg: may-make
regent-noun-subj: make-we
regent-det-subj: make-we
regent-verb-arg: make-sing
```

```
regent-noun-subj: sing-Peter
regent-det-subj: sing-Peter

COPREDICATIVE NETWORK:

copr("may",2,1,"cause",3,1,1,0,"arg")
copr("cause",3,1,"we",1,1,1,1,"arg")
copr("cause",3,1,"we",1,1,1,0,"arg")
copr("cause",3,1,"sing",5,1,2,0,"arg")
copr("sing",5,1,"Peter",4,1,1,1,"arg")
copr("sing",5,1,"Peter",4,1,1,0,"arg")

SEMANTICS:

provref("fixpoint",[e(2,1,1)])
provref("old",[r(1,1,1)])
pred("=",1,[r(1,1,1),r(0,2,1)])
provref("new",[e(2,1,1)])
provref("<",[e(2,1,1),e(3,1,1)])
pred("may",2,[e(2,1,1),e(3,1,1)])
provref("new",[e(3,1,1)])
provref("=",[e(3,1,1),e(5,1,1)])
pred("cause",3,[e(3,1,1),r(1,1,1),e(5,1,1)])
provref("old",[r(4,1,1)])
pred("Peter",4,[r(4,1,1)])
provref("new",[e(5,1,1)])
pred("sing",5,[e(5,1,1),r(4,1,1)])

yes
```

```
(64)  translate_Eng_Hun("We may make Peter sing."),fail.

In Hungarian:  Énekeltethetjük Pétert.
In Hungarian:  Mi énekeltethetjük Pétert.
no
```

Az előbbi példa végsőkig vitt változata olvasható (65)-ben, vagyis egy olyan mondat, ahol műveltetés, modalitás és zéró névmások is szerepelnek.

```
(65)  translate_Hun_Eng_print("Énekeltethetlek.").

LEXIKAI EGYSÉGEK:

énekel:  n(1,1,li(m("","énekel",""),labstem("sing",phonfst(1,2,2,2),2,[["NOM"]])))
tet:  n(1,2,li(m("t","A","t"),labder("cause",phonfsu(2,2,0.2,2),2,ac(-1,0,1))))
het:  n(1,3,li(m("h","A","t"),labsuff("may",phonfsu(1,1,1,2),2,1)))
l:  n(1,4,li(m("","l",""),labsuff("objperson2",phonfsu(3,2,1,1),2,2.5)))
ek:  n(1,5,li(m("V","k",""),labsuff("sg1",phonfsu(1,1,2,3),2,3)))

SZINTAXIS:

gr("suff","stem","free",1,2,1,1)
gr("suff","stem","free",1,3,1,1)
gr("suff","stem","free",1,4,1,1)
gr("suff","stem","free",1,5,1,1)

KOPREDIKÁCIÓS VISZONYOK:

copr("sing",1,1,"objperson2",1,4,1,1,"arg")
copr("cause",1,2,"sg1",1,5,1,1,"arg")
copr("cause",1,2,"sing",1,1,2,0,"arg")
copr("may",1,3,"cause",1,2,1,0,"arg")

SZEMANTIKA:

provref("fixpoint",[e(1,3,1)])
provref("new",[e(1,1,1)])
pred("sing",1,[e(1,1,1),r(0,1,2)])
```

```
provref("new",[e(1,2,1)])
provref("=",[e(1,2,1),e(1,1,1)])
pred("cause",1,[e(1,2,1),r(0,1,1),e(1,1,1)])
provref("new",[e(1,3,1)])
provref("<",[e(1,3,1),e(1,2,1)])
pred("may",1,[e(1,3,1),e(1,2,1)])
```

In English: I may make you sing.

LEXICAL ITEMS:

```
li(m("","I",""),labsteme("I",1,[["0","sg","1","NOM"]]))
li(m("","may",""),labsteme("may",2,[["VERB"]]))
li(m("","",""),labsuffe("not-E/3",2,"infl"))
li(m("","make",""),labsteme("cause",2,[["NOM","VERB"]]))
li(m("","",""),labsuffe("not-E/3",2,"infl"))
li(m("","you",""),labsteme("you",1,[["0","_","2","_"]]))
li(m("","sing",""),labsteme("sing",2,[["NOM"]]))
li(m("","",""),labsuffe("not-E/3",2,"infl"))
```

LEXICAL ITEMS without empty elements:

```
I:  n(1,1,li(m("","I",""),labsteme("I",1,[["0","sg","1","NOM"]])))
```

```
may:  n(2,1,li(m("","may",""),labsteme("may",2,[["VERB"]])))
```

```
make:  n(3,1,li(m("","make",""),labsteme("cause",2,[["NOM","VERB"]])))
```

```
you:  n(4,1,li(m("","you",""),labsteme("you",1,[["0","_","2","_"]])))
```

```
sing:  n(5,1,li(m("","sing",""),labsteme("sing",2,[["NOM"]])))
```

SYNTAX:

```
gr("noun","regent","subj",1,1,3,1)
gr("det","regent","_",1,1,3,1)
gr("regent","verb","arg",2,1,3,1)
gr("regent","noun","subj",3,1,1,1)
gr("regent","det","subj",3,1,1,1)
gr("regent","verb","arg",3,1,5,1)
gr("noun","regent","subj",4,1,5,1)
gr("det","regent","_",4,1,5,1)
gr("regent","noun","subj",5,1,4,1)
gr("regent","det","subj",5,1,4,1)
```

```
regent-verb-arg: may-make
regent-noun-subj: make-I
regent-det-subj: make-I
regent-verb-arg: make-sing
regent-noun-subj: sing-you
regent-det-subj: sing-you
```

COPREDICATIVE NETWORK:

```
copr("may",2,1,"cause",3,1,1,0,"arg")
copr("cause",3,1,"I",1,1,1,1,"arg")
copr("cause",3,1,"I",1,1,1,0,"arg")
copr("cause",3,1,"sing",5,1,2,0,"arg")
copr("sing",5,1,"you",4,1,1,1,"arg")
copr("sing",5,1,"you",4,1,1,0,"arg")
```

SEMANTICS:

```
provref("fixpoint",[e(2,1,1)])
provref("old",[r(1,1,1)])
pred("=",1,[r(1,1,1),r(0,1,1)])
provref("new",[e(2,1,1)])
provref("<",[e(2,1,1),e(3,1,1)])
pred("may",2,[e(2,1,1),e(3,1,1)])
provref("new",[e(3,1,1)])
provref("=",[e(3,1,1),e(5,1,1)])
```

```
pred("cause",3,[e(3,1,1),r(1,1,1),e(5,1,1)])
provref("old",[r(4,1,1)])
pred("you",4,[r(4,1,1)])
provref("new",[e(5,1,1)])
pred("sing",5,[e(5,1,1),r(4,1,1)])

yes
```

Látható, hogy a mondatok szerkezete közötti nagyfokú eltérés ellenére a két mondat kopredikációs hálózata és a hozzájuk rendelt szemantikai reprezentáció lényegileg nem különbözik egymástól. Visszafordításkor a `fail` paranccsal a szokásos változatokat kapjuk:

```
(66)   translate_Eng_Hun("I may make you sing."), fail.

       In Hungarian:  Énekeltethetlek.
       In Hungarian:  Énekeltethetlek téged.
       In Hungarian:  Énekeltethetlek titeket.
       In Hungarian:  Én énekeltethetlek téged.
       In Hungarian:  Én énekeltethetlek titeket.
       no
```

4.2.6.5. Igekötő

Korábban már volt szó arról, hogy ha egy igekötőt produktívan használunk, az gyakran behoz a mondatba valamilyen (általában szigorúan meghatározott esetben álló) vonzatot (pl. be vmibe, ki vmiből, le vmiről, rá vmire). A program ilyen típusú mondatokat is tud elemezni, és hozzájuk szemantikai reprezentációt társítani (Alberti et al. 2003, Balogh–Kleiber 2003). Ha az eset nem megfelelő, a válasz `no` lesz, vagyis nem grammatikus a mondat (67).

```
(67)   gramm("A fiú ráülteti a lányt a kocsiba.").

       no
```

(68)-ban egy olyan programfutás látható, ahol a *be* igekötő szerepel, amelynek produktív használata behozza a mondatba a *székembe* illatívuszi esetben álló bővítményt. A szintaktikai reprezentációban félkövérrel szedtem azokat a sorokat, ahol az igekötő által létesített viszonyok láthatók: keres egy tövet, amire rátapadhat (pref–stem–free viszony), meg is találja az *ül* igét (4,2); illetve keres egy oblikvuszi bővítményt két pilléren (noun és det), főnévi pillérnek a *székembe* szót találja meg (pontosabban a ragját: 9,3), determinánsi pillérnek pedig a szó névelőjét (8,1). A szemantikai reprezentációban az látható (szintén félkövérrel szedve), hogy az igekötő és az ige egy predikátumot alkot (’sit-into’), vagyis szemantikailag összeolvadnak.

```
(68)   gramm("A magyar fiú beültetheti az okos medvéjét a székembe.").

       LEXIKAI EGYSÉGEK:

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

       magyar:  n(2,1,li(m("","magyar",""),labstem("Hungarian",phonfst(2,2,2,1),4,[])))

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

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

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

```
okos:  n(6,1,li(m("","okos",""),labstem("clever",phonfst(2,1,2,1),4,[])))

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

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

szék:  n(9,1,li(m("","szék",""),labstem("chair",phonfst(1,2,3,2),1,[])))
em:  n(9,2,li(m("V","m",""),labsuff("possI",phonfsu(1,1,1,1),1,2)))
be:  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("Hungarian",[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("clever",[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",[r(0,1,1),r(8,1,1)])

yes
```

A fenti példa tartalmaz továbbá műveltetést (-tAt), modalitást kifejező morfémát (-hAt), illetve birtokos személyjeleket is. Ez utóbbiak a dőlt betűvel szedett sorokat adják hozzá a szemantikai reprezentációhoz, vagyis hogy valaki (egyes szám harmadik személyű entitás)

birtokolja r511-et (a medvét), illetve hogy az r011-gyel jelölt entitás (vagyis ’én’, 0: beépített referens, 1,1: egyes szám első személyű) birtokolja r811-et (vagyis a széket).

4.2.6.6. Vonzatos melléknév

Az adatbázis olyan mellékneveket is tartalmaz, amelyeknek lehetnek saját vonzataik, mint például a ’büszke’ (valakire). Egy ilyen nyelvi konstrukciót tartalmazó mondat elemzését mutatja (69).

(69)

```
gramm("Péter szereti a Marira büszke holland lányt.").

LEXIKAI EGYSÉGEK:

Péter:  n(1,1,li(m("","Péter",""),labstem("Peter",phonfst(1,2,0,2),1,[])))

szeret:  n(2,1,li(m("","szeret",""),labstem("love",
                                        phonfst(1,2,2,2),2,[["NOM","ACC"]])))
i:  n(2,2,li(m("","i",""),labsuff("sg3obj+def",phonfsu(1,3,1,3),2,3)))

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

Mari:  n(4,1,li(m("","Mari",""),labstem("Mary",phonfst(2,2,0,2),1,[])))
ra:  n(4,2,li(m("r","A",""),labsuff("SUBLATIV",phonfsu(1,2,2,3),1,4)))

büszke:  n(5,1,li(m("","büszke",""),labstem("proud",
                                        phonfst(1,2,2,1),4,[["SUBLATIV"]])))

holland:  n(6,1,li(m("","holland",""),labstem("Dutch",phonfst(2,2,1,1),4,[])))

lány:  n(7,1,li(m("","lány",""),labstem("girl",phonfst(2,2,3,2),1,[])))
t:  n(7,2,li(m("V","t",""),labsuff("ACC",phonfsu(1,1,1,3),1,4)))

SZINTAXIS:

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

regent-noun-subj: szereti-Péter
regent-det-subj: szereti-Péter
regent-noun-obj: szereti-lányt
regent-det-obj: szereti-a
det-noun: a-lányt
adj-noun: büszke-lányt
regent-noun-obl: büszke-Marira
regent-det-obl: büszke-Marira
adj-noun: holland-lányt

KOPREDIKÁCIÓS VISZONYOK:

copr("love",2,1,"Peter",1,1,1,1,"arg")
copr("love",2,1,"Peter",1,1,1,0,"arg")
copr("love",2,1,"girl",7,1,2,1,"arg")
copr("love",2,1,"the",3,1,2,0,"arg")
```

```
copr("the",3,1,"girl",7,1,0,1,"free")
copr("proud",5,1,"girl",7,1,1,1,"free")
copr("proud",5,1,"Mary",4,1,2,1,"arg")
copr("proud",5,1,"Mary",4,1,2,0,"arg")
copr("Dutch",6,1,"girl",7,1,1,1,"free")

SZEMANTIKA:

provref("fixpoint",[e(2,1,1)])
provref("old",[r(1,1,1)])
pred("Peter",1,[r(1,1,1)])
provref("new",[e(2,1,1)])
pred("love",2,[e(2,1,1),r(1,1,1),r(3,1,1)])
provref("old",[r(3,1,1)])
provref("<or=",[r(3,1,1),e(2,1,1)])
provref("old",[r(4,1,1)])
pred("Mary",4,[r(4,1,1)])
pred("proud",5,[r(3,1,1),r(4,1,1)])
pred("Dutch",6,[r(3,1,1)])
pred("girl",7,[r(3,1,1)])

yes
```

A program angol nyelvű vonzatos mellékneveket is tartalmaz, a szórendjük azonban kicsit más, mint a magyar nyelv esetében (a rangparaméterek kapcsán volt már róla szó): a melléknév, amely alapesetben a főnév előtt helyezkedik el, ha vonzatot kap, a főnév mögé kerül. Ezért nem lesz grammatikus (70).

```
(70)   gramme("The proud of Mary Dutch girl loves Peter.").

       no
```

A következő néhány programfutás a két nyelv közötti fordításra példa, ahol a mondatok vonzatos melléknevet tartalmaznak:

```
(71)   translate_Hun_Eng("A Marira büszke holland fiú szereti Julit.").

       In English:  The Dutch boy proud of Mary loves Julie.
       yes

(72)   translate_Eng_Hun("The boy proud of Mary loves Julie.").

       In Hungarian:  A Marira büszke fiú szereti Julit.
       yes

(73)   translate_Hun_Eng("A rám váró fiú szereti Marit.").

       In English:  The boy waiting for me loves Mary.
       yes

(74)   translate_Eng_Hun("The boy waiting for me loves Mary.").

       In Hungarian:  Az engem váró fiú szereti Marit.
       yes
```

Érdekes nyelvészeti problémákat vet fel az az eset, amikor a melléknév vonzata (a bővített köznévvel megegyező határozottságú) köznév (pl. *a fiúra büszke lány*), amikor is a magyarban névelőtörlés történik (Alberti et al. 2004b). A probléma kezelését határozottsági rangparaméterekkel (Alberti–Balogh 2003) oldjuk meg. A (75)-ben olvasható mondat esetében például két 'a' névelőre lenne szükség szintaktikailag, de hangtanilag csak egy realizálódik. További érdekessége a mondatnak, hogy dupla beágyazást tartalmaz, vagyis a melléknév (büszke) vonzatát (lányra) is bővíti egy jelző (váró), amelynek szintén van vonzata (Pétert). (76)-ban a mondat angol változatának a magyarra fordítása látható mindkét nyelvű elemzéssel együtt.

(75) translate_Hun_Eng("A Pétert váró lányra büszke fiú énekel.").

```
In English:  The boy proud of the girl waiting for Peter sings.
yes
```

(76) translate_Eng_Hun_print("The boy proud of the girl waiting for Peter sings.").

```
LEXICAL ITEMS:

the:  n(1,1,li(m("","the",""),labsteme("the",3,[])))
boy:  n(2,1,li(m("","boy",""),labsteme("boy",1,[])))
proud:  n(3,1,li(m("","proud",""),labsteme("proud",4,[["of"]])))
of:  n(4,1,li(m("","of",""),labsteme("of",8,[["of"]])))
the:  n(5,1,li(m("","the",""),labsteme("the",3,[])))
girl:  n(6,1,li(m("","girl",""),labsteme("girl",1,[])))
waiting:  n(7,1,li(m("","waiting",""),labsteme("waiting",4,[["for"]])))
for:  n(8,1,li(m("","for",""),labsteme("for",8,[["for"]])))
Peter:  n(9,1,li(m("","Peter",""),labsteme("Peter",1,[["0"]])))
sing:  n(10,1,li(m("","sing",""),labsteme("sing",2,[["NOM"]])))
s:  n(10,2,li(m("E","s",""),labsuffe("E/3",2,"infl")))

SYNTAX:

gr("det","noun","free",1,1,2,1)
gr("det","regent","_",1,1,10,1)
gr("noun","regent","subj",2,1,10,1)
gr("adj","noun","free",3,1,2,1)
gr("regent","noun","obl",3,1,6,1)
gr("regent","det","obl",3,1,5,1)
gr("det","noun","free",5,1,6,1)
gr("det","regent","_",5,1,3,1)
gr("noun","regent","obl",6,1,3,1)
gr("adj","noun","free",7,1,6,1)
gr("regent","noun","obl",7,1,9,1)
gr("regent","det","obl",7,1,9,1)
gr("noun","regent","obl",9,1,7,1)
gr("det","regent","_",9,1,7,1)
gr("regent","noun","subj",10,1,2,1)
gr("regent","det","subj",10,1,1,1)

det-noun: the-boy
adj-noun: proud-boy
regent-noun-obl: proud-girl
regent-det-obl: proud-the
det-noun: the-girl
adj-noun: waiting-girl
regent-noun-obl: waiting-Peter
regent-det-obl: waiting-Peter
regent-noun-subj: sings-boy
regent-det-subj: sings-the

COPREDICATIVE NETWORK:

copr("the",1,1,"boy",2,1,0,1,"free")
copr("proud",3,1,"boy",2,1,1,1,"free")
copr("proud",3,1,"girl",6,1,2,1,"arg")
copr("proud",3,1,"the",5,1,2,0,"arg")
copr("the",5,1,"girl",6,1,0,1,"free")
copr("waiting",7,1,"girl",6,1,1,1,"free")
copr("waiting",7,1,"Peter",9,1,2,1,"arg")
copr("waiting",7,1,"Peter",9,1,2,0,"arg")
copr("sing",10,1,"boy",2,1,1,1,"arg")
copr("sing",10,1,"the",1,1,1,0,"arg")

SEMANTICS:

provref("fixpoint",[e(10,1,1)])
provref("old",[r(1,1,1)])
determiner("<or=",[r(1,1,1),e(0,0,0)])
pred("boy",2,[r(1,1,1)])
```

```
pred("proud",3,[r(1,1,1),r(5,1,1)])
provref("old",[r(5,1,1)])
determiner("<or=",[r(5,1,1),e(0,0,0)])
pred("girl",6,[r(5,1,1)])
pred("waiting",7,[r(5,1,1),r(9,1,1)])
provref("old",[r(9,1,1)])
pred("Peter",9,[r(9,1,1)])
provref("new",[e(10,1,1)])
pred("sing",10,[e(10,1,1),r(1,1,1)])
```

In Hungarian: A Péterre váró lányra büszke fiú énekel.

LEXIKAI EGYSÉGEK:

```
li(m("","a","Z"),labstem("the",phonfst(1,3,3,3),3,[]))
li(m("","Péter",""),labstem("Peter",phonfst(1,2,0,2),1,[]))
li(m("r","A",""),labsuff("SUBLATIV",phonfsu(1,2,2,3),1,4))
li(m("","váró",""),labstem("waiting",phonfst(2,1,2,1),4,[["ACC"],["SUBLATIV"]]))
li(m("","lány",""),labstem("girl",phonfst(2,2,3,2),1,[]))
li(m("r","A",""),labsuff("SUBLATIV",phonfsu(1,2,2,3),1,4))
li(m("","büszke",""),labstem("proud",phonfst(1,2,2,1),4,[["SUBLATIV"]]))
li(m("","fiú",""),labstem("boy",phonfst(2,1,3,2),1,[]))
li(m("","",""),labsuff("NOM",phonfsu(3,3,3,3),1,4))
li(m("","énekel",""),labstem("sing",phonfst(1,2,2,2),2,[["NOM"]]))
li(m("","",""),labsuff("objpersonnot2",phonfsu(3,3,3,3),2,2.5))
li(m("","",""),labsuff("sg3obj-def",phonfsu(3,3,3,3),2,3))
```

LEXIKAI EGYSÉGEK üres elemek nélkül:

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

Péter:  n(2,1,li(m("","Péter",""),labstem("Peter",phonfst(1,2,0,2),1,[])))
re:  n(2,2,li(m("r","A",""),labsuff("SUBLATIV",phonfsu(1,2,2,3),1,4)))

váró:  n(3,1,li(m("","váró",""),labstem("waiting",
                                phonfst(2,1,2,1),4,[["ACC"],["SUBLATIV"]])))

lány:  n(4,1,li(m("","lány",""),labstem("girl",phonfst(2,2,3,2),1,[])))
ra:  n(4,2,li(m("r","A",""),labsuff("SUBLATIV",phonfsu(1,2,2,3),1,4)))

büszke:  n(5,1,li(m("","büszke",""),labstem("proud",
                                phonfst(1,2,2,1),4,[["SUBLATIV"]])))

fiú:  n(6,1,li(m("","fiú",""),labstem("boy",phonfst(2,1,3,2),1,[])))

énekel:  n(7,1,li(m("","énekel",""),labstem("sing",phonfst(1,2,2,2),2,[["NOM"]])))
```

SZINTAXIS:

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

```
det-noun: a-fiú
adj-noun: váró-lányra
regent-noun-obl: váró-Péterre
regent-det-obl: váró-Péterre
adj-noun: büszke-fiú
```

```
regent-noun-obl: büszke-lányra
regent-det-obl: büszke-lányra
regent-noun-subj: énekel-fiú
regent-det-subj: énekel-a

KOPREDIKÁCIÓS VISZONYOK:

copr("the",1,1,"boy",6,1,0,1,"free")
copr("waiting",3,1,"girl",4,1,1,1,"free")
copr("waiting",3,1,"Peter",2,1,2,1,"arg")
copr("waiting",3,1,"Peter",2,1,2,0,"arg")
copr("the",4,1,"girl",4,1,0,1,"free")
copr("proud",5,1,"boy",6,1,1,1,"free")
copr("proud",5,1,"girl",4,1,2,1,"arg")
copr("proud",5,1,"girl",4,1,2,0,"arg")
copr("sing",7,1,"boy",6,1,1,1,"arg")
copr("sing",7,1,"the",1,1,1,0,"arg")

SZEMANTIKA:

provref("fixpoint",[e(7,1,1)])
provref("old",[r(1,1,1)])
provref("<or=",[r(1,1,1),e(7,1,1)])
provref("old",[r(2,1,1)])
pred("Peter",2,[r(2,1,1)])
pred("waiting",3,[r(4,1,1),r(2,1,1)])
provref("old",[r(4,1,1)])
provref("<or=",[r(4,1,1),e(7,1,1)])
pred("girl",4,[r(4,1,1)])
pred("proud",5,[r(1,1,1),r(4,1,1)])
pred("boy",6,[r(1,1,1)])
provref("new",[e(7,1,1)])
pred("sing",7,[e(7,1,1),r(1,1,1)])

yes
```

A hangtanilag nem realizálódó névelő problémáját az elmélet úgy oldja meg, hogy nem csak a tulajdonnév, hanem bizonyos kontextusokban akár a köznév is tartalmazhatja a saját determinánsi pillérjét (Alberti–Balogh 2003), és egy ilyen kontextus lehet például a vonzatos melléknevek esete. A fenti (*A Péterre váró lányra büszke fiú énekel*) mondatban három determináló elemnek kell lenni szemantikailag: az *a* névelő tartozik a *fiú* köznévhez, a *Péter* mint tulajdonnév tartalmazza a saját determinánsi pillérjét, hiányzik azonban a *lányra* szóhoz tartozó determinánsi pillér, amit az elmélet értelmében a köznév önmagában tud megtalálni (ebben a speciális esetben). Ez látható a kopredikációs hálózaton is (a releváns sorokat félkövérrel szedtem): a 'the' predikátumot a negyedik szó első morfémája tartalmazza (4,1: vagyis a *lányra* szó), amely önmagával ('girl',4,1) létesít determinánsi viszonyt (0,1, free); illetve a 'proud' predikátum második argumentumának determinánsi pillére (2,0) ugyanúgy a *lányra* kifejezés (4,1), mint ahogy a főnévi pillére (2,1) is az (a felette lévő sorban). Az angol nyelvű reprezentációban ez a jelenség nem tapasztalható, hiszen ott más a szórend, és nem kerül egymás mellé a két egyforma névelő.

4.2.6.7. Mellérendelés

Az utolsó nyelvi jelenség, amelynek kezeléséről a dolgozatban beszámolok, a mellérendelés. Hogy a megközelítés működőképességét kipróbáljuk, első lépésként a főnévi kifejezések közötti bináris 'és' kezelését építettük a programba (Alberti et al. 2004a). Ez szintaktikailag úgy történik, mintha az 'és' egy olyan argumentum lenne, amely kötelezően régens is, és van két azonos esetű argumentuma, amelyek sorrendjére szintén szigorú megkötések érvényesek: az 'és' mindig a két argumentuma között áll. Tehát például a *szeret* ige alanyául vagy egy főnévi fejet és egy hozzá tartozó valamilyen determináló elemet talál meg, vagy pedig az *és*

szócskát, az utóbbi esetben az *és* tehát a *szeret* ige egyik argumentuma lesz. Ha pedig az 'és' követelményeinek teljesülését ellenőrzi a program, akkor azt vizsgálja, hogy van-e egy-egy főnévi kifejezés (fej és determináló elem) előtte és utána, azok azonos esetragot viselnek-e, illetve megfelelő-e az igei egyeztetés (erre a következőkben röviden kitérek). Ha bármelyik követelmény nem teljesül, a program kiírja, hogy az adott szósor nem grammatikus.

A mellérendelés behatóbb tanulmányozását tehát a főnévi kifejezések közötti bináris 'és' vizsgálatával kezdtük, ezért most csak annak a tárgyalására fogok szorítkozni. Először a szintaktikai, majd a szemantikai jellemzőit mutatom be, végül néhány programfutáson keresztül megmutatom, hogyan elemzi a program az ilyen típusú mondatokat.

Egyes szám harmadik személyű kifejezések esetében legtöbbször nemigen találunk megszorításokat a mellérendelt tagokra és a mondat egyéb összetevőire. Bármely két főnévi kifejezést össze lehet kapcsolni *és*-sel, ha a thematikus szerepek engedik. Mindkettőnek viselnie kell a megfelelő esetragot (77), és, ha alanyról van szó, utánuk a (harmadik személyű) ige egyes és többes számban is állhat (77)[35].

(77) a. Elvittem a könyvet és a cd-t Péterhez és Marihoz.
b. Péter és Mari nővérének az új munkatársa elkérte/elkérték tőlem a kocsimat.
c. Ő és ő menjen/menjenek haza.[36]

Más a helyzet, ha tárgyról van szó, illetve ha alanyról, és legalább az egyik koordinált összetevő (nem egyes szám harmadik személyű) személyes névmás (a többes számú tartalmas kifejezések a többes szám harmadik személyű névmás mintájára viselkednek). Ekkor további (részben az eddigiektől eltérő) megszorításokra van szükség, és van úgy, hogy két tagot nem is lehet főnévi kifejezésként mellérendelni, mert sehogy sem kapunk grammatikus mondatot. Ennek oka az egyeztetésben keresendő, ami abból is nyilvánvaló, hogy az alany esetében mindig az ige száma/személye a kérdéses, a tárgy esetében pedig annak határozottsága.

Az egyszerűbbik eset az, amikor a névmásokat vizsgáljuk alanyi helyzetben. Létezik egy jól működő szabály (Bánréti 1992): ha az egyik (vagy mindkét) összetevő névmás, az ige többes számban lesz, személye pedig az alacsonyabb személlyel lesz azonos (78).

(78) a. Péter és én elment*ünk* a bálba.
b. Te és Mari olvassá*tok* el a könyvet!
c. Ti és mi mindig egyetért*ünk*.
d. Ő és ők menje*nek*/*menj*en* haza!

A mellérendelt tárgy vizsgálata már nem ilyen egyszerű. Bizonyos esetekben az anyanyelvi beszélők kompetenciája eltér. Ha a két összetevő megegyezik határozottságban, vagy az adott igealak nem érzékeny a határozottságra, nincs ellentmondás az egyeztetésben, így a mondatot mindenki grammatikusnak találja (79).

(79) a. Szeretem Jánost és a mákos tésztát.
b. Kérek egy sütit és egy pohár vizet.
c. Olvastam egy könyvet és a cikkedet.

Ha a két főnévi kifejezés határozottsága nem egyezik meg, és az igéhez közelebb a vele ellentétes határozottságú összetevő van, agrammatikusnak ítéljük a mondatot (80).

---

[35] Ez utóbbi szabály alól kivételt jelentenek azok a szerkezetek, ahol logikailag nem mellérendelés történik (lásd később).
[36] Az utóbbi két mondat két variációja közötti jelentésbeli különbségre még visszatérek.

(80) a. *Szeretem egy holland fiút és Jánost.
b. *Kérek a sütit és egy pohár vizet.
c. *Jánost és egy holland fiút szeretem.

Viszont ha az igéhez közelebb a vele egyező határozottságú főnévi kifejezés van, bizonyos beszélők számára helyes, mások számára agrammatikus lesz a mondat[37] (81).

(81) a. ?Szeretem Jánost és egy holland fiút.
b. ?Kérek egy pohár vizet és a sütimet.
c. ??Egy holland fiút és Jánost szeretem.

Ha a tárgy legalább egyik összetevője személyes névmás, a korábban leírtak érvényesek. Egyetlen eset van, ami újabb problémákat vet fel: amikor a tárgy második személyű, ekkor ugyanis van személybeli egyeztetés a tárggyal is, ami az alanyi egyeztetéshez nagyjából hasonlóan működik (82).

(82) a. Szeretlek téged és titeket.
b. *Szeretem téged és Jánost.
c. ?Szeretlek téged és Jánost.

Logikailag az *és* legtöbbször a konjunkciónak felel meg, vagyis azt állítjuk, hogy mindkét entitásra igaz a predikátum, a DRS-ben tehát két sor fog megjelenni. Vannak azonban olyan predikátumok, amelyek esetében nem ez lesz a jelentés (83).

(83) a. Péter és Mari házastársak.
b. András és Marci barátok.

Itt mindkét esetben egyetlen állításról van szó. A predikátum egy szimmetrikus relációt fejez ki, az *és* pedig a két argumentumát kapcsolja össze. Mivel mindkét argumentum egyforma szerepben van, azok fel is cserélhetőek egymással. A predikátum csak többes számban állhat. Azt az információt, hogy egy predikátum ebbe a típusba tartozik, rögzíteni kell a szótárban, egyrészt azért, hogy csak olyan szerkezetet találjunk grammatikusnak, ahol pontosan kettő (*házastársak*), vagy legalább kettő (*barátok*) mellérendelt főnév fejezi ki az alanyt; másrészt pedig azért, hogy tudjuk, hogy a jelentését máshogy kell megragadni.

Ha két egyes szám harmadik személyű főnévi kifejezés mellérendelése alkotja az alanyt, a predikátum egyes és többes számban is állhat (kivéve az előbb említett esetben, amikor a predikátum csak többes számú lehet), és a két olvasat között finom jelentéskülönbség fedezhető fel (84).

(84) a. Péter és Mari hazament.
b. Péter és Mari hazamentek.

Ahol egyes számban áll a predikátum, ott a disztributív olvasat a preferált (Péter és Mari külön-külön ment haza, vagy legalábbis az az információ releváns, hogy mindketten hazamentek már), ahol pedig többes számban, ott valamiféle csoport olvasat (Péter és Mari együtt mentek haza). (85)-ben talán még erőteljesebb a különbség:

[37] Több megkérdezett szerint (81) grammatikus ugyan, de *is*-sel a végén jobb. Ez arra utal, hogy számukra itt nem a főnévi csoportok szintjén van a mellérendelés, hanem mondatszinten ellipszissel.

(85) a. Péter és Mari teniszezik.
b. Péter és Mari teniszeznek.

A második mondatban valószínűbb, hogy egymással játszanak, míg az elsőben inkább egymástól függetlenül végzik épp most a cselekvést, vagy igaz rájuk, hogy szoktak teniszezni. A két olvasat közötti különbség nem mindig és mindenki számára egyértelmű, lehetnek helyzetek, ahol éppen a másik olvasat kerülhet előtérbe, és az is elképzelhető, hogy bizonyos beszélőknek a két mondat jelentése mindig teljesen ugyanaz. Az implementációban megkülönböztetjük a két jelentést.

Mindezek alapján a program grammatikusnak ítéli a (86)-ban olvasható mondatot, és a következő elemzést társítja hozzá (csak az egyszerűsített szintaxist és a szemantikát másoltam ide, a CD-mellékleten természetesen az összes reprezentáció olvasható):

(86)
```
gramm("Péter és Mari szereti az epret.").

SZINTAXIS:

regent-noun-conj: és-Péter
regent-det-conj: és-Péter
regent-noun-conj: és-Mari
regent-det-conj: és-Mari
regent-noun-subj: szereti-és
regent-det-subj: szereti-és
regent-noun-obj: szereti-epret
regent-det-obj: szereti-az
det-noun: az-epret

SZEMANTIKA:

provref("fixpoint",[e(4,1,1)])
provref("old",[r(1,1,1)])
pred("Peter",1,[r(1,1,1)])
provref("new",[r(2,1,1)])
provref("<or=",[r(2,1,1),e(4,1,1)])
pred("element",2,[r(1,1,1),r(2,1,1)])
pred("element",2,[r(3,1,1),r(2,1,1)])
provref("old",[r(3,1,1)])
pred("Mary",3,[r(3,1,1)])
provref("new",[e(4,1,1)])
pred("love",4,[e(4,1,1),r(1,1,1),r(5,1,1)])
provref("new",[e(4,1,2)])
pred("love",4,[e(4,1,2),r(3,1,1),r(5,1,1)])
provref("old",[r(5,1,1)])
provref("<or=",[r(5,1,1),e(4,1,1)])
pred("strawberry",6,[r(5,1,1)])
```

A szintaktikai reprezentáción látható, hogy régens–főnév, illetve régens–determináns viszonyban van az *és* kötőszó *Péter*-rel, illetve *Mari*-val, a viszony a mellérendelés (`conj`); régens–főnév, illetve régens–determináns viszonyban van a *szereti* ige az *és* kötőszóval, az *és* az ige alanya (`subj`); illetve az is látszik, hogy a tárgy pedig *az epret* kifejezés. A szemantikai reprezentáció szintén a kívánt eredményt mutatja: két sor található a 'love' predikátummal, mint állítással (félkövérrel szedve): az első azt állítja, hogy r111 (vagyis Péter) szereti r511-et (az epret), a második pedig azt, hogy r311 (Mari) szereti r511-et (az epret). Létrejött továbbá egy csoportreferens (r211), amelynek tagjai (element) Péter (r111) és Mari (r311), ami lehetővé teszi, hogy a későbbiekben kettőjükre egy anaforikus elemmel utaljunk vissza („… a szüleik mégsem adnak *nekik* soha").

Az előző mondat a disztributív olvasatot szemléltette: az ige egyes számban szerepelt, a DRS-ben pedig külön sorban állítottuk, hogy mindkét entitásra teljesül a predikátum. (87) ezzel szemben arra példa, amikor a két entitás együtt végzi a cselekvést (csoport olvasat), az

ige többes számú, a DRS-ben pedig egyszer szerepel a 'look-for' predikátum (szintén csak az egyszerűsített szintaxist és a szemantikai reprezentációt szerepeltetem).

```
(87)    gramm("A medve és a kutya keresik a nagymamát.").

        SZINTAXIS:

        det-noun: a-medve
        regent-noun-conj: és-medve
        regent-det-conj: és-a
        regent-noun-conj: és-kutya
        regent-det-conj: és-a
        det-noun: a-kutya
        regent-noun-subj: keresik-és
        regent-det-subj: keresik-és
        regent-noun-obj: keresik-nagymamát
        regent-det-obj: keresik-a
        det-noun: a-nagymamát

        SZEMANTIKA:

        provref("fixpoint",[e(6,1,1)])
        provref("old",[r(1,1,1)])
        provref("<or=",[r(1,1,1),e(6,1,1)])
        pred("bear",2,[r(1,1,1)])
        provref("new",[r(3,1,1)])
        provref("<or=",[r(3,1,1),e(6,1,1)])
        pred("element",3,[r(1,1,1),r(3,1,1)])
        pred("element",3,[r(4,1,1),r(3,1,1)])
        provref("old",[r(4,1,1)])
        provref("<or=",[r(4,1,1),e(6,1,1)])
        pred("dog",5,[r(4,1,1)])
        provref("new",[e(6,1,1)])
        pred("look-for",6,[e(6,1,1),r(3,1,1),r(7,1,1)])
        provref("old",[r(7,1,1)])
        provref("<or=",[r(7,1,1),e(6,1,1)])
        pred("grandma",8,[r(7,1,1)])
```

A kétféle olvasat nem eredményez különböző szintaktikai reprezentációkat, a különbség csak a DRS-ben jelenik meg: ebben az esetben egyetlen állítás szerepel a 'keres' (look-for) predikátummal, amely szerint r311 – vagyis a csoportreferens, melynek tagjai r111 (a medve) és r411 (a kutya) – keresi r711-et (vagyis a nagymamát).

A (78)-ban bemutatott egyeztetési jelenségre példa (88), (89) és (90), ahol az első és a harmadik grammatikus, mert megfelel a szabálynak, a másodikra viszont azt mondja a program, hogy `no`, mert az ige nem többes számú (most csak a szemantikai reprezentációkat szerepeltetem).

```
(88)    gramm("Mari és te kértek egy almát.").

        SZEMANTIKA:
        provref("fixpoint",[e(4,1,1)])
        provref("old",[r(1,1,1)])
        pred("Mary",1,[r(1,1,1)])
        provref("new",[r(2,1,1)])
        provref("<or=",[r(2,1,1),e(4,1,1)])
        pred("element",2,[r(1,1,1),r(2,1,1)])
        pred("element",2,[r(3,1,1),r(2,1,1)])
        pred("=",3,[r(3,1,1),r(0,1,2)])
        provref("new",[e(4,1,1)])
        pred("ask-for",4,[e(4,1,1),r(2,1,1),r(5,1,1)])
        provref("new",[r(5,1,1)])
        provref("<or=",[r(5,1,1),e(4,1,1)])
        pred("apple",6,[r(5,1,1)])

        yes
```

```
(89)    gramm("Mari és te kérsz egy almát.").

        no

(90)    gramm("Egy holland lány és én keresünk egy okos magyar fiút.").

        SZEMANTIKA:
        provref("fixpoint",[e(6,1,1)])
        provref("new",[r(1,1,1)])
        provref("<or=",[r(1,1,1),e(6,1,1)])
        pred("Dutch",2,[r(1,1,1)])
        pred("girl",3,[r(1,1,1)])
        provref("new",[r(4,1,1)])
        provref("<or=",[r(4,1,1),e(6,1,1)])
        pred("element",4,[r(1,1,1),r(4,1,1)])
        pred("element",4,[r(1,1,1),r(4,1,1)])
        pred("=",5,[r(5,1,1),r(0,1,1)])
        provref("new",[e(6,1,1)])
        pred("look-for",6,[e(6,1,1),r(4,1,1),r(7,1,1)])
        provref("new",[r(7,1,1)])
        provref("<or=",[r(7,1,1),e(6,1,1)])
        pred("clever",8,[r(7,1,1)])
        pred("Hungarian",9,[r(7,1,1)])
        pred("boy",10,[r(7,1,1)])

        yes
```

Végül pedig álljon itt egy olyan példa, amelyben vonzatos melléknév is található (most a teljes elemzést szerepeltetem).

```
(91)    gramm("Péter és egy rátok büszke fiú énekel.").

        LEXIKAI EGYSÉGEK:

        Péter:  n(1,1,li(m("","Péter",""),labstem("Peter",phonfst(1,2,0,2),1,[])))

        és:  n(2,1,li(m("","és",""),labstem("and",phonfsu(1,1,1,1),5,[])))

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

        rátok:  n(4,1,li(m("","rátok",""),labstem("youpl",
                                        phonfst(2,2,2,0),1,[["SUBLATIV"]])))

        büszke:  n(5,1,li(m("","büszke",""),labstem("proud",
                                        phonfst(1,2,2,1),4,[["SUBLATIV"]])))

        fiú:  n(6,1,li(m("","fiú",""),labstem("boy",phonfst(2,1,3,2),1,[])))

        énekel:  n(7,1,li(m("","énekel",""),labstem("sing",phonfst(1,2,2,2),2,[["NOM"]])))

        SZINTAXIS:

        gr("noun","regent","conj",1,1,2,1)
        gr("det","regent","conj",1,1,2,1)
        gr("regent","noun","conj",2,1,1,1)
        gr("regent","det","conj",2,1,1,1)
        gr("noun","regent","subj",2,1,7,1)
        gr("regent","noun","conj",2,1,6,1)
        gr("regent","det","conj",2,1,3,1)
        gr("det","regent","subj",2,1,7,1)
        gr("det","noun","free",3,1,6,1)
        gr("det","regent","_",3,1,2,1)
        gr("noun","regent","obl",4,1,5,1)
        gr("det","regent","_",4,1,5,1)
        gr("adj","noun","free",5,1,6,1)
        gr("regent","noun","obl",5,1,4,1)
        gr("regent","det","obl",5,1,4,1)
        gr("noun","regent","conj",6,1,2,1)
        gr("regent","noun","subj",7,1,2,1)
        gr("regent","det","subj",7,1,2,1)
```

```
regent-noun-conj: és-Péter
regent-det-conj: és-Péter
regent-noun-conj: és-fiú
regent-det-conj: és-egy
det-noun: egy-fiú
adj-noun: büszke-fiú
regent-noun-obl: büszke-rátok
regent-det-obl: büszke-rátok
regent-noun-subj: énekel-és
regent-det-subj: énekel-és

KOPREDIKÁCIÓS VISZONYOK:

copr("and",2,1,"Peter",1,1,1,0,"arg")
copr("and",2,1,"Peter",1,1,1,1,"arg")
copr("and",2,1,"a(n)",3,1,2,0,"arg")
copr("and",2,1,"boy",6,1,2,1,"arg")
copr("a(n)",3,1,"boy",6,1,0,1,"free")
copr("a(n)",4,1,"youpl",4,1,0,1,"free")
copr("proud",5,1,"boy",6,1,1,1,"free")
copr("proud",5,1,"youpl",4,1,2,1,"arg")
copr("proud",5,1,"youpl",4,1,2,0,"arg")
copr("sing",7,1,"and",2,1,1,1,"arg")
copr("sing",7,1,"and",2,1,1,0,"arg")

SZEMANTIKA:

provref("fixpoint",[e(7,1,1)])
provref("old",[r(1,1,1)])
pred("Peter",1,[r(1,1,1)])
provref("new",[r(2,1,1)])
provref("<or=",[r(2,1,1),e(7,1,1)])
pred("element",2,[r(1,1,1),r(2,1,1)])
pred("element",2,[r(3,1,1),r(2,1,1)])
provref("new",[r(3,1,1)])
provref("<or=",[r(3,1,1),e(7,1,1)])
provref("new",[r(4,1,1)])
provref("<or=",[r(4,1,1),e(7,1,1)])
pred("youpl",4,[r(4,1,1)])
pred("proud",5,[r(1,1,1),r(4,1,1)])
pred("boy",6,[r(3,1,1)])
provref("new",[e(7,1,1)])
pred("sing",7,[e(7,1,1),r(1,1,1)])
provref("new",[e(7,1,2)])
pred("sing",7,[e(7,1,2),r(3,1,1)])

yes
```

4.3 Összegzés – GeLexi-projekt

A GeLexi-projekt arra vállalkozott, hogy implementál egy totálisan lexikalista nyelvtant, ezáltal bizonyítva, hogy az adott nyelvtan (a GASG) egy egzakt, szigorúan formalizálható elmélet, így mind a nyelvtudományban, mind a számítógépes nyelvészetben építeni lehet rá. A Prologban elkészített elemző volt az első lépés e felé a cél felé.

A program magyar és angol nyelvű lexikai egységeket tartalmaz, és képes a generatív alapfeladat végrehajtására: ellenőrizni tudja a szavak jólformáltságát és a mondatok grammatikalitását, illetve szintaktikai és szemantikai reprezentációt tud társítani első lépésként mondatokhoz, de az elméleti háttér úgy lett kidolgozva, hogy szövegek elemzése is ugyanezzel a mechanizmussal legyen lehetséges. A totális lexikalizmus értelmében nem épít frázisstruktúrát, a szórendről rangparaméterek segítségével ad számot, amellyel nem csak az egy nyelven belüli, de a nyelvek közötti különbségek is elegánsan megragadhatók. Képes továbbá gépi fordításra az elemző kétirányú használatával az ún. kopredikációs hálózaton keresztül, amely egy köztes szint szintaxis és szemantika között, ahol a nyelvek közötti

különbségek jelentős része már nem jelenik meg. Ezért nem nehezebb feladat a totálisan lexikalista megközelítés számára a nagyon eltérő szerkezetű nyelvek közötti fordítás sem.[38] Mivel a lexikai egységek morfémák és nem önálló szavak (totálisan lexikalista morfológia), az elméletnek és a programnak a nyelvek közötti eltérő szószint sem okoz problémát (pl. a magyarban gyakran képző, ami az angolban segédige, vagyis önálló szó). A megközelítés eredményességét többféle nyelvi jelenségen teszteltük sikeresen, ilyen például a műveltetés, a hangalakot nem öltő névmások esete, a vonzatos melléknév kezelése vagy a mellérendelés.

Programunk több téren is újdonságot nyújt. A legfontosabb, hogy működő szemantikai komponenst tartalmaz, és ténylegesen képes a beírt mondatokhoz modern szemantikai reprezentációt társítani. A megközelítésünk alapjául szolgáló totális lexikalizmusnak elméleti és gyakorlati előnyei is vannak. Elméleti előny a homogenitás, azaz nincs külön szintaxis és lexikon, csak lexikon van[39]. Elméleti és gyakorlati előny egyben maga a lexikalizmus, amellyel a szabadabb szórendű nyelvek (mint például a magyar is) könnyebben kezelhetők. Programunk tisztán gyakorlati előnye pedig a számítástechnikában manapság kívánatos „minimális processzálás – maximális adattár". Végül pedig a fordítás totálisan lexikalista megközelítésének előnye, hogy univerzális tud lenni a keret, amelyet használ, nem pedig nyelvspecifikus, ezért nem kell külön-külön kidolgozni minden egyes nyelvpárra a fordítás mechanizmusát. Amint a nyelvek elemzői rendelkezésünkre állnak, bármely nyelvről fordítani tudunk a másikra. Ezért is van, hogy egyidejűleg működik programunkban a magyarról angol nyelvre, illetve az angolról magyar nyelvre történő fordítás.

A GeLexi-projekt által készített elemző tehát igazolta, hogy a totálisan lexikalista megközelítés eredményes lehet, a mechanizmusok működnek, és valóban nincs szükség frázisstruktúra építésére a mondatok elemzéséhez. A mondatok szórendjéről egy sokkal egységesebb eszközzel (a rangparaméterekkel) is számot lehet adni, amely ugyanolyan követelmény, mint hogy milyen esetű argumentumot vár egy régens. Az így nyert nem hierarchikus szintaxis közvetlenül kompozicionális a szemantikával, így a szemantikai reprezentáció egyszerűbben előállítható, mint a frázisstruktúrát használó elméletek esetében.

A következő lépés annak vizsgálata, hogy a megközelítés kellő mértékben hatékony tud-e lenni, fel tudja-e venni a versenyt egyéb alkalmazásokkal. Ehhez elsődlegesen az adatbázis méretének növelése szükséges, amihez a Prolog már nem bizonyult megfelelőnek. Erre a célra alakult meg a LiLe-projekt, amely a GeLexi-projekttel párhuzamosan működött: míg az utóbbi célja továbbra is az volt, hogy kis adatbázison próbáljon ki minél változatosabb nyelvi jelenségeket, a LiLe-projekt elkezdett egy nagyobb méretű (relációs) adatbázist építeni. A következő fejezetben erről számolok be részletesebben.

---

[38] Valójában pont a nagyon eltérő nyelvek esetében térül meg az, hogy a mondatokat alaposan elemezzük, hozzájuk szemantikai reprezentációt társítunk, és a fordítást ezen (illetve egy ehhez közeli szinten) keresztül végezzük. Hasonló szerkezetű nyelvek esetében hatékonyabb gépi fordítást lehet megvalósítani, ha egyszerűen „szóról szóra" fordítunk.

[39] Egyéb homogén rendszerek is léteznek, amelyek azonban inkább a lexikont száműzik, és csak szintaktikai szabályokkal dolgoznak (pl. Prószéky et al. 2004). Azonban a nyelvészetben az utóbbi években megfigyelhető erősen lexikalista fordulat inkább a mi megközelítésünket igazolja, legalábbis elméleti szemszögből.

5 LiLe-projekt

A GeLexi-projekt által fejlesztett Prolog program arra a célra megfelelőnek bizonyult, hogy kis adatbázison megvizsgáljuk a totálisan lexikalista megközelítés életképességét. Megmutattuk, hogy a mechanizmusok működnek, a mondatokhoz képesek vagyunk morfológiai, szintaktikai és szemantikai reprezentációt társítani, illetve a magyar és az angol nyelv viszonylatában a gépi fordítást is megvalósítottuk. A következő lépés az adatbázis bővítése volt. Úgy gondoltuk azonban, hogy ha szeretnénk kilépni a „laboratóriumi méretek" közül, akkor célszerű lenne áttérni hatékonyabb adattárolásra, illetve egy modernebb programnyelvre. Így megtehetjük az első lépést afelé, hogy a totálisan lexikalista alapú programunkból egyszer – ha megfelelőnek bizonyul – akár piacképes szoftver is lehessen.

Az elkészült Prolog programnak alapvetően három olyan hátránya, hiányossága van, ami miatt célszerűnek láttuk az adatbázis bővítését más struktúrában megvalósítani. Az első az, hogy nem felhasználóbarát a felülete, főleg a kimeneti oldalon, a kiírt reprezentációk nehezen olvashatók. A második, hogy újabb jegyek felvétele a lexikonba nehézkes, nem beszélhetünk rugalmas bővíthetőségről. Egyrészt azért, mert a kódolási rendszer, amit használtunk, rendkívül helytakarékos, ezért az ember számára nehezen olvasható, ami igencsak megnehezíti az adatbázis karbantartását, javítását és bővítését (Novák 2003). Másrészt kezdetben nem gondoltuk át elég alaposan, milyen jegyekre lesz majd szükség az egyes szinteken, ami talán lehetetlen is lett volna a nyelvfüggetlenség (mint cél) miatt. Elkerülhetetlen tehát, hogy időnként új tulajdonságokat kelljen rögzíteni, ami akár az egész program módosításával is járhat. Végül a harmadik hátrány, hogy a Prolog alapvető működési elve miatt a növekvő szókészlet exponenciálisan lassuló futást eredményez.

A hátrányok kiküszöbölésére megalakult a LiLe-projekt, amelynek fő célja egy olyan adatbázis létrehozása, amely totálisan lexikalista alapokra épül, és a GeLexi-projekt új lexikona lehet. A projekt tagjai: Bódis Zoltán, Kleiber Judit, Szilágyi Éva és Viszket Anita voltak. A projekt neve a Linguistic Lexicon rövidítéséből származik, de magyarul is értelmes szót ad: a *lile* (Charadriinae – lile-félék) a futómadarak rendjébe tartozó, Magyarországon is honos költözőmadár. Mozgásával a célkitűzéseinket szimbolizálja: nem elméleti magasságokban szárnyal, hanem a gyakorlati megvalósítás talaján jár, ám ez semmivel sem csorbítja a haladásának hatékonyságát, gyorsaságát.

Megtartottuk a GASG, illetve a már elkészült implementáció legfontosabb jellemzőit: totálisan lexikalista megközelítés, egyszintűség, morféma alapúság, univerzalitásra törekvés. Néhány változtatást végeztünk, például a LiLe lexikonában a lexikai egységek és a „szabályok" is az adatbázis részét képezik, így az még jobban megvalósítja a totális lexikalizmust, mint a korábbi implementáció, ahol a szabályok (technikailag) az elemző részét képezik.

Az adatbázis tervezése, építése és feltöltése során a kiinduló célhoz képest új (közbeeső) célokat is megfogalmaztunk, amelyek két nagy terület, a kutatás és az oktatás köré csoportosulnak. Ezekről részletesebben a 5.1 pontban szólok. Az adatbázishoz két program is készült Delphi programnyelven: egy feltöltőprogram a kényelmesebb adatbevitel érdekében, illetve egy elemző a tesztelhetőség miatt[40]. Így a projekt az adatbázis létrehozásán túl a grammatikus szóalakok és mondatok elemzését és generálását is célul tűzte ki. A megvalósításban a morfológiai szintig jutottunk: a program egy beírt szóról eldönti, hogy jól formált-e, és morfológiai elemzést ad rá. Két szempontból több, mint egy

[40] A programozási munkákért köszönet illeti Lőcsei Gábort és Lucza Gábort.

hagyományos morfológiai elemző: egyrészt a szabályok kikapcsolhatók, vagyis kérhetjük a programot, hogy például a hangrendi illeszkedésre vagy a morfémák sorrendjére vonatkozó megszorításokat ne vegye figyelembe; másrészt ha a szóalak nem bizonyult jól formáltnak, kiírja azt is, mi volt ennek az oka, milyen szabályt sért. Az eredményeink a 5.2 fejezetben olvashatók, ahol kitérek az adatbázis részletesebb felépítésére is, és néhány példán keresztül bemutatom a program működését.

5.1 Célok

A LiLe-projekt fő célkitűzése egy nyelvészeti lexikon MS-SQL-adatbázis-formában való felépítése: megtervezése, megépítése és feltöltésének koordinálása. A lexikont többféle célra szántuk. A kiinduló cél a GASG implementálhatóságának bizonyítására készülő programhoz modernebb, hatékonyabb adatbázis létrehozása, amelyben a lexikai egységeket (szabad és kötött morfémákat) minden tulajdonságukkal együtt tárolhatjuk. Az adatbázis fejlesztése során az így készülő lexikont és a hozzá tartozó elemzőprogramot alkalmasnak ítéltük egyéb célokra is. Úgy véltük, hogy mind a nyelvészeti kutatómunkát, mind a nyelvtan, illetve a nyelvek oktatását is segíteni tudná.

5.1.1 Kutatási célok

A totális lexikalizmus elvének nagyobb adatbázison való kipróbálása mellett alkalmasnak ítéltük a készülő lexikont arra, hogy egy *leíró nyelvtan* szülessen, mégpedig úgy, hogy egyszerűen „kigenerálására” kerülne az adatbázisból. Ezt a leíró nyelvtant nem szabályok halmazának képzeljük, hanem az egyes, egyedileg definiált morfémákon vagy szavakon működő jelenségek halmazának, amely halmazokhoz nem is szándékozunk feltétlenül külön megnevezést, címkét rendelni. A leíró nyelvtan megalkotásához (összhangban a lexikalista szemlélettel) nem a hagyományos utat követjük tehát szabályok és kivételek felvételével. Az egyedi eseteket rögzítjük – az egyes lexikai egységek tulajdonságait, viselkedését –, amelyekből statisztikai alapon a szabályok és kivételek „generálhatók”: a nagy elemkészleten működő eljárásokat lehet szabályként megfogalmazni, míg a kisebb halmazokon működők lesznek a kivételek.

Egy másik fontos, megvalósíthatónak tűnő célunk egy *„dinamikus korpusz”* létrehozása. Azért hívjuk a korpuszt dinamikusnak, mert az adatbázis nem a *létező* (valaha létezett) alakokat tartalmazza, hanem az azokból visszaszármaztatott elemeket és szabályokat, így az adott nyelvállapot *lehetséges* szavai, kifejezései vagy akár mondatai bármikor generálhatók – a kompetenciánkat modellálja tehát. Amiben többet nyújt, mint egy hagyományos korpusz (a dinamikusságán túl), hogy tulajdonságra is kereshetünk benne. Kérhetjük például a programot arra, hogy generáljon főnévi igenevet tartalmazó mondatokat, vagy olyan szavakat, amelyekben több fonológiai váltakozás is van. Úgy véljük, hogy az így létrejövő dinamikus korpusz segítségével egyszerűen lehet majd egy-egy elmélethez példákat, illetve ellenpéldákat találni, vagy akár statisztikákat készíteni, hogy egy bizonyos tulajdonság (jegy) mennyire gyakori a nyelv lexikai egységeinek körében. (A dinamikus korpuszról részletesebben lásd Bódis et al. (2004).)

A távolabbi célok között szerepel, hogy az adatbázist más nyelvű lexikai egységekkel is feltöltsük, hiszen a struktúráját nyelvfüggetlenre terveztük. Így a lexikon használható lenne mint n-nyelvű szótár (az adatbázist mint lexikont mutatja be Kleiber (2006)).

5.1.2 Oktatási célok

Az általunk épített adatbázis a különféle kutatási célokon kívül alkalmas arra is, hogy a modern közoktatási tananyag mellé oktatási segédeszközök épüljenek rá, amelyek élményszerűbbé teszik a nyelvtan tanulását, önálló felfedezésekre adnak lehetőséget a tanulóknak, és segítik a magyar nyelv és az idegen nyelvek összehasonlítását is.

Köztudott, hogy napjainkban a magyar nyelvtan tanítása igencsak elavultnak tekinthető, továbbá hiányzik belőle minden játékosság, problémaorientáltság, semmi nem motiválja a gyerekeket arra, hogy meg akarják ismerni anyanyelvük szabályszerűségeit (Takács 2000). Úgy gondoljuk, egy olyan programmal, amivel gyakoroltatni lehet az egyes szabályokat, megvizsgálni működésüket azáltal, hogy „ki-bekapcsolgatjuk" őket, érdekesen lehetne megtanítani a diákoknak, hogyan is működik a magyar nyelv. A program természetesen nem csupán a szabályok megtanítására és szemléltetésére lenne alkalmas, hanem a nyelvi tudatosság fejlesztésére is, ami nem csak a közoktatásban tanuló diákoknak fontos, hanem talán még inkább a felsőoktatásban tanuló, később magyartanárként elhelyezkedő fiatalok számára.

Programunk nem csak nyelvtan, hanem idegen nyelv tanítására is alkalmas lehet. Mivel jelenleg magyar lexikai egységeket tartalmaz, egyelőre a magyar nyelvet lehetne rajta keresztül szemléletesen bemutatni a magyarul tanuló külföldieknek, illetve a magyart mint idegen nyelvet (hungarológiát) tanító jelenlegi és majdani oktatóknak. A felhasználó nem csupán azt tudhatja meg egy szóalakról vagy mondatról, hogy grammatikus-e, hanem azt is, hogy ha nem jól formált, milyen szabály(oka)t sért. Így könnyen és érdekesen fedezhetik fel, hogyan, milyen szabályok szerint működik a magyar nyelv, illetve mely szavak vagy kifejezések viselkednek kivételesen.

A felsőoktatásban nem csupán a hungarológia oktatására alkalmas a projektünk. Az adatbázis feltöltésének segítésére több kurzust is hirdettünk magyar szakos hallgatóknak, bevontuk őket is a munkába. A kurzusok azért voltak különösen hasznosak a hallgatók számára, mert nem azt mutatták be, hogyan működik a nyelv valamely grammatikai modellben, hanem bármilyen, jelenleg ismert grammatikai modelltől függetlenül a nyelv leírására törekedtek. Ilyen leíró jellegű kérdések közé tartozik az igék különféle vonzat-kereteinek kutatása, a szemantikai szelekciók kérdése stb. A feltöltés természetesen visszahatott az adatbázis fejlesztésére is, hiszen azáltal, hogy a hallgatók érdekesen viselkedő lexikai egységekre leltek, néhol módosítani kellett a struktúrát. Így a kurzus hasznos volt mindkét fél számára.

5.2 Eredmények

A projekt céljai közül már megvalósítottuk az adatbázis-struktúra megtervezését és kiépítését (mindegyik nyelvi szintre), továbbá egy feltöltő- és egy elemzőprogram megalkotását Delphi programnyelven, amelyek a morfológiai információkat tudják kezelni. Az adatbázist feltöltöttük párszáz lexikai egységgel, amelyek jelentős része kötött morféma (toldalékok), de tartalmaz nyílt szóosztálybeli elemeket is, névszókat és igéket egyaránt. Az egységek tulajdonságai közül a morfofonológiai és a morfológiai információkat rögzítettük. (A munka azért akadt meg, mert a GeLexi és a LiLe projektek egyesülésével jelenleg egy új, reményeink szerint még hatékonyabb, és még alaposabb adatbázis és program kidolgozásán fáradozunk.) Az adatbázist XML-formátumba is átkonvertáltuk, így a tartalmát weben keresztül is egyszerűen meg lehet jeleníteni.

5.2.1 Morfológia

Szóelemző programunk jelen állapotában a magyar főnévi és igei inflexiós morfológia és az írásban jelölt fonológiai szabályok alapján működik. A beírt szóról eldönti, hogy jól formált-e, és morfológiai elemzést ad rá. Figyelembe veszi a szófaji egyezést, a helyes morfémasorrendet és a fonológiai szabályokat. Az egyedisége abban rejlik, hogy nemcsak azt tudja megállapítani, hogy a beírt szóalak helyes-e, hanem azt is megmondja a felhasználónak, hogy mely nyelvtani szabály[41] helytelen működése vagy figyelmen kívül hagyása eredményezte a nem megfelelő szóalakot. Továbbá a szabályok kikapcsolhatók, mondhatjuk például a programnak, hogy a szófaji megszorításokat, a morfémasorrendet, vagy a hangrendi illeszkedés szabályát ne vegye figyelembe, amivel szemléltethető, hogy az adott jelenség mennyiben hat a nyelvre.

Mivel programunk kompetencia alapú, felmerül a nyelvi regiszter problémája: ha egy nyelvi elem több lehetséges formában él a nyelvben, kérdés, hogy az elemző melyeket fogadja el, és melyeket ne, illetve a generáló melyik alakot állítsa elő. Úgy gondoltuk, az a legszerencsésebb, ha nem csupán a köznyelvi regiszterbe tartozó formákat vesszük fel a lexikonba, hanem az azon kívüli gyakran előforduló alakokat is, amelyeket elfogadja ugyan a program, de stílusértéket rendel hozzájuk (dialektus, szleng stb.). Mint nyelvészek, nem vagyunk hívei a preskriptív (előíró) nyelvtannak, de ha céljaink között szerepel a magyar nyelv tanítása is a programunk segítségével, akkor meg kell különböztetnünk a köznyelvi és a nem köznyelvi alakokat. Ennél a megkülönböztetésnél nem csupán szavak különféle ejtésváltozataira kell gondolnunk (mint például *jön* vs. *gyün*), hanem tulajdonságokra is. Például a köznyelvben a *tányér* alak nem nyitótő (tárgyragos alakja *tányért*), bizonyos dialektusokban viszont nyitótőként viselkedik (*tányérat*). Annak megállítására, hogy milyen szubsztenderd alakok gyakoriak annyira a nyelvben, hogy célszerű legyen felvenni őket a lexikonba, a Magyar Nemzeti Szövegtár több, mint 187 millió szövegszavas korpuszát is felhasználjuk.

5.2.2 Technológia

A lexikonunkat relációs adatbázisban terveztük meg, ami lehetővé teszi, hogy struktúránk dinamikusan bővíthető legyen. A következő alpontban röviden ismertetem a relációs adatbázis fogalmát és legfontosabb tulajdonságait[42].

5.2.2.1. Relációs adatbázis

Adatbázisnak nevezzük az adatok és a köztük levő összefüggések (kapcsolatok) egy helyen tárolt rendszerét. A legelterjedtebb típusa a relációs adatbázis (mert egyszerű és rugalmas), amelyben az adattárolás alapjait *táblák* (adatállományok) alkotják. Egy adott adatbázis általában több táblából áll, ahol egy tábla nem más, mint a logikailag összetartozó adatok sorokból és oszlopokból álló elrendezése. A táblák sorait *rekordoknak*, oszlopait *mezőknek* nevezzük. Tehát az adatbázist alkotó *egyedek* a táblázat soraiban, az egyedtulajdonságok pedig az oszlopaiban találhatók. A relációs adatmodellben az egyedek közötti kapcsolat az adatértékeken keresztül valósul meg, vagyis az ún. *kapcsolómezők* azonosságán alapszik.

---

[41] „Szabály” alatt természetesen (összhangban a totálisan lexikalista szemlélettel) valójában elvárásokat, tulajdonságokat kell érteni, amelyek ha tudnak unifikálódni, teljesül a „szabály”.

[42] Felhasznált források: http://gisfigyelo.geocentrum.hu/informatika/kisokos_relacios_adatbazis.html és Szilágyi (2005).

A leírás az egyed–kapcsolat modellen alapul. Minden egyes entitás egy rendezett n-es, amelynek elemei valamilyen relációt alkotnak – ez egy rekord. Az n-esek elemei az attribútumok, ezek a rekord mezőit jelentik. A reláció maga a tábla, ahol minden egyes sor (rekord) egy rendezett n-es, és minden oszlop egy attribútum (Halassy 1994). Az adatbázisban (gyakorlatilag) bármennyi reláció (azaz tábla) definiálható, amelyek bármennyi rekordot tartalmazhatnak, bármennyi attribútum értékével.

Az attribútumok között kell, hogy legyen legalább egy olyan, amely minden egyed-előfordulásra különböző értéket vesz fel, azaz *azonosító* (elsődleges kulcs, primary key)[43]. Azonosítója minden egyednek van, ami nem lehet üres vagy ismeretlen. Minden egyednek csak egy azonosító tulajdonsága lehet (a redundanciával szemben), és ugyanaz a tulajdonság csak egyetlen egyednek lehet az azonosítója (az inkonzisztenciával szemben). Az olyan tulajdonságot, amely az egyik egyedben azonosító, a másikban leíró szerepű, kapcsoló szerepű tulajdonságnak (külső vagy idegen kulcs, foreign key) nevezzük. Az azonosítók az egyedek közötti átmenetek eszközei is.

A relációk egyedei közötti kapcsolatok többféleképpen állhatnak fel. Kapcsolatban lehet az egyik reláció egy egyede a másik reláció pontosan egy egyedével (1:1); az egyik reláció egy egyede a másik reláció n egyedével (1:n), illetve az egyik reláció n egyede a másik reláció m egyedével (n:m). A kapcsolat az egyes esetekben különféleképpen realizálódik: az első esetben az egyik reláció pontosan egy egyedének egy attribútuma hivatkozást tartalmaz a másik reláció pontosan egy egyedének megfelelő attribútumára; a második eset ettől abban különbözik, hogy ugyanaz az azonosító több egyednél is előfordul; a harmadik eset pedig egy újabb táblát jelent, amelynek egy rekordja tartalmaz mindkét reláció valamely egyedére hivatkozást, és emellett egyéb attribútumai is lehetnek.

A rekordok között lehetőség nyílik az adatok egy szűkebb részletét lekérni egy egyszerű lekérdező nyelv segítségével, valamely attribútum(ok) alapján (azokat, amelyek egy vagy több megadott feltételnek eleget tesznek), illetve nézeteket létrehozni. A lekért adatok bármely elérhető attribútum szerint sorba is rendezhetők vagy csoportosíthatók. A relációs adatbázis kezelésére szolgáló legelterjedtebb strukturált lekérdező nyelvet, az SQL-t (Structured Query Language), a hetvenes évek közepén dolgozták ki, és azóta is fejlesztik.

Napjainkban számtalan adatbázis-kezelő rendszer és több SQL-megvalósítás létezik. A LiLe projekt választása az Microsoft SQL Server 2000 rendszerre esett, mivel amellett, hogy ezáltal egy komplett relációsadatbázis-kezelő keret rendelkezésünkre áll, annak minden előnyével (felhasználóbarát-felület, SQL-nyelvű lekérdezések, jól rendezett adatok, nézetek, meta-adatok stb.) számos olyan szolgáltatással rendelkezik, amelyet hatékonyan ki tudunk használni céljaink eléréséhez:

- a kliens-szerver architektúrának köszönhetően az adatbázis egyidejűleg több felhasználó vagy programmodul által is elérhető (többféle kapcsolaton keresztül)
- XML támogatás: a lekérdezések eredményei XML-formátumban is kérhetők (online is), illetve XML-file-ok beolvasása az adatbázisba
- felhasználó által definiálható adattípusok, függvények, tárolt eljárások, triggerek (melyek mind az adatbázis részei)
- automatikus azonosítóosztás
- adattranszformálási, illetve -elemzési lehetőségek.

Ez a technológia tehát lehetővé teszi, hogy struktúránk dinamikusan bővíthető legyen. Egy-egy újonnan felvett tulajdonság rekordként és nem mezőként jelenik meg a táblákban, tehát ha egy új tulajdonság felvétele látszik szükségesnek, nem kell az egész struktúrát

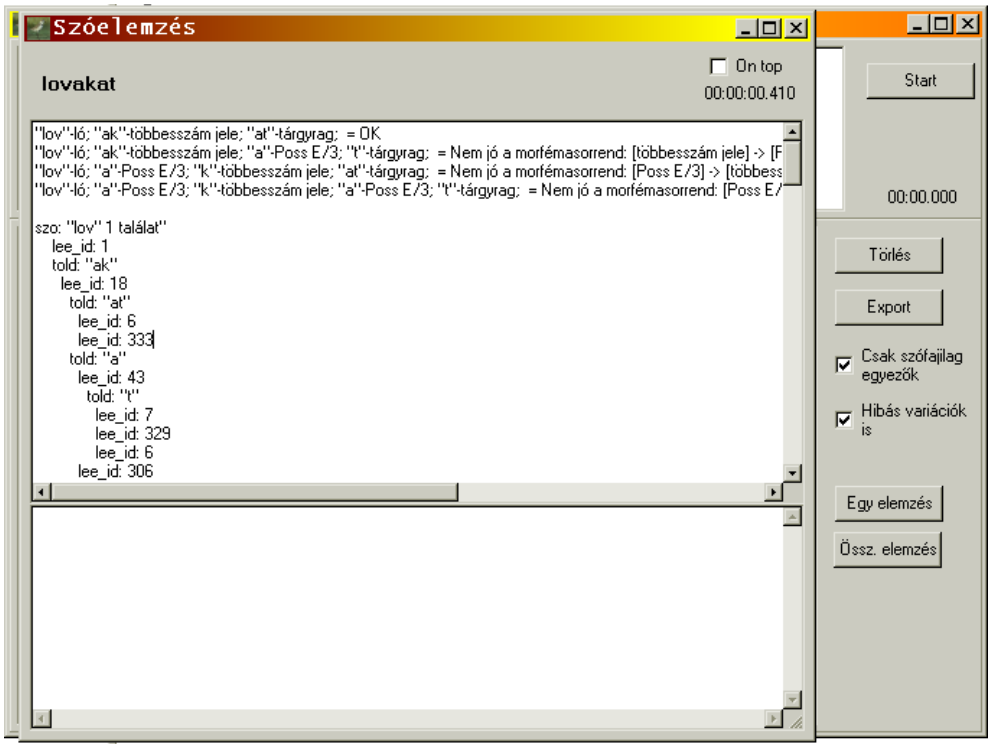
---

[43] Összetett azonosító esetén (composite key) az összetétel részei együtt nem ismétlődhetnek más egyedeknél. (Egyik rész sem lehet üres vagy ismeretlen.)

átalakítani, csupán egy új elemet felvenni az adatbázisba. Így nem jelent problémát az sem, hogy jelen fázisban az adatokkal való feltöltés még visszahathat magára az adatbázis tervezésére is, hiszen a több ezer lexikai egység felvitelének tapasztalataiból olyan következtetésekre juthatunk, amelyek alapján esetleg változtatásokat kell végrehajtanunk a struktúrán. Az adatbázis feltöltésére írtunk egy feltöltőprogramot Delphi programnyelven, ezen keresztül kerültek felvitelre az első elemek. Az adatokat XML/XSL segítségével weben keresztül is megjeleníthetjük. A lexikon méretének jelentősebb megnöveléséhez automatikus feltöltést terveztünk, esetleg a disztribúciós módszer felhasználásával.

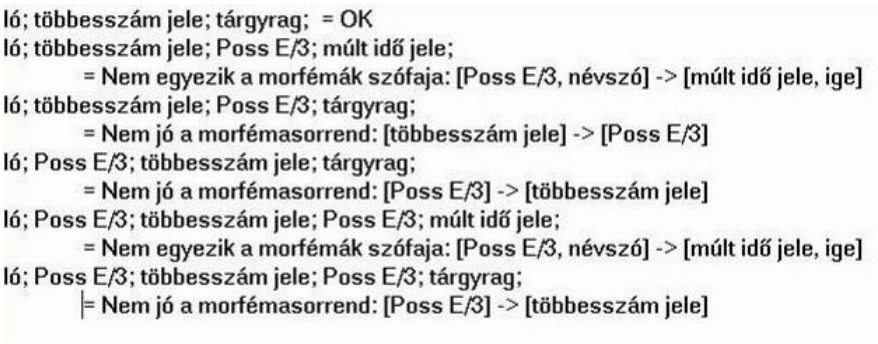
5.2.3 A program működése

A program működését egy példa segítségével szemléltetem, amely a *lovakat* szó elemzése. A 9. ábra mutatja az elemzőprogram felületét, ha a fenti szót mint bemenetet kapja meg a program. Jobb oldalon láthatók a különféle opciók: hogy kérjük-e a hibás variációkat is (hogy lássuk, milyen elvárás nem teljesült), a hibás variációk közül kérjük-e a szófajilag sem egyezőket, vagy hogy csak egy elemzést, vagy az összeset szeretnénk. Az elemzést mutató ablak felső részén látható, hogy a programnak 0,41 mp-ig tartott az elemzést előállítani és részletesen kilistázni.



9. ábra: Az elemzőprogram felülete

Ha a hibás elemzések listázását úgy kérjük, hogy nem csak a szófajilag egyezőket, akkor a program a következő kimenetet (hat elemzést) állítja elő (10. ábra):



10. ábra: A *lovakat* szóalak elemzése

Az első sorban olvasható rögtön a jó megoldás: szavunk három morfémát tartalmaz: a *ló* szótövet, a főnévi többes számot kifejező morfémát, valamint a tárgyesetet kifejező morfémát. A következő sorokban a hibás elemzések láthatók, hol a morfémasorrendre vonatkozó megszorítások sérülnek, hol a morfémák szófaja nem egyezik meg. Így az egyetlen jó megoldás az első, ahol a morfémák szófaja megegyezik, a morfémák a helyes sorrendben fordulnak elő, a *ló*-nak a megfelelő allomorfja szerepel benne, megfelel a hangrendi illeszkedés törvényének és a nyitásra vonatkozó hangtani szabályoknak is.

Ahhoz, hogy mindezeket megállapíthassuk, az adatbázisban tárolni kell a szabályokat és az egyes morfémák és allomorfok tulajdonságait. A tulajdonságokat jegyekben tároljuk. Az egyes jegyeket el is neveztük, ami nem a program működéséhez szükséges, hanem a szabályok megnevezéséhez, és így a felhasználó informálásához. A következő leírásban az egyes jegyek megnevezését adjuk meg, de a programban valójában csak a jegy azonosítójára hivatkozunk, a jegy nyelvészeti tartalma (szófajok, fonológiai tulajdonságok stb.) a program működése szempontjából irreleváns. Ezt azért fontos megjegyezni, mert az egyes jegyek elnevezésénél nem törekedtünk arra, hogy valamely konkrét grammatikai (a példánkban: fonológiai) modell terminológiájához illeszkedjünk. A terminológia végső kialakítása az adatbázisunkat alapul vevő tananyagok része lehet.

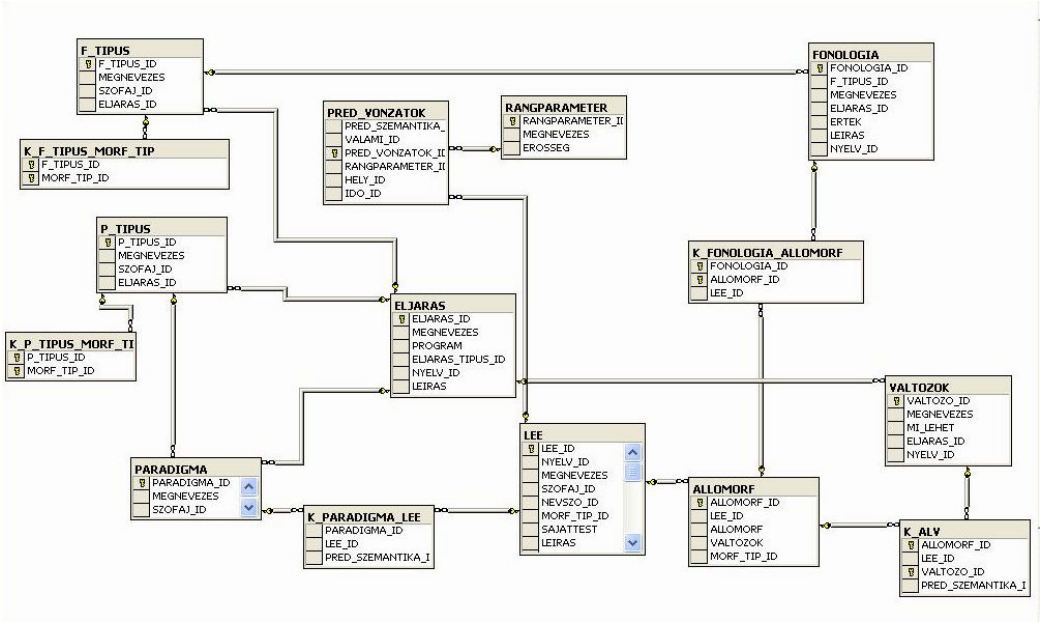
A *ló* szó tulajdonságai közé tartozik, hogy főnévi tő, ami *ló* és *lov* alakokban fordulhat elő; továbbá mély hangrendű. A *lov* allomorf (a megjelölt toldalékok esetében) kötőhangot kíván meg; nyitótő, ami befolyásolja az őt követő kötőhangot; valamint v-vel bővülő tő. A többes szám jele (*-k*) tulajdonságai közé tartozik, hogy névszói toldalék, ami közepes erősséggel akar a szótőhöz közel kerülni (rangparaméterek); kiváltja a v-vel bővülést; és a következő morfémára vonatkozóan nyitótőként viselkedik. A lehetséges allomorfjai közül az *-ak* mély hangrendű és a nyitótövekhez társul. A tárgyeset ragjának a példánkban releváns tulajdonságai: névszói toldalék; kis erősséggel akar közel kerülni a tőhöz; az *-at* allomorfja pedig mély hangrendű és nyitó „tövekhez” (vagyis megelőző morfémákhoz) társul. A jegyek listájában megadjuk, hogy mely jegyek párosíthatóak, illetve azt is, hogy melyek zárják ki egymást, így egyfajta unifikációs módszerrel ellenőrizhető, hogy a szóalakban szereplő morfok jegyei illeszthetőek-e.

5.2.4 Az adatbázis felépítése

A 11. ábra az adatbázis-struktúra egy részletét – a morfofonológiával kapcsolatos táblákat – mutatja[44]. A központi tábla a LEE, amely magukat a lexikai egységeket tartalmazza. Ehhez

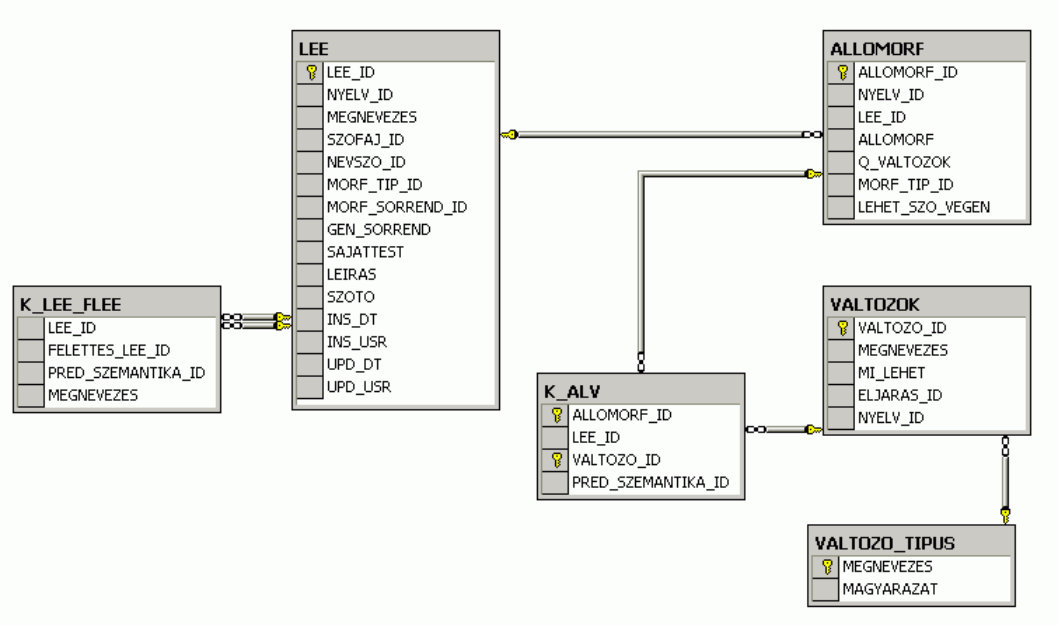
[44] Korábban már említettem, hogy az egész adatbázis-struktúrát létrehoztuk (szintaxissal és szemantikával kapcsolatos táblákat is), a dolgozatban azonban csak a morfofonológiai komponenst mutatom be. Ennek oka, hogy első lépésként a morfofonológiát dolgoztuk ki részletesen, ezzel kapcsolatos tulajdonságokat rögzítettünk, így ezeket a táblákat tudtuk „tesztelni” a programmal, ezekről tudjuk, hogy működő rendszert eredményeznek.

kapcsolódik az ALLOMORF tábla, amelyet a lexikai egységek sajáttestéből állít elő a program, a bennük szereplő változók feloldásával. Láthatók továbbá a különböző készlettáblák (változók, fonológiai tulajdonságok, azok lehetséges értékei stb.), és a kapcsolótáblák (a 'K_' kezdetűek), amelyek az egyes táblák elemeinek összerendeléséről gondoskodnak. A táblastruktúra legfontosabb pontjait (allomorfok előállítása, fonológiai tulajdonságok rendszere) részletesebben is bemutatom.



11. ábra: A morfofonológia táblastruktúrája

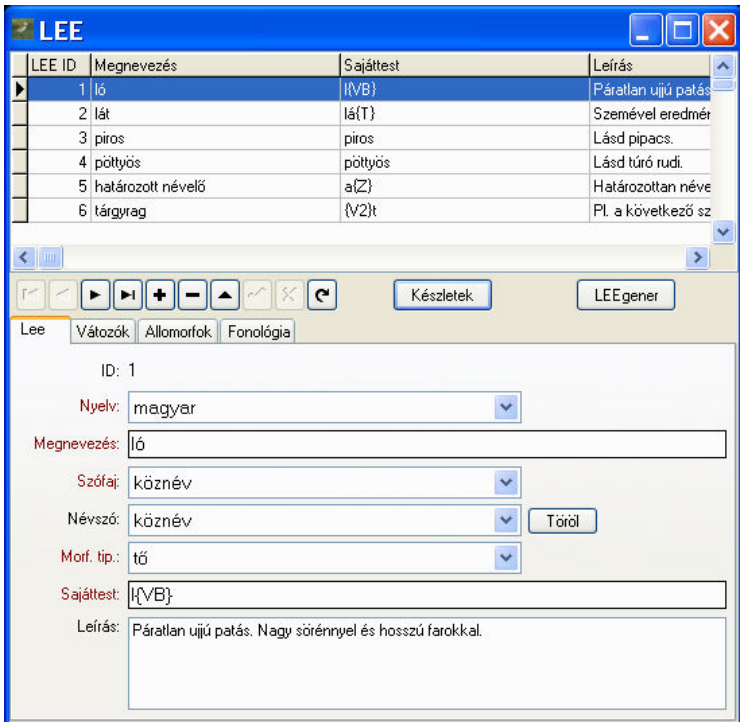
A 12. ábra mutatja az adatbázis-struktúra azon részét, amely a lexikai egységek felszíni megvalósulásért felelős.



12. ábra: LEE, változók, allomorfok

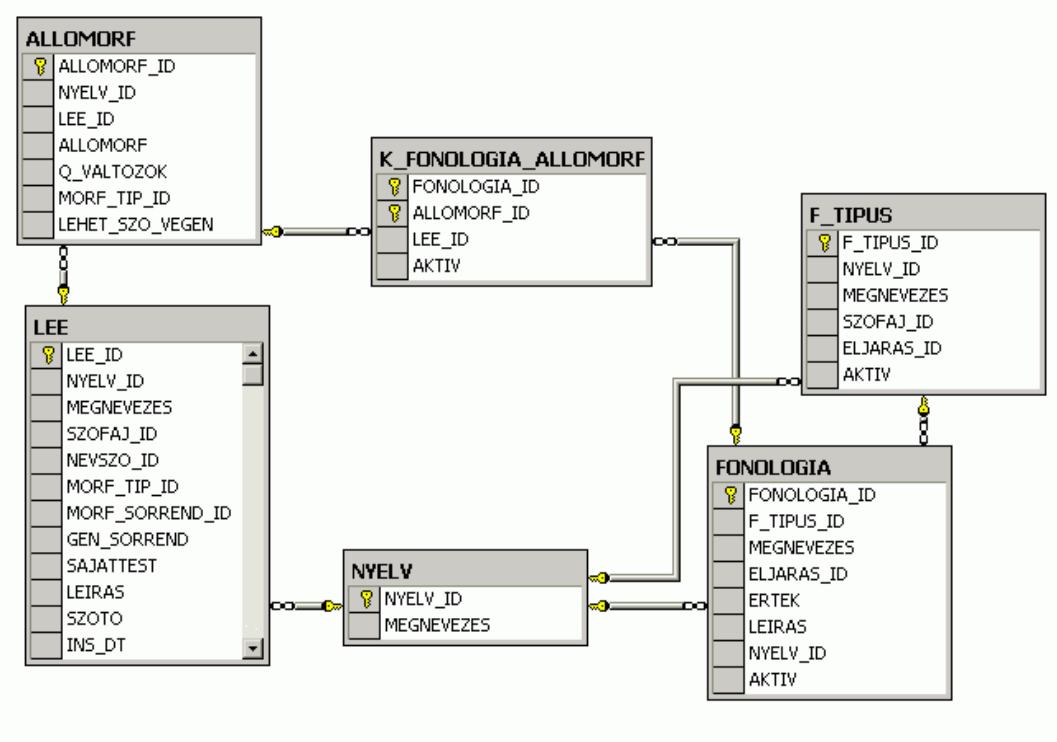
A LEE táblában rögzítjük a lexikai egység tulajdonságait: a nyelvi azonosítóját, a megnevezését, az egység szófaját (egy külön táblázat tartalmazza a szófajok hierarchiáját, például hogy egy köznév az főnév, névszó, és szó), a morfématípust (tő, képző, jel, rag vagy

összetett típus), a morfémasorrendet (rangparaméterekkel), a lexikai egység sajáttestét (változókkal) és egy leírást. Tároljuk továbbá a változók lehetséges értékeit (VALTOZO tábla, illetve a típusok összefogására VALTOZO_TIPUS tábla), ami alapján a programmal kigeneráltattuk az allomorfokat, majd eltároltuk őket egy táblában (ALLOMORF tábla); így hatékonyabbnak bizonyult, mint minden programfutáskor kiszámoltatni az allomorfokat. A K_ALV táblában van lehetőség annak rögzítésére, ha egy változó bizonyos feloldása valamilyen szemantikai értékhez köthető (pl. *mezője* vagy *mezeje*). A LEE tábla tehát alapvetően morfémákat tartalmaz, amelyeknek ha több allomorfja van, azok a sajáttestben található változó alapján számolódnak ki. Néha azonban két alak annyira eltér (pl. *jön* és *gyere*, *-i* és *-ja*), hogy nehézkes lenne egyetlen sajáttestet hozzájuk rendelni, ekkor külön lexikai egységként tároljuk őket, és a K_LEE_FLEE táblában mondjuk meg, hogy valójában egy morfémához tartoznak. A 13. ábra a 'ló' lexikai egység LEE táblában tárolt tulajdonságait mutatja a feltöltőprogramon keresztül.



13. ábra: A feltöltőprogram felülete

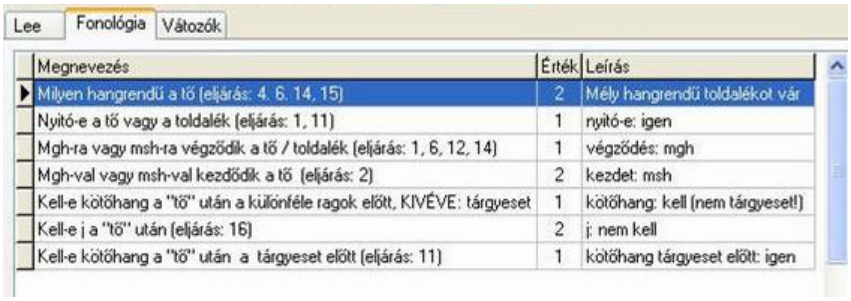
A legalapvetőbb tulajdonságokon kívül minden egyéb információt kapcsolótáblákon keresztül rögzítünk az adott lexikai egységhez. Ettől lesz a struktúra könnyen változtatható, javítható és bővíthető. Ez jól érzékeltethető a fonológiával kapcsolatos táblák kapcsán (14. ábra). Szerepel az ábrán a korábban már bemutatott LEE tábla is, de látható, hogy a fonológiai tulajdonságokat valójában nem a lexikai egységekhez (LEE tábla elemeihez), hanem az allomorfokhoz rögzítettük, hiszen egy morféma allomorfjainak különböző fonológiai tulajdonságai lehetnek (pl. a *-ban* hátul képzett, de a *-ben* elöl képzett). (A szintaktikai és a szemantikai tulajdonságok a LEE-hez kapcsolódnak.)



14. ábra: Fonológia

Az allomorfok fonológiai tulajdonságait a K_FONOLOGIA_ALLOMORF nevű kapcsolótábla tartalmazza, amely tehát a FONOLOGIA és az ALLOMORF táblákat köti össze. A FONOLOGIA tábla a különféle fonológiai tulajdonságtípusok (F_TIPUS) lehetséges értékeit tartalmazza. Ilyen típus például, hogy nyitó-e az adott elem (lehetséges értékei igen vagy nem), kiváltja-e a rövidülést egy toldalék (igen, nem), vagy milyen hanggal kezdődik egy szó (releváns: magánhangzó vagy mássalhangzó), illetve milyen hangra végződik (mássalhangzó esetében számít, hogy pontosan mi). A kapcsolótáblában kerül rögzítésre tehát, hogy egy adott allomorfra a releváns jegyek közül melyik teljesül.

A fonológiai tulajdonságok tehát nem mezőként, hanem rekordként jelennek meg az adatbázisban, vagyis nem kell őket előre (az adatbázis létrehozásakor) lerögzíteni. Ez azért is nagyon hasznos, mert egy nyelv esetében még talán látható előre, milyen tulajdonságokra lenne szükség, de ha az univerzalitás is cél, vagyis hogy bármely nyelv adataival feltölthető legyen az adatbázis, akkor nem lehet előre minden releváns jegyet felvinni, hiszen ezek köre nyelvenként nagyon különbözhet. Mindenképpen rugalmasabb tehát egy olyan adatbázis, ahol egy táblában rögzítjük a relevánsnak tartott (fonológiai vagy egyéb) tulajdonságokat (bármikor vehetünk fel újabbakat), egy másikban ezek lehetséges értékeit, majd ezek közül a megfelelőt összekapcsoljuk a lexikai egységgel. Például egy fonológiai tulajdonság a hangrend, amelynek két értéke lehet: magas vagy mély, a *kutya* szóhoz a mélyet rendeljük hozzá, a *kék* szóhoz a magasat, a *farmer* szóhoz pedig mindkettőt (nyelvtechnológiai szempontból természetesen az számít, hogy milyen hangrendű toldalékot vár, tehát pl. a *híd* mély hangrendűnek számít). A 15. ábra a *ló* lexikai egység releváns fonológiai tulajdonságait mutatja.



15. ábra: Fonológiai tulajdonságok

A „szabályok” (korábban már említett) „ki-bekapcsolhatósága” az adatbázisban úgy jelenik meg, mint a tulajdonságok egy mezője (hogy aktívak-e). Így tehát bármelyik tulajdonságot (mint szabályt, például hangrendi illeszkedés) ki lehet kapcsolni, ami azt eredményezi, hogy az elemző helytelen alakokat is el fog fogadni, illetve a generáló ugyanezeket az alakokat elő fogja állítani. Ennek egyrészt az oktatásban lehet haszna (például meg lehet mutatni, milyen formában létezne egy adott morfémakapcsolat, ha nem lenne például hangrendi illeszkedés, vagy nem lennének nyitótövek), másrészt pedig a kutatókat is segítheti (tesztelni lehet egy adott szabály létjogosultságát a nyelvben).

5.3 Összegzés – LiLe-projekt

A LiLe-projekt fő célkitűzése egy nyelvészeti lexikon MS-SQL-adatbázis-formában való felépítése (kifejlesztése és feltöltésének koordinálása). A kidolgozott adatbázis-struktúra legfontosabb tulajdonsága, hogy a grammatika különböző szintjein létező szabályokat a morfémákkal azonos módon tárolja, így valósítva meg a legteljesebb lexikalizmust (a szabályok is a lexikon részét képezik). A projektünk a grammatikus szóalakok és mondatok elemzésére és generálására vállalkozik.

Számos számítógépes nyelvészeti projekt dolgozik azon, hogy valamilyen alkalmazást, szótárt fejlesszen, elméleti (kutatási, pl. korpuszok, szótárak) vagy gyakorlati (pl. helyesírás-ellenőrzés, fordítás) célokra. A LiLe projekt újítása ezekhez a munkákhoz képest több szinten megfogalmazható. Egyrészt a célkitűzésekben, mint például egy dinamikus korpusz megalkotása, vagy az oktatás támogatása. Másrészt a felhasznált elméleti keretben, amely egy totálisan lexikalista grammatika (a GASG), ahol az elemek és a szabályok egységesen kezelhetők, és amely morfémaszinten kezeli a szintaxist és a szemantikát is. Harmadrészt az alkalmazott technológiában, mivel a lexikont SQL-adatbázis formájában tároljuk. Célunk, hogy a különböző elemek és összefüggéseik egyféleképpen legyenek tárolhatóak, és hogy az adatbázis XML-kompatibilis legyen, többek között azért, hogy weben keresztül egyszerűen meg lehessen jeleníteni. Végül pedig a működésbeli újdonság, hogy nem csupán az elemek, hanem a szabályok is az adatbázis részét képezik (rekordként vesszük fel mindegyiket).

A LiLe-projekt lexikona számos pozitív tulajdonsággal rendelkezik. Több információt tartalmaz a lexikai egységekről, mint akármilyen másik szótár, ami nagyon hasznos bármilyen adatbázisnál. A struktúra nyelvfüggetlen, így egyszer kell csak kidolgozni, később (amikor más nyelvek kerülnek bevonásra) csak újabb rekordokat (lexikai egységeket és „szabályokat”) kell felvinni. Végül pedig a szabályok ki-bekapcsolhatók, ami nem csupán az oktatásban hasznos, hanem lehetővé teszi, hogy ne csak a tökéletesen grammatikus alakokat ismerje fel a program (Prószéky (2005) érvel például az ilyen rendszerek szükségessége mellett).

A LiLe-projekt eredetileg arra a célra alakult, hogy a GeLexi-projekt eredményeit egy modernebb fejlesztőkörnyezetbe vigye át, hogy tesztelhessük a totálisan lexikalista megközelítés hatékonyságát nagyobb adatbázison. Ezt a célt csak részben érte el: létrehoztuk

ugyan az adatbázist, amely a feltöltött adatokon jól működik, viszont nem növeltük jelentősen a lexikon méretét (továbbra is párszáz lexikai egységet tartalmaz, csak nem egészen ugyanazokat), illetve a megvalósításban csak a morfofonológia szintjéig jutottunk, a program nem képes tehát mondatok elemzésére és szemantikai reprezentáció hozzárendelésére. A munka azért szakadt meg, mert a két projekt egyesülésével új adatbázison dolgozunk; kicsit más alapokon és kicsit más célokat helyezve előtérbe (erről szól a következő fejezet). Az új projekt keretében azonban maximálisan hasznosítani tudjuk a LiLe-projekt eredményeit, hiszen az alapokat kidolgoztuk, és megmutattuk, hogy az elmélet egy modernebb struktúrában is implementálható. Továbbá a morfofonológiai komponensnek nem csupán a táblastruktúráját, hanem a bevitt adatokat (szófajok, fonológiai tulajdonságok stb.) is be tudtuk építeni az új alkalmazásba, hiszen ezek működőképességét már bizonyítottuk. Végül pedig a LiLe-projekt által megfogalmazott célokat továbbra is szem előtt tartva abban bízunk, hogy programunk nem „csupán” elemzésre lesz képes, hanem hatékony eszköz tud lenni az oktatás vagy a kutatás támogatására is.

6 Jövőkép: ReALIS-projekt

A GeLexi-projekt megmutatta, hogy a totálisan lexikalista megközelítés alkalmazható a számítógépes nyelvészetben, eldönthető egy mondatról annak grammatikus volta csupán a lexikonban tárolt jegyek unifikációja alapján. Az elmélet implementációjaként készített Prolog program az elemzés kimeneteként különféle reprezentációkat tud társítani a jól formált mondatokhoz, amelyek közül a legfontosabb egy modern szemantikai reprezentáció. Végül pedig megmutattuk, hogy más nyelv lexikai egységeivel is feltölthető az adatbázis; a program angol nyelvű mondatokat is képes elemezni, illetve a két nyelv között a gépi fordítást is megvalósítja, szintén totálisan lexikalista alapokon.

A LiLe-projekt kiinduló célja az volt, hogy kidolgozzon egy modernebb struktúrát az implementáció számára, alapvetően egy olyan adatbázist, amely dinamikusan bővíthető, és felhasználóbarát felülettel rendelkezik (a Prolog programra ezek nem teljesültek); továbbá hogy feltöltse azt olyan mennyiségű adattal (lexikai egységgel), hogy a totálisan lexikalista megközelítés hatékonyságáról is nyilatkozni tudjunk. Az adatbázist létrehoztuk, viszont csak a morfofonológiai komponens került részletes kidolgozásra (az azzal kapcsolatos rekordokat rögzítettük), és az adatbázis méretét sem sikerült lényegesen megnövelnünk. A struktúra azonban működőképesnek bizonyult, amit egy morfológiai elemzést végző program is bizonyít.

2006-ban úgy döntöttünk, hogy a két projektet egyesítjük, és felhasználva az eredményeiket, létrehozunk egy új, nagyobb méretű és hatékonyabban működő adatbázist[45]. Az új projekt kiinduló célja továbbra is a totális lexikalizmus működőképességének igazolása elméletben és gyakorlatban. Az új rendszer struktúrájában (technológiájában) a LiLe adatbázisához hasonlít, a lényeges különbség a szemantikai reprezentációban lesz: míg a korábbi implementációk az LDRT-t tekintették a reprezentáció alapjának, az új projekt szemantikai komponensét egy még alaposabb interpretációs rendszer – a ReALIS – alkotja. A projekt implementációért felelős részének tagjai Alberti Gábor, Szilágyi Éva és Kleiber Judit, de az elméleti kutatásban Komlósi Kata, Ohnmacht Magdolna, Anne Tamm és Viszket Anita is részt vett.

A ReALIS-projekt elméleti célja tehát továbbra is a totális lexikalizmus legitimálása, viszont a hatékonyabb fejlesztőkörnyezetnek és az alaposabb szemantikai rendszernek köszönhetően bízunk abban, hogy gyakorlati célokat is meg tudunk majd valósítani. Ezek közül a legfontosabb egy olyan (megfelelően hatékony) elemző program létrehozása, amely első lépésként fókuszos mondatok kezelésére lenne képes (hozzájuk szintaktikai és szemantikai reprezentációt társítana); majd egyéb nyelvi jelenségeket tartalmazó mondatokat is elemezni tudna (a GeLexi-projekt e téren elért eredményeit felhasználva). Következő lépésként más nyelvek lexikai egységeivel is feltöltenénk az adatbázist (amelyet a nyelvfüggetlenséget szem előtt tartva terveztünk meg), és végső célként az intelligens gépi fordítást szeretnénk megvalósítani, először kis adatbázison, majd – ha úgy tűnik, hogy a mechanizmusok működnek – tesztelnénk a megközelítés hatékonyságát nagyobb szóállományon.

Ebben a fejezetben először röviden ismertetem a projekt által használt szemantikai elméletet (a ReALIS-t), ezt követően kitérek a legfontosabb elméleti változtatásokra a korábbi munkákhoz képest, majd az adatbázis felépítéséről szólok néhány szót, végül bemutatom a fókusz kezelésére tett javaslatunkat.

---

[45] A kutatásokat (és részben a dolgozat megírását is) a K60595 számú OTKA pályázat támogatja.

6.1 ReALIS

A ReALIS (Reciprocal and Lifelong Interpretation System, Alberti 2005a-b, Alberti et al. 2007b) egy *reprezentacionalista*, *dinamikus* szemantikai rendszer. A világ és a nyelv közé tehát felvesz egy köztes szintet, ahol a tudás ábrázolódik[46], és a mondatok egyik alapvető tulajdonságának (a jelentésük egyik komponensének) azok információállapot-változtató képességét tartja. (A fogalmak részletesebb ismertetése a 4.1.1 pontban olvasható).

A ReALIS egy négy komponensből álló rendszer, amely így világmodellt, interpretálói tudatmodellt, valamint statikus és dinamikus interpretációt is kínál. A következő definíció Alberti et al. (2007b)-ből származik (a részletes definíció Alberti (2005a)-ban olvasható):

> **Definíció:** Az $\Re = \langle W, W, Dyn, Tru \rangle$ *ReALIS-keret* felépítése
> a. $W = \langle U, T, I, M, \Omega \rangle$: *külső / igazi* világ, avagy *„az orákulum világa”*
> b. $W(i,t) = W^i_t = \langle U^i_t, Con^i_t, Ide^i_t, Acc^{\Lambda i}_t, \kappa^i_t, \alpha^i_t \rangle$: az i interpretáló *belső világa / információállapota* a t pillanatban
> c. $Dyn: M \times W \rightarrow W$: *dinamikus interpretáció*
> d. $Tru(m) = W[m(0), \Theta]$: *általánosított igazságértékelés / statikus interpretáció*

Az első komponens a *külső („igazi”) világ*, amely a létező entitásokat és a köztük ténylegesen létesülő relációkat (Ω) tartalmazza (akárcsak a hagyományos predikátum-logikában). Az univerzum (U) fontos részhalmaza az időintervallumok halmaza (T), az interpretálók halmaza (I) és az interpretálókat érő (nyelvi és nem nyelvi) impulzusok halmaza (M).

A második komponens egy parciális függvény, amely folyamatosan változó („felfrissülő”) *információállapotokat* társít minden interpretálóhoz. Ezek az állapotok egyben belső világoknak is tekinthetők, ahol egy világ univerzumában (U^i_t) található referensek egyben „belső entitások” is (pl. a mesék szereplői). A referensek között három reláció definiálható: a Con reláció rendezi őket a DRT-ben megszokott kondíciósorokba, az Ide reláció azonosságot határoz meg közöttük, az Acc reláció pedig elérhetőségi hierarchiába szervezi őket. Az elérhetőség szerint egyenértékű referensek osztályát *világocskának* nevezzük; nagyjából ez felel meg egy-egy doboznak. A világocskákhoz címkék rendelhetők (hiedelem, vágy stb.), a részbenrendezett struktúra „gyökere” pedig az adott interpretáló számára valóságnak gondolt alapvilág. A κ^i_t komponens öt kurzort foglal magába, és azt mutatja meg, hogy az adott interpretáló figyelme éppen milyen idő-, tér-, topik- és egyéb referensekre irányul. Végül az α^i_t parciális függvény a gyökérvilágocskabeli belső entitások és a külső entitások között teremt folyamatosan épülő kapcsolatot: *horgonyként* szolgál közöttük. A létező entitások (és predikátumok) referenseihez külső világbeli referenseket kapcsol (a nemlétezőkhöz semmit), de ez a horgonyzás lehet téves, és változhat is, ahogy az interpretáló egyre többet tud meg a világról (vagy esetleg felejt).

A harmadik komponens a *dinamikus interpretációt* megvalósító parciális függvény, amely az *információállapot-változásért* felelős. Ez általában növekvő, amikor új entitásokat fedezünk fel, vagy új relációkat építünk ki a referensek között; de lehet csökkenő is, amikor elfelejtünk valamit. Ha az interpretáló valósnak gondolja az új információt, akkor az alapvilágába rakja bele, ha viszont álmodik, könyvet olvas, vagy egy pletykát hall, aminek

---

[46] Bár a reprezentacionalizmus „definíció szerint” ezt jelenti; a ReALIS által feltételezett „köztes” szint (a reprezentáció) tekinthető a világ részének, így gyakorlatilag nincs szükség egy extra szint bevezetésére (ami több nyelvész szerint problematikus).

nincs meggyőződve az igazságtartalmáról, akkor olyan világocskába kerülnek az új elemek, amelyhez a megfelelő címke rendelődik.

Végül a rendszer negyedik komponense a *statikus interpretáció* vagy *általánosított igazságértékelés*. Az igazságértékelés azt jelenti, hogy egy mondathoz igazságértéket rendelünk, amihez vagy a külső világot, vagy valamely belső világocskát kell megnézni, hogy szerepel-e benne az adott állítás. Az általánosításra azért van szükség, mert bizonyos mondatokat nem lehet a hagyományos módon kiértékelni (igaz vagy hamis), mégis mondhatunk valamit róluk, például hogy egy ígéret őszinte-e, vagy hogy egy parancsot követnek-e. Ha például a beszélő azt mondja, hogy „Mari valószínűleg otthon van", akkor lehet, hogy jól gondolja, de lehet, hogy nem (összehasonlítva a külső világ tényeivel), az állítás lehet igaz abban az értelemben, hogy a beszélő hiszi, amit mond (az egyik 'likely' címkéjű világocskájában szerepel az az állítás, hogy „Mari otthon van"), de lehet hamis is, illetve lehet félrevezető, ha a beszélő ténylegesen tudja, hogy Mari otthon van, stb.

A rendszer nevéből két alapvető jellemzője is kiolvasható. Az egyik a *kölcsönösség* (reciprocal), vagyis hogy a beszélő tudja (vagy legalábbis azt hiszi, hogy tudja), hogy a hallgató mit tud (hisz, gondol), és tekintetbe is veszi, amikor a hallgatóhoz beszél. A beszélgetőpartnerek tehát kölcsönösen modellálják egymást, és nem egyszerűen egy szimmetrikus közös tudás alapján utalnak entitásokra vagy eseményekre. Például ha a beszélő több holland lánnyal is találkozott előző nap, de a hallgatóról úgy tudja, hogy ő csak eggyel, mondhatja, hogy „a tegnapi holland lány". Ezt nem tehetné meg a hagyományos logikai ábrázolás szerint, hiszen – ha csak a saját tudását nézzük mint világmodellt – a deskripciónak nincs jelölete (mert egynél több entitásra igaz a benne szereplő állítás). Ugyanígy szükség van egymás tudásának modellálására a „másik oldalról": a fenti kifejezést a hallgató akkor is értelmezni tudja, ha ő is több holland lánnyal találkozott előző nap, de tudja, hogy a beszélő úgy tudja róla, hogy csak eggyel.

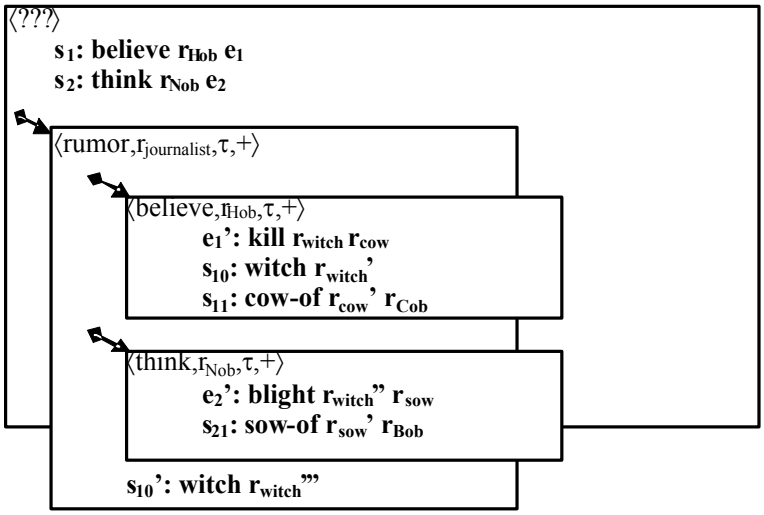
A ReALIS másik alapvető tulajdonsága pedig az LDRT-ben már bevezetett *élethosszig* tartó építkezés (lifelong), vagyis hogy a DRS-ek belső (mentális) struktúrák, amelyek az interpretáló információállapotát a születése pillanatától reprezentálják. Amikor megszületik, a hozzá tartozó „doboz" gyakorlatilag üres, majd az évek során belekerül minden tudás, amit megszerez, illetve időnként törlődik belőle, amit elfelejt, vagy átkerülnek referensek vagy állítások egyik világocskából a másikba (pl. amikor egy vágy beteljesül, vagy egy pletykáról kiderül annak igazságtartalma), esetleg változik a horgonyzás (kiderül, hogy Péter felesége nem is Mari, hanem Juli), stb.

Az elméletben kognitivista felfogás is érvényesül, az ábrázolásba az „én" és a pragmatika is bevonódik, de szigorúan formalizáltan (ami az eddigi elméletekből hiányzott).

Az elmélet számos problémára megoldást nyújt, amelyek a legtöbb szemantikaelmélet számára nehézséget okoznak. Ezek közül egyet emelek ki, a már említett cikkekben (Alberti 2005a-b, Kleiber 2005, vagy magyarul Alberti et al. 2007b) további példák is találhatók. Az itt röviden felvázolt probléma az *intenzionális azonosság* (intensional identity) kérdése, vagyis hogy a különféle lehetséges világokban lévő referensek között is létesíthető azonossági reláció (ami a hagyományos logikai értékelés számára nem megoldható). Az egyik leghíresebb példa erre a jelenségre az ún. „Hob-Nob mondatok" (92).

(92) Hob azt hiszi, hogy egy boszorkány megölte Cob tehenét, Nob pedig úgy véli, hogy *az* mételyezte meg Bob emséjét.[47]

A fenti mondatokat az is mondhatja, aki nem hisz a boszorkányok létezésében, például egy újságíró, aki az eseményről tudósít. A hagyományos igazságfeltételes (statikus) szemantika számára azonban csak akkor kezelhető a mondat (csak úgy lehet kötve a *she* névmás által bevezetett változó), ha a beszélő világában is létezik boszorkány. A ReALIS-ben azonban elegendő, ha a beszélő alapvilágában (vagy valamely beágyazott világában, ezért a három kérdőjel a címke helyén a legkülső „dobozban") van egy „rumor" címkéjű világ, és ebben a világban létezik a boszorkány. A beszélő releváns világocskáit a 16. ábra mutatja (Alberti (2005b) alapján):



16. ábra: A Hob-Nob mondatok elemzése

Mivel Hob és Nob hiedelemvilágai nem a valós világba vannak beágyazva, ezért a beszélőnek nem kell elköteleznie magát a boszorkány létezése mellett; elegendő, ha a pletyka világában van egy r_{witch}''' entitás, akiről tudjuk, hogy boszorkány, és amelyet Hob és Nob hiedelemvilágaiban azonosíthatunk az ott bevezetett boszorkányokkal ($r_{witch}' = r_{witch}'''$ és $r_{witch}'' = r_{witch}'''$); így lesz a Nob emséjét megmételyező boszorkány azonos azzal, aki Cob tehenét megölte.

Az ábrázolási mód komplexitását a 2. sz. mellékletben található elemzés jól mutatja, ahol egy befektetési szöveghez készítettem el annak ReALIS-féle „doboz"-struktúráját, alapvetően a belső világok hierarchiájára koncentrálva.

6.2 Újítások

A korábbiakhoz képest több ponton is másként képzeljük az új rendszert. A legfontosabb különbség, hogy a szemantikai reprezentációt a ReALIS alapján fogjuk megvalósítani. Ez a gyakorlatban annyit jelent, hogy még részletesebb jelentéstani ábrázolást rendelünk a mondatokhoz, amely így számot ad többek között az aspektusról és a retorikai viszonyokról

[47] Az eredeti mondat így hangzik: Hob believes that a witch has killed Cob's cow and Nob thinks that *she* has blighted Bob's sow. (A fordítás Alberti Gábortól származik.)

is[48]. Továbbá a lexikont nem csupán nyelvészeti, hanem ontológiai adatbázisként is használnánk, valamiféle világról való (kulturális/enciklopédikus) tudást is tárolnánk.

Annak érdekében, hogy a rendszer robusztus lehessen, tervezünk hiányos vagy agrammatikus mondatokhoz is szemantikai reprezentációt társítani, mégpedig az RMRS-hez hasonló formában (3.2.2.2 pont). Ahelyett, hogy *szereti(e1,t1,r1,r2)* (az *e1* esemény az, hogy egy *t1* időpillanatban *r1* szereti *r2*-t), a rendszer információdarabkákra bontaná az állítást, olyasmi kimenetet adna, hogy *pred(e1): szereti, time(e1): t1, arg1(e1): r1, arg2(e1): r2*. Így abban az esetben is megtudnánk valamit a mondat jelentéséről, ha valamelyik alapvető rész hiányozna.

Végül pedig fontos változtatás a korábbi implementációkhoz képest, hogy az eddig csupán a szórend kezelésében alkalmazott rangparamétereket kiterjesztjük más funkciókra is: bármely tulajdonság vagy elvárás mellé rögzíthető annak erőssége, így egy optimalitás-elmélethez hasonló megközelítést kapunk. Mivel a modell morfémaszinten lévő lexikai egységeket tárol, így az unifikálandó jegyek és követelmények is morfémákhoz kötődnek; és ezzel a módszerrel „kiszámolható", hogy az egyes szavak (vagy kifejezések) milyen tulajdonságokkal és elvárásokkal rendelkeznek. Például a *keres* szó gyenge ranggal (pl. 3-as) jelen idő, kijelentő mód, egyes szám, harmadik személy és határozatlan ragozás; ha nincs ellentmondó követelmény, ezek lesznek a jegyei, illetve ilyen tulajdonságú argumentumokat fog keresni. Ha azonban kap egy *-né* toldalékot, amely erősebb ranggal (pl. 2-es) állítja magáról, hogy a tulajdonságai: jelen idő, feltételes mód, egyes szám, harmadik személy és határozott ragozás; akkor – ha nincs nála erősebb rangú neki ellentmondó követelmény – ezek lesznek a szó jegyei. Viszont ha egy harmadik elem is „beleszól", amely erősebb ranggal állítja valamelyik tulajdonságot, akkor az fog „győzni"; például a *-k* személyrag a legerősebb (1-es) ranggal azt állítja magáról, hogy egyes szám első személyű és alanyi ragozású (vagyis a *-né* két jegyét is felülírja). Főnevek esetében például a *fiú* lexikai egység gyenge ranggal állítja magáról, hogy egyes számú és alanyesetű, viszont erős ranggal állítja, hogy harmadik személyű; az *engem* szó tulajdonságai (egyes szám, első személy, tárgyeset) viszont erős ranggal szerepelnek az adatbázisban, vagyis nem írhatók felül. A rangparamétereknek ez a kiterjesztése nem csupán a szón belül alkalmazható: például az angol nyelv esetében azt mondhatjuk, hogy egy ige magáról nagyon gyenge ranggal azt állítja, hogy jelen idő, felszólító mód, egyes szám, második személy; de ezt sok elem felülírhatja, szón belül például a múlt idő jele vagy az *-s* rag, szón kívül például egy alany vagy valamilyen segédige.

A megvalósítást ezúttal a szintaxisnál (illetve vele párhuzamosan a szemantikánál) kezdtük, mert itt akadt meg a munka a LiLe-projekt esetében, ezért először arra kell választ találnunk, hogy lehetséges-e SQL-adatbázisban a szintaxist kezelni (totálisan lexikalista alapokon). Ha az alkalmazás sikeresnek bizonyul, a rendszerbe (kisebb változtatásokkal) beépítjük a LiLe-projekt által készített morfológiai táblastruktúrát, és áttöltjük a táblák tartalmát is (néhány táblánál ezt már meg is tettük, amelyek a szintaxis kezeléséhez is szükségesek, mint például szófajok).

6.3 Az adatbázis

A lexikont (akárcsak a LiLe-projekt esetében) egy relációs adatbázis képviseli. A számtalan adatbázis-kezelő rendszer és SQL-megvalósítás közül választásunk az Microsoft SQL Server 2005-re esett, mert ezáltal egy komplett relációsadatbázis-kezelő keret rendelkezésünkre áll.

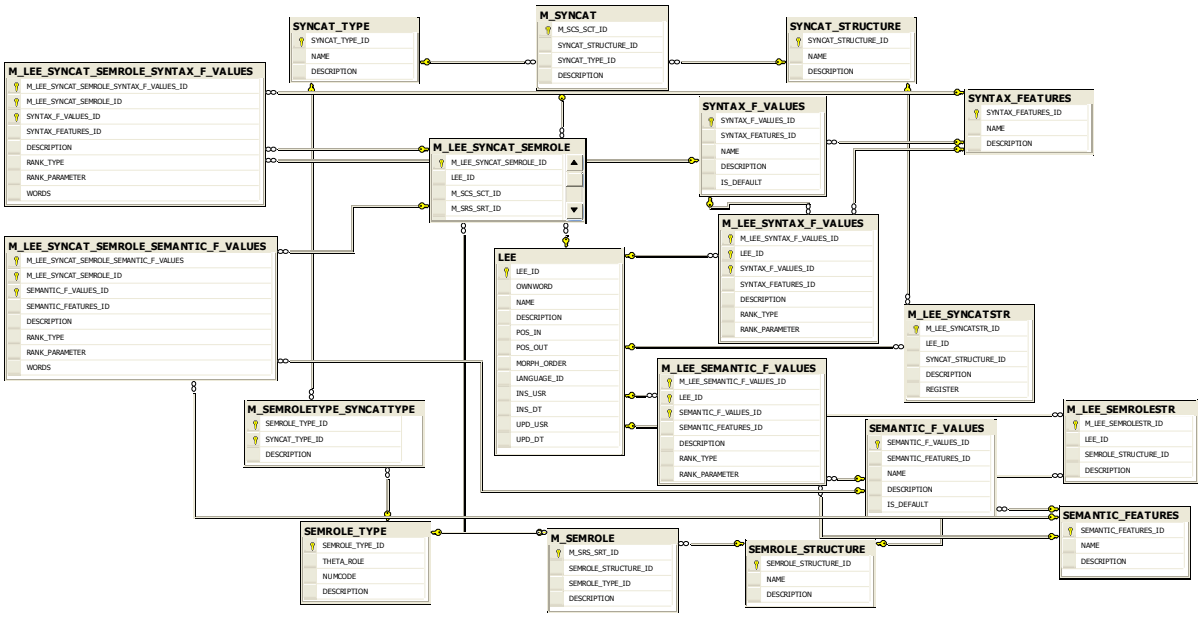
---

[48] Létezik a DRT-nek olyan kiterjesztése is, amely kimondottan a retorikai viszonyokra épít, azokat állítja a középpontba (SDRT, Segmented DRT, Lascarides–Asher 1993).

Továbbá a rendszer számos olyan szolgáltatással is rendelkezik, amelyet céljaink eléréséhez ki tudunk használni: elsődlegesen a kliens-szerver architektúrát, amelynek köszönhetően az adatbázis egyidejűleg több felhasználó és programmodul által is elérhető, vagyis a felhasználók a rá épített webes felületen keresztül, ingyenesen használhatják a rendszert.

Az adatbevitel egyelőre kézzel történik egy feltöltőprogramon keresztül, amelyet asp programnyelven készítettünk. Úgy tervezzük, hogy a zárt szóosztálybeli elemeket (és a nyílt szóosztálybeli elemek egy részét) kézzel rögzítjük[49], majd áttérünk részben automatikus feltöltésre. A kézzel történő adatbevitel azért nem szerencsés hosszú távon, mert nem hatékony, nagyon sok munkaórába kerülne olyan mennyiségű adatot rögzíteni, amellyel már tetszőleges szöveg elemzése lehetséges lenne. Azonban a tisztán automatikus módszer sem működne olyan magas nyelvi szintek esetében, mint a szintaxis és a szemantika (pl. Kálmán et al. 2003), ezért valamiféle hibrid módszer lenne a megfelelő megoldás.

A munka jelenleg ott tart, hogy létrehoztuk az új adatbázis szintaxissal és (részben) szemantikával kapcsolatos tábláit (17. ábra), illetve a LiLe adatbázisából átvettük a morfológiai táblákat és az azokban lévő adatok nagy részét. A szintaktikai (és vele párhuzamosan a szemantikai) rész feltöltését is megkezdtük: rögzítettünk számos vonzatkeretet, argumentumstruktúrát és egyéb (szintaktikai és szemantikai) tulajdonságot, illetve pár tucat lexikai egységet (töveket és toldalékokat), amelyekhez a különféle jegyeket és elvárásokat visszük fel jelenleg. Első lépésként nem tervezünk sokkal több lexikai egységet rögzíteni, előbb az elemzés pontos algoritmusát szeretnénk kidolgozni, majd megírni magát a programot, hogy tesztelhessük, megfelelő-e az adatbázisunk. Akkor sikerült megvalósítani a kitűzött célt (a totálisan lexikalista nyelvtan implementációját), ha a program alig tartalmaz majd többet, mint a szegmentálás menetét és az unifikáció algoritmusát, illetve ha semmilyen (vagy nagyon minimális) nyelvspecifikus információt hordoz, tehát ténylegesen a lexikonban tárolunk minden (a mondat összeépüléséhez szükséges) jegyet.



17. ábra: A táblastruktúra (szintaxis és szemantika)

[49] A kézi rögzítésbe – akárcsak a LiLe-projekt esetében – magyar szakos hallgatókat is bevonunk majd, ami mindkét fél számára hasznos: a projekt számára azért, mert így egy félév alatt jelentősen lehet növelni az adatbázis méretét; a hallgatók számára pedig azért, mert így egyrészt alaposan megismerkedhetnek az adott szóosztályra vonatkozó szintaktikai és szemantikai tulajdonságokkal, másrészt pedig betekintést nyerhetnek egy számítógépes nyelvészeti kutatásba.

A lexikai egységeket továbbra is a morfémákat tartalmazó LEE táblában tároljuk, és ezzel van kapcsolatban a többi (szintaktikai és szemantikai tulajdonságokat tartalmazó) tábla.[50] A „szemantikai oldalon" rögzítjük, hogy milyen argumentumkeretek (szemantikai szerep struktúrák, SEMROLE_STRUCTURE) lehetségesek univerzálisan, és azokban milyen szemantikai szerepek (SEMROLE_TYPE) találhatók (M_SEMROLE)[51]. Szemantikai szerepnek valami olyasmit tekintünk, mint a thematikus szerepek (ágens, páciens stb.), de körük kicsit más (vettünk fel újabbakat is), és nem ragaszkodunk valamiféle „címkéhez": a szerepeknek ugyan adunk nevet, hogy könnyebb legyen azonosítani őket, de az elemzés a hozzájuk rendelt szám alapján történik, amely valamiféle hierarchiába rendezi őket (Alberti 2006b). Az M_LEE_SEMROLESTR tábla pedig azt mondja meg, hogy az egyes lexikai egységeknek milyen argumentumstruktúráik vannak. Több argumentum-keretnek tekintjük például azt a névszókra jellemző jelenséget, hogy lehetnek ők maguk vonzatok, ez esetben predikátumot fognak keresni (*A kutya ugat*); illetve lehetnek névszói állítmányként predikátumok is, amikor alanyt keresnek (*Bodri egy kutya*). Ezek mellé fel kell még vennünk a rövid birtokos formát is, amely egy birtokos személyjelet keres (*a kutya csontja*).

A „szintaktikai oldalon" (a szemantikaival teljesen párhuzamosan) azt rögzítjük, hogy milyen vonzatkeretek (SYNCAT_STRUCTURE) lehetségesek, azokban milyen szintaktikai viszonyok (SYNCAT_TYPE) találhatók (M_SYNCAT), és hogy az egyes lexikai egységeknek milyen vonzatkereteik vannak (M_LEE_SYNCATSTR). Szintaktikai viszonyon is kissé bővebb halmazt értünk, mint a hagyományos vonzatkategóriák (NOM, ACC stb.): a készlet nemcsak esetragokat tartalmaz, hanem az olyan alakzatokat is, mint a névutók vagy az infinitívusz, illetve olyan állandósult kifejezésekként használt szerkezeteket is, amelyekre egy-egy esetragos alak cserélhető (például: *Péter elárult pár dolgot Mariról / Marival kapcsolatban*). Egy lexikai egység egy argumentumkeretéhez több vonzatkeret is tartozhat, például a *vár* igéhez a <NOM, ACC> (*Péter várja Julit*) és a <NOM, SUBLAT> (*Péter vár Julira*) rekordokat is hozzárendeljük.

A szemantikai szerepeket és a szintaktikai kategóriákat közvetlenül is összekötöttük (M_SEMROLETYPE_SYNCATTYPE), amire azért van szükség, mert bizonyos lexikai egységek nem írják elő valamelyik argumentumuk pontos szintaktikai formáját. Például a *lakik* predikátum második argumentuma egy HELY szemantikai szerepű elem, amelynek az esetét az ige nem specifikálja (de azt előírja, hogy megjelenjen, hiánya agrammatikus mondatot eredményez). Ezért a lexikai egységhez a <NOM, _> vonzatkeret kerül rögzítésre, ahol a '_' egy bármire illeszthető típust jelent. Ez természetesen nem azt eredményezi, hogy bármilyen esetű vonzattal grammatikus lesz a mondat, hanem csak olyannal, ami a HELY típust kifejezheti, mint például a -bAn *(egy szép házban),* az -On *(Pécsen)* vagy a mellett *(az iskola mellett)*. Ezt az információt rögzíti a szemantikai szerepeket és a szintaktikai kategóriákat összekötő tábla.

Az elemzéshez szükség van arra is, hogy hivatkozni tudjunk egy adott lexikai egység egy adott argumentumkeretében és egy adott vonzatkeretében lévő valamely elemre, ezt a célt az M_LEE_SYNCAT_SEMROLE tábla szolgálja. Ebben egy rekord például a *vár* lexikai egység <AG, PAT> argumentumstruktúrájának, illetve <NOM, ACC> vonzat-keretének páciensi szerepű, illetve tárgyesetű argumentuma (akit vagy amit vár valaki vagy valami). Ehhez rendelünk egy azonosítót, így később ezzel a számmal tudjuk hivatkozni ezt az adott argumentumot.

---

[50] A morfofonológiai komponens esetében az ábrán nem szereplő ALLOMORPH tábla elemeihez rögzítjük a különféle tulajdonságokat.

[51] Az 'M' betű mutatja, hogy kapcsolótáblaról (matching table) van szó.

A lexikai egységek szintaktikai tulajdonságait az M_LEE_SYNTAX_F_VALUES tábla tartalmazza, amely egy kapcsolótábla a lexikai egységek (LEE) és a szintaktikai tulajdonságok értékei (SYNTAX_F_VALUES) között, ahol az utóbbi tartalmazza az összes releváns szintaktikai jegy (SYNTAX_FEATURES, például szám, személy, határozottság, eset) összes lehetséges értékét. Ugyanígy létezik három tábla a szemantikai tulajdonságokra is: SEMANTIC_FEATURES a jelentéstani jegyeknek (például élő, ember, absztrakt), SEMANTIC_F_VALUES a lehetséges értékeiknek, és M_LEE_SEMANTIC_F_VALUES a lexikai egységekkel való kapcsolat megteremtésére. Mindkét kapcsolótáblának van két rangokkal kapcsolatos mezője (rank_type és rank_parameter), ahol a korábban már felvázolt (optimalitáselmélethez hasonló) módon az adott lexikai egység adott tulajdonságáról megmondjuk, hogy milyen erősségű, vagyis más elem által felülírható-e.

A szintaktikai és a szemantikai jegyek értékeit nem csak a LEE-vel kötjük össze, hanem az M_LEE_SYNCAT_SEMROLE táblával is (amely a lexikai egység egy adott argumentumát „ragadja meg”): itt mondjuk meg, hogy az elemnek milyen elvárásai vannak pontosan, vagyis hogy az egyes argumentumaiktól milyen szintaktikai és szemantikai jegyeket követel meg (M_LEE_SYNCAT_SEMROLE_SYNTAX_F_VALUES és M_LEE_SYNCAT_SEMROLE_SEMANTIC_F_VALUES). Ezekhez az elvárásokhoz is rögzítünk rangparamétereket, hiszen a legtöbb elvárás ugyanúgy felülírható, mint ahogy egy tulajdonság. A *vár* ige tárgyához például rögzítjük, hogy (gyenge ranggal) határozatlan legyen, amit felül tud írni például az *-om* személyrag; illetve hogy (szintén gyenge ranggal) az ige a tárgyát maga mögött keresi, amit felül tud írni például a fókusz. Egy argumentumot a régens több lexikai egységben találhat meg: az esetét a rag mutatja meg, a határozottságát vagy a névelő vagy (tulajdonnevek esetében) a névszói tő, a különféle szemantikai jegyeket pedig a főnév hordozza. Hogy a keresést megkönnyítsük, felvettünk egy mezőt (words), amelyben azt rögzítjük, hogy a régens mely jegyeket keressen ugyanabban a szóban, és melyeket keressen egy másikban; ami nyelvenként (és tulajdonságonként) különbözhet.

A szintaxisnak természetesen a szabad határozókról is számot kell adnia. Ebben a modellben szabad határozónak azt tekintjük, ahol az esetrag kompozicionálisan szerepel (akárcsak Gábor–Héja (2006b)-nél), ám mégis vonzatnak tekintjük az olyan (esetenként kompozicionális) elemeket, amelyeknek a jelenlétét valamelyik másik elem előírja. Miután minden régens vonzatkövetelménye kielégült, a „maradék” elemekről el kell dönteni, hogy lehetnek-e szabad határozók, vagyis hogy tartalmaznak-e olyan morfémát, amelyet kompozicionálisan értelmezve (megfelelő szemantikai jegyekkel bíró) idő-, hely-, mód- vagy egyéb határozót kapunk.

6.4 A fókusz kezelése

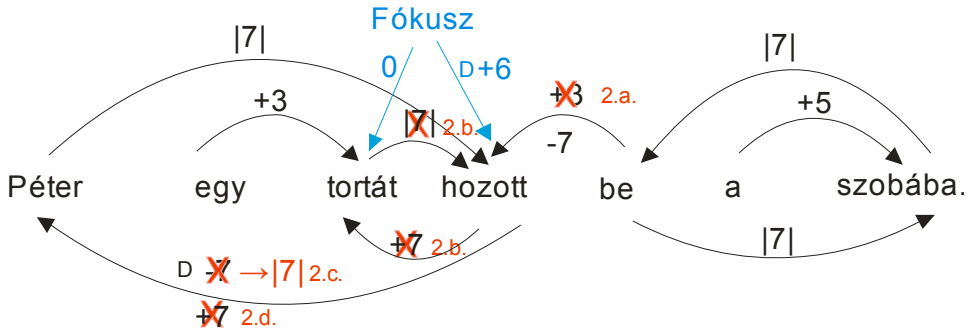
Azért döntöttünk elsőként a fókusz kidolgozása mellett, mert a legtöbb számítógépes nyelvészeti alkalmazás nem tűzi ki célul a fókusz felismerését, így a rendszerünk már kis adatbázison is tudna egy olyan gyakorlati célt szolgálni, amelyet más alkalmazások nem valósítanak meg. Fókusz alatt azonosító fókuszt és nem információs fókuszt értek, vagyis azt a jelenséget, amely a fókuszált elemet azonosító/kirekesztő értelmezéssel látja el, és (a magyarban) az ige előtti pozícióba helyezi.

A modell kétféle rangparaméterrel dolgozik. Korábban szó volt már a szórendért felelős (szomszédossági viszonyokat megadó) rangparaméterekről, ahol a követelményt mindenképpen ki kell elégíteni, akár részlegesen is (vagyis ha például a névelő a főnév előtt akar lenni 5-ös rangban, akkor mindenképpen előtte kell lennie, legfeljebb beékelődhetnek közéjük más, erősebb rangú elemek); ez az ún. *recesszív* rangparaméter. A másik típust a *domináns* rangparaméterek képviselik, amelyek esetében elegendő, ha egy ellentmondó, erősebb követelmény ki van elégítve; ilyen a tulajdonságokhoz és elvárásokhoz rendelt

rangparaméter (ha valami felülírja azt a jegyet, hogy az ige egyes számú alanyt vár, akkor annak már semmilyen formában nem kell teljesülnie), vagy ilyen a fókusz kezelésénél használt rangparaméter is.

Az igekötők esetében[52] például kétféle recesszív rangparaméter is él (Szilágyi et al. 2007). Az egyik esetben az igekötő az ige elé kell, hogy kerüljön, ezt egy erős (+3) rangparaméter mondja meg (a plusz jelenti, hogy az igekötő az igét maga mögött keresi, a szám pedig az erősséget: minél kisebb, annál erősebb); vagyis ha az igekötő az ige előtt van, nem sok elem ékelődhet be közéjük (egy ilyen például az *is*). Ekkor a hangsúly az igekötőre esik (és az igéről lekerül), így egy fonológiai szót alkotnak. A második esetben az igekötő kerülhet az ige mögé is egy gyengébb (-7) rangparaméter szerint, amely rangparaméter megegyezik a vonzatok rangjával, vagyis az ige és az igekötő közé ebben az esetben akár az ige egy vonzata is beékelődhet (például *Marinak adta Juli oda a kulcsot*). Ekkor mindkét elem külön fonológiai szóként realizálódik. Ha nincs domináns rangparaméterű elem a mondatban, akkor az erősebb (+3) rang elégül ki, vagyis az igekötő az ige elé kerül.

A fókusz egy olyan lexikai egység a modellben, amelyhez (a magyarban) nem tartozik sajátszó, a hangalakja csupán egy hansúlyjel; de vannak olyan nyelvek is (például eszkimó, kecsua, tamil), ahol önálló morféma. Két elemet keres a mondatban: a fókuszált elemet (0 ranggal) és az igét. Az igétől +6 domináns rangban várja el, hogy közvetlenül mögötte legyen, vagyis az igekötő +3-as recesszív rangú követelménye (hogy ő legyen az ige előtt) nem fog tudni teljesülni, ezért a másik rangparaméter fog kielégítődni, vagyis az igekötő az ige mögé kerül, és a fókusz lesz közvetlenül az ige előtt.[53] Megfigyelhetünk olyan elemeket is, amelyek jelenlétében mindig található fókusz is, ilyen például a *csak,* illetve a mondatrésztagadást jelző *nem.* Ezek az elemek erős domináns ranggal vonzzák a fókuszt maguk mögé, ami a szórendi változásról gondoskodik. Egy példa a 18. ábrán látható, a számok magyarázata (az elemzés algoritmusa) pedig alatta olvasható.



18. ábra: Egy fókuszos mondat lexikai egységeinek sorrendre vonatkozó követelményrendszere.

A régensek a vonzataikat szintaktikailag maguk mögött várják (+7 rang), szemantikailag az alanyukat maguk előtt, ami egy domináns követelmény (D -7); a vonzatok nem szabják meg, hogy a régensüket maguk előtt vagy maguk mögött várják-e (|7| rang). A keresést az időjeles igétől görgetjük: mindegyik elemnek megkeressük a követelményeit, majd a megmaradt elemek (szabad bővítmények) legitimálódnak (ha találnak olyan elemet, amelyhez kapcsolódhatnak). Az elemzés menete:

[52] Illetve az ige aspektualizáló argumentumának esetében, amely rendszerint az igekötő, de lehet más elem is (a *lakik* ige esetében a hely: *András Szegeden lakik*), illetve alkalmanként lehet maga az ige (például *Péter csalódik Mariban*).

[53] Hasonlóan kezelhető a telikus szituációkra vonatkozó progresszív alak működése is.

1. A domináns (D) rangparaméterek ellenőrzése
   itt: *tortát* (fókuszált elem) közvetlenül megelőzi az igét (D+6 rang)
2. A dominánssal ellentétes követelmények törlése:
   a. ugyanarra vonatkozó ugyanolyan előjelű rangok
      itt: igére vonatkozó igekötőtől jövő +3 irtódik (marad a -7)
   b. a két elem közti egyéb rangok
      itt: *tortát* (fókuszált elem) és *hozott* (ige) közti +7, illetve |7| rang
   c. másra vonatkozó ellentétes előjelű iránya
      itt: az ige (*hozott*) alanyra (*Péter*) vonatkozó -7 rangja |7|-re változik
      (fókusznál az alany – természetes szubjektum – az ige mögött is lehet)
   d. ugyanannak az elemnek két ellentétes követelménye közül a domináns győz
      itt: az ige (*hozott*) alanyra (*Péter*) vonatkozó |7| és +7 követelményei közül az |7| marad – ami a mondat egyéb variánsainak is megfelel
      (ha nincs fókusz, az alany lesz elöl: -7 domináns vs. +7 recesszív rang)
3. Recesszív rangparaméterek ellenőrzése: két elem vagy közvetlenül szomszédos, vagy van köztük olyan elem, amelyik valamelyikhez ugyanolyan vagy erősebb rangban kötődik (ez behozhat hozzá kapcsolódó elemeket).
   itt: közvetlenül szomszédosak OK
   igekötő -7 rangja OK (ige előtte, köztük nincs elem, de lehetne másik 7 rangú)
   szobába és *be* közti |7| rang OK (köztük csak az erősebb rangú névelő)
   Péter és *hozott* közti |7| rang OK (köztük a 6 rangú *tortát* és a hozzá kapcsolódó *egy*)

A megközelítés egyik előnye, hogy bármely elem kaphat fókuszértelmezést, akár egy melléknév vagy egy számnév is (93), szemben a frázisstruktúra nyelvtanokkal, ahol mindig az egész főnévi csoport kerül fókuszba. Ekkor nem az ige következik közvetlenül a fókuszált elem után, hanem más hozzá kapcsolódó szó, amely az erősebb rang miatt beékelődhet (a fókusz +6 rangja gyengébb, mint a melléknév +2, vagy a számnév +3 rangja).

(93) a. Péter egy LÁNNYAL találkozott.
b. Péter egy OKOS lánnyal találkozott.
c. Péter KÉT okos lánnyal találkozott.

Ha gyengébb rangú szó akar a fókuszált elem és az ige közé beékelődni, a mondat agrammatikus lesz: például ha egy olyan főnév kap fókuszértelmezést, amelynek van vonzata (amelyet +7 rangban vár maga mögé), akkor a vonzat nem tud beékelődni (94), így az ige mögé kerül (94).

(94) a. *Péter egy VERSET Adytól olvasott fel.
b. Péter egy VERSET olvasott fel Adytól.

Ebben a pontban felvázoltam a fókusz egy lehetséges – totálisan lexikalista – kezelési módját. A megoldás elméletben megfelelőnek tűnik, az elemzett mondatokon a várt eredményt adja. Hogy mennyire működik a gyakorlatban, az akkor derül ki, amikor implementáljuk: felvisszük a releváns lexikai egységeket és megírjuk az elemzőprogramot.

6.5 Összegzés – ReALIS-projekt

A totális lexikalizmus legitimálásának harmadik lépése a ReALIS-projekt, amelynek gyakorlati célja a korábbi eredmények felhasználásával egy nagyobb adatbázison működő hatékony mondat-, illetve szövegelemző (majd gépi fordító) rendszer létrehozása. Jelenleg a

szintaktikai és részben a szemantikai komponens megvalósításán dolgozunk, a morfofonológiát alapvetően a LiLe-projektből vesszük át. A következő lépés a szemantikai reprezentáció hozzárendelésének az implementálása lesz, amely a rendszer központi részét képezi; mert a különféle számítógépes nyelvészeti alkalmazások példája azt mutatja, hogy intelligens alkalmazások csak „igazi" nyelvészeti alapokon készíthetők, ahhoz pedig alaposan kidolgozott szemantikai rendszerre van szükség.

A munka első részfeladatát már teljesítettük: létrehoztuk az adatbázist, amely (reményeink szerint) kellően univerzális ahhoz, hogy bármely nyelv lexikai egységeit (és azok minden tulajdonságát) tartalmazni tudja. Jelenleg az elemző algoritmuson dolgozunk, amelyet implementálva tesztelhető lesz az adatbázis szerkezete, hogy ténylegesen minden tulajdonság tárolható-e benne. Ezzel párhuzamosan zajlik a különféle nyelvi jelenségek kidolgozása, ezek közül egynek – a fókusznak – egy lehetséges kezelését mutattam be részletesebben, mert először ezt tervezzük olyan mértékben kidolgozni, hogy gyakorlati alkalmazást lehessen rá építeni (mert ez egy olyan terület, amelyet jellemzően nem céloznak meg más projektek). Az implementáció során a részletek még változhatnak; ahogy a korábbi implementációk is visszahatottak magára az elméletre, úgy erről a megoldásról is kiderülhet, hogy nem teljesen írja le a jelenséget (alul és/vagy túlgenerál); ebben az esetben tovább finomítjuk majd az elemzést, kissé máshogy paraméterezzük, egészen addig, amíg úgy nem tűnik, hogy tökéletesen leírja a fókusz működését.

Rendszerünk előnye és újszerűsége abban rejlik, hogy a lexikonban ugyanazon keretek között kezelhető a szintaxis körébe (is) tartozó számos tényező (predikátum-argumentum, illetve régens-vonzat viszonyok, szabad határozók, szórend). A rangparaméterek elegánsan számot adnak az egy nyelven belüli szórendi variációkról (a szón belül a morfémák sorrendjéről), valamint a nyelvek közötti különbségekről is. A domináns rangparaméterek használatával pedig a szórendet megvariáló, sokszor láthatatlan elemek (fókusz, progresszív) is kezelhetőek.

7 Összefoglalás

7.1 Elméleti háttér

A generatív nyelvészet kezdeti szakaszában a mondatok grammatikalitásának vizsgálata során a szintaktikai szerkezetnek jutott a vezető szerep, míg a szótári komponens feladata pusztán a szavak felsorolásából állt. Ez a megközelítés többé-kevésbé sikeresnek bizonyult az angolhoz hasonló konfigurációs nyelvek esetében (bár a transzformációk létjogosultságát nem sikerült igazolni), a szórend helyett inkább morfológiai eszközöket használó nyelvek esetében azonban nehézkesnek bizonyult.

A hetvenes évektől kezdődően egyre több olyan elmélet született, amely a lexikont helyezi előtérbe: a nyelvi jellegzetességeket inkább a szótárban tárolja, míg a szintaktikai komponens csupán néhány nagyon általános frázisstruktúra-építő szabályt tartalmaz. Ezek az ún. *lexikalista nyelvtanok* (pl. LFG, HPSG) eredményesebben kezelik a magyarhoz hasonló szabadabb szórendű nyelveket, miközben a konfigurációs nyelvek leírására ugyanúgy alkalmasak, mint a transzformációs elméletek.

Ezt a szemléletet követik a kategoriális nyelvtanok is, amelyeknek egy változata – az unifikációs kategoriális nyelvtan – megvalósítja a *radikális lexikalizmust* (Karttunen 1986), ahol annyira gazdag a lexikai egységek szótári leírása, hogy a frázisstruktúra gyakorlatilag feleslegessé válik. A mondatok összeépítéséhez két művelet elegendő: a szomszédos egységek kategóriái között működő *függvényalkalmazás* és az egymással relációba kerülő elemek szótári jegyein működő *unifikáció*.

A szintaktikai komponens (frázisstruktúra) redukálásának ötletét a végsőkig viszi Alberti (1999) azáltal, hogy megalkot egy „totálisan” lexikalista nyelvtant, a GASG-t. Az elmélet egy olyan módosított unifikációs kategoriális nyelvtannak tekinthető, amely már a függvényalkalmazás műveletét sem tartalmazza, így tehát minden információt a lexikonban tárol. Egy nagyon erős univerzális grammatikai megszorítást tesztel: azt vizsgálja, hogy lehetséges-e úgy leírni a nyelvet, hogy nem támaszkodunk semmiféle szintaktikai szabályrendszerre. Tehát nem csupán a transzformációk létét tagadja, hanem frázisstruktúrát sem épít; a grammatikalitásról és a mondatok jelentéséről pusztán a gazdagon strukturált lexikon és az unifikáció művelete gondoskodik.

A GASG létjogosultsága melletti egyik fő elméleti érv, hogy *kompozicionális* szemantikai partnereként szolgálhat a DRT-szerű diskurzus-szemantikai elméleteknek. A Frege-féle kompozicionalitási elv értelmében a kutatók régóta keresnek egy izomorf szintaxis–szemantika párt. Montague szolgáltatott először (a ’70-es években) egy olyan szemantikaelméletet, amely a chomskyánus szintaxissal kompozicionális tudott lenni, amihez azonban (a két rendszer közötti nagymértékű különbözőség miatt) egy nagyon erős eszköz – a λ–absztrakció – alkalmazására volt szükség. Egy másik lehetséges megoldása a problémának, ha egy jól működő szemantikaelméletet tekintünk kiindulásnak, és ahhoz keresünk egy vele kompozicionális szintaxist; ezt valósítja meg a GASG, amelyre ugyanaz a hierarchiamentesség jellemző, mint a DRT-re, így a két rendszer között egy szigorúbb értelemben vett kompozicionalitás tud megvalósulni.

A nyelvtan tehát nem más, mint egy óriási lexikon, ahol a lexikai egységek jellemzése tartalmazza az elem saját tulajdonságait (fonológiai forma, szófaj, vonzatkeret, az egységhez tartozó proto-DRS stb.), valamint a „környezeti követelményeit”, vagyis hogy hány és milyen tulajdonságú elemet keres a mondatban. A szórendre vonatkozó megszorítások ugyanúgy egy lexikai tulajdonságként vannak tárolva, mint például az egyeztetéssel kapcsolatos elvárások, ezért nincs szükség frázisstruktúra építésére. A GASG tehát egy

homogén nyelvtan, amely egységesen kezeli a különféle nyelvi jelenségeket, mint például a szavak jólformáltsága, morfémasorrend, a vonzatok megléte, eset, egyeztetés, szórend, szemantikai jegyek, vagy akár szemantikai reprezentáció hozzárendelése: csak tulajdonságok és követelmények vannak, amelyek ha tudnak unifikálódni, grammatikus lesz a mondat, és előállnak a különféle reprezentációk. Ezáltal nagyfokú univerzalitás érhető el: bármely nyelv leírását ugyanabban a keretben lehet megtenni, attól függetlenül, hogy az adott nyelv milyen eszközökkel (szórend vagy morfológia) fejezi inkább ki a grammatikai viszonyokat; az elmélet nem tulajdonít nagyobb jelentőséget egyik módszernek sem a másikhoz képest.

A szintaktikai elemzés eredménye – mivel frázisstruktúra nem épül – gyakorlatilag egy függőségi viszonyrendszer; viszont az elmélet nem csupán egy függőségi nyelvtan, mert számot ad a szórendről is: egyrészt a különböző szórendi variánsok grammatikalitásáról, másrészt az eltérő szórenddel járó szemantikai változásokról (például fókusz). Mindezt az ún. *rangparaméterek* segítségével éri el, amelyek azt rögzítik, hogy egy adott követelmény milyen erősségű. Például a szórend esetében a kiindulás az, hogy minden elem szomszédos kíván lenni minden olyan elemmel, amellyel szintaktikai viszonyt létesít, ami gyakran nem tud teljesülni, hiszen egy elemmel több másik is kapcsolatot létesíthet, de legfeljebb kettő tud vele szomszédos lenni. Ezért van szükség arra, hogy ez a közvetlen megelőzési igény rangparaméterekkel legyen ellátva, vagyis rögzítve legyen, melyik igény milyen erősségű, és ha ellentmondó követelmények lépnek fel, akkor az erősebb „győz”. A rangparaméterek alkalmazása kiterjeszthető más jelenségekre is, például a fókusz vagy a progresszivitás kezelésére; illetve segítségével még nagyobb fokú univerzalitás érhető el, hiszen elvileg bármilyen tulajdonság vagy elvárás mellé rögzíthető rangparaméter, amelyek egymással „harcolva” végül kiadják az adott nyelv adott jelenségének felszíni megvalósulását az adott mondatban (az optimalitáselmélet filozófiájához hasonlóan).

7.2 Nyelvtechnológia

A nyelvelemző rendszerek alapvetően két típusba sorolhatók: lehetnek szabály alapúak, illetve korpusz alapúak, amelyek közül a számítógépes nyelvészet korábbi szakaszára az előbbi, míg az elmúlt néhány évre inkább az utóbbi megközelítés volt jellemző. Mindkettőnek vannak hiányosságai, emiatt napjainkban az igazán hatékony rendszerek mindkettőt alkalmazzák. A szabály alapú rendszerek esetében az adatrögzítés lassú, rengeteg munkaórát igényel; illetve bizonyos jelenségek (például a többértelműség) kezelését nem is igazán lehet szabállyal megragadni, hatékonyabb példákat sorakoztatni, vagy a kontextusra támaszkodni. A korpusz alapú rendszerek esetében lényegesen rövidebb idő alatt lehet nagy mennyiségű adathoz jutni, azonban a megközelítés akkor a legeredményesebb, ha a felhasznált korpuszok alaposan annotáltak (mint például a treebankok, amelyek nem csupán morfológiai, hanem szintaktikai vagy akár szemantikai információt is tartalmaznak), ami viszont szintén sok befektetett munkát igényel, ezért számos nyelvre nem is léteznek ilyen adatbázisok. További hátránya a korpusz alapú megközelítésnek, hogy (akármekkora is a méretük) nem minden lehetséges nyelvi forma található meg bennük; illetve hogy intelligensebb célokra nem igazán alkalmasak. A két eszközt kombináló hibrid rendszerek igyekeznek mindkét megközelítés előnyeit kihasználni, így a korábbiaknál pontosabb alkalmazásokat tudnak viszonylag rövid idő alatt fejleszteni.

Különbség van a szabály alapú rendszerek között abban, hogy mennyire alaposan elemzik a mondatokat (shallow vs. deep parsing). A sekélyelemzés előnye, hogy gyors és robusztus, hátránya viszont, hogy mivel csupán részleges elemzést végez, nem annyira precíz (Frank 2003), így komplexebb célokra (mint például az igazán jó minőségű gépi fordítás) nem alkalmas. A mélyelemzést végző rendszerek sokkal alaposabbak és

pontosabbak, és az utóbbi időben lefedettségben is felveszik a versenyt a sekélyelemző rendszerekkel.

Egyetértés mutatkozik abban, hogy ha intelligens alkalmazások megalkotása a cél (például szövegek lényegének összefoglalása, kérdésmegválaszolás, jó minőségű gépi fordítás), akkor arra a szabály alapú, mélyelemzést végző rendszerek a legalkalmasabbak, mert pontosabbak, és mert képesek szemantikai reprezentációt társítani a mondatokhoz. A kezdeti nagyobb energiabefektetés (nyelvészeti alapok, kézi adatrögzítés) pedig megtérül később, például amikor új nyelvekre dolgozzák ki a rendszert (Forst et al. 2005). További előnyük az újrahasznosíthatóság, vagyis hogy az ilyen módszerrel készült rendszerek más alkalmazásokban is használhatók (például egy elemző gépi fordításra). Az igazán jól működő és hatékony mélyelemző rendszerek unifikációs mechanizmusokat használnak, ami azért hasznos, mert az ilyen nyelvtan egyszerűbb szabályokat alkalmaz, gyorsabb, egyszintű, lexikalista és megfordítható (Mitkov 2003), továbbá nem csak a konfigurációs nyelveket (mint az angol) kezeli hatékonyan, hanem a magyarhoz hasonló szabadabb szórendű nyelveket is.

A lexikalista megközelítés tehát nem csupán elméletben, hanem a nyelvtechnológiában is sikeresnek bizonyult. Közülük a legeredményesebbek LFG vagy HPSG formalizmust használnak. Számos nyelvre léteznek már nagy lefedettségű elemzőik, amelyek nagyon jó minőségű és alapos elemzést adnak, továbbá szemantikai reprezentációt is társítanak a mondatokhoz. Egyik legnagyobb előnyük, hogy különböző nyelvekre alkalmazzák ugyanazt az elméleti keretet, így el tudják érni, hogy a különböző nyelvű elemzések szinte teljesen párhuzamosak legyenek, ami jelentősen megkönnyíti az elemzőkre épülő gépi fordító rendszerek fejlesztését. Az LFG-t mint elméleti keretet használó elemzők közül a legígéretesebbek azok, amelyeket a Parallel Grammar projekt (ParGram, Butt et al. 2002) keretében fejlesztenek; a legkomplexebb HPSG-formalizmust használó alkalmazás pedig a DELPH-IN (Deep Linguistic Processing with HPSG Initiative, pl. Bond et al. 2005), amely nyelvészeti és statisztikai módszerek kombinációját alkalmazza.

Nem csupán gyakorlati, hanem elméleti haszna is van, ha egy nyelvelmélethez implementáció készíthető, hiszen a működő algoritmusok teszik kétségtelenné, hogy jól formalizált, egzakt rendszer áll rendelkezésünkre. Alapvetően emiatt döntöttünk úgy, hogy elkészítjük a totálisan lexikalista GASG számítógépes implementációját. További célunk megvizsgálni a totális lexikalizmus használhatóságát a nyelvtechnológiában, hogy alapjain lehetséges-e jól működő nyelvelemzőt készíteni, és ha igen, az kellően hatékony tud-e lenni. Az implementáló programunk tevékenysége nem más, mint a „generatív alapfeladat” végrehajtása: egyértelműen el kell döntenie egy beírt szósorról, hogy az grammatikus-e, és amennyiben az, morfoszintaktikai és szemantikai reprezentációt kell hozzá rendelnie. Ezek alapján a gyakorlati célunk egy mondatelemző létrehozása, amely kezdetben magyar, majd angol, illetve egyéb nyelvű szövegeket tud elemezni, hozzájuk reprezentációkat társítani, és köztük a gépi fordítást megvalósítani. A kiinduló (elméleti) cél megvalósítására alakult a GeLexi-projekt, a gyakorlatibb (hatékonyságot is számításba vevő) célra pedig a LiLe-projekt, majd a még alaposabb szemantikai reprezentációt célzó ReALIS-projekt.

Az implementáció készítése visszahathat magára az elméletre is, hiszen felszínre kerülhetnek olyan problémák, amelyeket az elmélet eredeti formájában nem tudna kezelni, illetve születhetnek olyan megoldások, amelyek hatékonyabban kezelik a nyelvet, mint az eredeti elképzelés, de az elmélet szellemiségével nincsenek ellentmondásban, vagy akár még jobban megvalósítják azt. Ez utóbbi történt a GASG implementálása során is, amikor egyértelművé vált, hogy a nyelvtan kívánatos teljes homogenitását az biztosítja, ha a morfológiát is totálisan lexikalista módon közelítjük meg: nem a szavakhoz, hanem közvetlenül a morfémákhoz rendeljük a gazdagon strukturált, minden grammatikai szintről egyidejűleg információt hordozó lexikai egységeket, és a mondatelemzés során az is eldől,

hogy mely elemek épülnek össze szóvá, és melyek alkotnak (egymás közelében maradó) külön szavakat. A totálisan lexikalista morfológia számára tehát nem okoz gondot, hogy a különböző nyelvekben különböző helyen található a szószint, hiszen az is csak egy tulajdonság, hogy az adott lexikai egység (állítást hordozó morféma) külön szót alkot-e, vagy affixumként valósul meg.

7.3 Eredményeink

A GeLexi-projekt célja tehát annak igazolása volt, hogy a GASG egy jól formalizált, egzakt rendszer, illetve hogy a totálisan lexikalista megközelítés alkalmazható a számítógépes nyelvészetben, eldönthető egy mondatról annak grammatikus volta csupán a lexikonban tárolt jegyek unifikációja alapján. Az elmélet implementációját Prolog programnyelven készítettük, amelyet kimondottan arra a célra fejlesztettek, hogy unifikációval tudjon különféle problémákat megoldani; így tehát a GASG által alkalmazott egyetlen műveletet beépítve tartalmazza. Az adatbázisba csupán néhányszáz lexikai egységet rögzítettünk, hiszen a célunk (első lépésként) a mechanizmusok működőképességének vizsgálata volt, a hatékonyság kérdését egyelőre félretettük.

A program bemenete egy magyar vagy angol nyelvű szósor, amelyről az elemző eldönti, hogy grammatikus mondatot alkot-e, és amennyiben igen, négyféle reprezentációt társít hozzá. Az első kimenet a releváns lexikai egységek listáját tartalmazza, ezt követi a szintaktikai viszonyrendszer ábrázolása, hogy mely lexikai egységek között milyen grammatikai viszony létesül, majd a mondat ún. kopredikációs hálózatát olvashatjuk, vagyis hogy mely elemek tesznek állítást ugyanarról a referensről, végül pedig a szemantikai reprezentáció látható, egy (L)DRS. Az implementálás során többféle nyelvi jelenség kezelését kidolgoztuk: a program számot ad a morfofonológiai váltakozásokról, a zéró névmásokról, elemezni tud műveltetést, igekötőt vagy vonzatos melléknevet tartalmazó mondatot, illetve a mellérendelés kidolgozását is megkezdtük.

A szemantikai reprezentáció egy egyszerűsített angol nyelven tartalmazza a mondatban található összes információt, így rendszerünk a géppel támogatott fordítás területén is hasznosítható. Míg egy külföldinek évekbe telne megtanulnia magyarul, addig a program által előállított DRS-szerű szemantikai reprezentáció olvasását bármely angolul értő felhasználó alig egy óra alatt elsajátíthatja. A ragozó nyelvek esetében különösen hasznos lehet egy ilyen eszköz – szemben egy egyszerű szótárral –, hiszen például a *Kerestelek* mondat fordításakor nem található meg a szótárban az alany és a tárgy számára és személyére vonatkozó információ, ahogy az sem, hogy a cselekvés a múltban zajlott.

Angol nyelvű lexikai egységeket azért vettünk fel az adatbázisba, hogy megmutassuk a rendszer univerzalitását, vagyis hogy a struktúra más nyelvek lexikai egységeinek a tárolására is alkalmas, amelyek lexikonban tárolt jegyein ugyanúgy működik az unifikáció. A program tehát angol nyelvű mondatokat is képes elemezni, és ugyanolyan típusú reprezentációkat társít hozzájuk, mint a magyar nyelvűekhez. A nyelvek közötti különbségek jelentős része már a kopredikációs hálózat szintjén eltűnik, egy magyar nyelvű mondathoz és annak angol megfelelőjéhez rendelt reprezentáció csak az elemek számozásában tér el.

Kihasználva, hogy amit a Prolog elemezni tud, azt generálni is, a program kétirányú használatával a gépi fordítást is megvalósítottuk a két nyelv között (továbbra is totálisan lexikalista alapokon). A rendszer először elemzi a forrásnyelvi mondatot – közben előállítja a különféle reprezentációkat –, azután a kopredikációs hálózat alapján betölti a célnyelvi lexikai egységeket, majd (az egyeztetésért felelős morfémák helyére változókat feltételezve) generálja a lehetséges morfémasorokat, végül ezekre meghívja a célnyelvi elemzőt, így a fordítás kimenete csak grammatikus mondat lehet. A kidolgozott keret univerzális, így a program elvileg bármely két nyelv között meg tudná valósítani a fordítást, amelyek lexikai

egységeit tartalmazná. A rendszer számára a nagyon különböző szerkezetű nyelvek közötti fordítás sem nehezebb, mivel minden állítással bíró elem egy-egy lexikai egység (totálisan lexikalista morfológia), így nem okoz gondot a szószint esetleges különbözősége; illetve minden nyelvi tulajdonság, minden elvárás és minden „szabály" (még a sorrendre vonatkozó megszorítások is) az adott lexikai egység leírásában kerülnek tárolásra, vagyis minden információ a lexikonban található, így nincs szükség nyelvspecifikus szabályokra.

A GeLexi-projekt tehát egy működő implementációval igazolta, hogy a GASG egy jól formalizált, egzakt rendszer, és hogy a totális lexikalizmus alkalmazható a nyelvtechnológia területén. Következő lépésként a LiLe-projekt vállalkozott arra, hogy a rendszert egy modernebb fejlesztőkörnyezetbe helyezi, ahol már a hatékonyság is vizsgálható, és a felhasználóbarát felület is fontos szempont. A lexikon számára egy SQL-adatbázist készítettünk, amelynek a tartalma weben keresztül is megjeleníthető, és hozzá Delphi programnyelven feltöltő- és elemzőprogramot írtunk; így a modernebb környezetbe való helyezést és a felhasználóbarát felület létrehozását teljesítettük. A megvalósításban a morfológiai komponensig jutottunk, az azzal kapcsolatos táblákat töltöttük fel adatokkal, így a program is csupán morfológiai szempontból tudja elemezni a beírt szavakat. Az adatbázis méretét sem sikerült jelentősen megnövelni, így a hatékonyságot továbbra sem tudtuk vizsgálni. A LiLe-projekt jelentősége a totális lexikalizmus legitimálásában az, hogy kidolgozta a lexikai egységek SQL-adatbázisban való tárolásának módját, létrehozott egy olyan struktúrát, amely a későbbi kutatásokhoz alapul szolgálhat; illetve újabb (közbeeső) célokat fogalmazott meg, mint például az oktatás támogatása, vagy egy „dinamikus korpusz" létrehozása. Továbbá ebben a munkában jelent meg először a totális lexikalizmus eszméjébe maximálisan illeszkedő lehetőség, hogy bármely unifikálandó jegy (tulajdonság vagy elvárás) kikapcsolható, ami a rendszer robusztusságát tudja biztosítani.

Jelenleg a két projekt egyesítésével egy új adatbázison dolgozunk, amely központi komponense egy minden eddiginél részletesebb szemantikai reprezentáció lesz (ReALIS, Alberti 2005). A lexikont SQL-adatbázis formájában építjük, és célunk, hogy olyannyira minden tulajdonságot tartalmazzon a lexikai egységekről, hogy az adatbázishoz tartozó programnak az unifikáción kívül csupán néhány nagyon általános eljárást kelljen tartalmaznia. A megvalósítást ezúttal a szintaktikai komponensnél kezdtük, a morfológiával kapcsolatos táblákat (azok tartalmával együtt) a LiLe-projekt szolgáltatja. A ReALIS-projekt a korábbiakhoz képest számos újítást hordoz, amelyek közül a legfontosabb a korábban csak a szórend kialakításánál alkalmazott rangparaméterek kiterjesztése más nyelvi jelenségek kezelésére (mint például a fókusz).

Az elmúlt néhány évben érdekes kísérletet végeztünk: megvizsgáltuk, mi történik, ha a nyelvleírásban egyre sikeresebb lexikalizmust a végsőkig fokozzuk; arra kerestük a választ, hogy egy „totálisan" lexikalista nyelvtan eredményes tud-e lenni elméletben, illetve gyakorlatban. A nyelvtan előnyei, hogy a hangsúlyt a szemantikára helyezi, kompozicionális a DRT-vel, egyszintű, homogén, és bármely nyelv leírására egyformán alkalmas. Annak igazolására, hogy a mechanizmusok működnek, készítettünk egy implementációt, amely magyar és angol nyelvű mondatokhoz képes többféle (közük szemantikai) reprezentációt társítani, és a két nyelv között a gépi fordítást is megvalósítja. Működő rendszerünk bizonyíték arra, hogy a kiinduló nyelvtan jól formalizált és egzakt, és hogy alapjain készíthetők számítógépes nyelvészeti alkalmazások. A következő lépés a megközelítés hatékonyságának a vizsgálata, amelyhez az adatbázis méretének jelentős növelése az egyik fontos feladat, és további nyelvi jelenségek kezelésének a részletes kidolgozása a másik. Ezeken fogunk dolgozni a jövőben, hogy bebizonyítsuk, hogy a totálisan lexikalista megközelítést érdemes alkalmazni a nyelvtechnológia területén is.

Summary

From the Theory of Total Lexicalism to the Experimental Implementation

1. Introduction

The paper summarizes a four-year project, whose aim has been to try out a new kind of grammar in theory and in practice alike. The grammar is "totally" lexicalist, which means that all the information (needed for a sentence to be put together) is stored in the lexicon, thus language-specific syntactic rules are not required. Our main goal has been to legitimize this grammar by creating a working implementation, which is the best evidence for the exactness and formalizability of a system. A Prolog-implementation has been made on this basis (GeLexi project), which can (on a small corpus) decide whether a sentence is grammatical, and can provide morphological, syntactic and semantic representations. The lexical items of the parser are Hungarian and English stems and affixes. By means of two-way application of the program (parsing and generating), machine translation is also achieved. After proving that the mechanisms can work, we decided to switch to a more effective developing environment and continue our work (LiLe project, then ReALIS project) to prove that total lexicalism is worth applying to computational linguistic tasks.

Section 2 gives an overview of language technology, goals and methods, Section 3 is about lexicalist grammars in theory (the advantages of the approach) and in practice (the success of unification-based parsers). Section 4 introduces our work (GeLexi project): the starting point (total lexicalism), why we considered that this field is worth doing research into; what our goals and expectations have been; and finally our achievements with examples from the program. Section 5 is about the limitations this approach might have, and the possible solutions to these problems (LiLe project); and the directions of further work (ReALIS project) are discussed in Section 6. Finally to conclude, the significance of our results is explained in Section 7: why we find total lexicalism suitable for computational linguistic applications.

2. Language technology

The field of computational linguistics was born in the 1950's, when the need for machine translation first emerged and seemed possible to achieve. After a few years, it became evident that the task is more difficult than it was first considered. In order to achieve the main goal (machine translation), the problem had to be split to various tasks: segmentation, categorization, parsing, disambiguation, anaphora resolution, generation, etc. Furthermore, strictly formalized linguistic theories had to be elaborated: the mathematical bases of linguistics had to be worked out. Chomsky's grammar hierarchy (Partee et al. 1990) was the first step toward this goal. Natural languages are claimed to be mildly context-sensitive, which means that they are almost context-free, there are only a few phenomena which cannot be handled with context-free formalisms. Several context-free and mildly context-sensitive grammars have been elaborated to describe natural language, such as indexing grammars, tree-adjoining grammars or categorial grammars.

Computational linguistic applications are created for several goals, such as machine translation, information retrieval, information extraction, question answering, text summarization, etc. (Mitkov 2003). These systems have to deal with various phenomena: morphology, syntax, and – for intelligent applications – semantics, even pragmatics and

discourse. Several of them aim to be more or less universal, in order to be able to parse texts in different languages.

There are various methods and techniques for achieving these goals: some systems define rules, others are based on (more or less annotated) corpora; several applications use statistical methods for parsing rather than linguistic issues; some programs use only shallow parsing (partial analyses), which takes less time and resource, while others aim at more precise analyses, so they use full (deep) parsing (fine-grained analyses). The main advantage of the corpus-based approach is that it takes much less time to get much more data, but the annotation also takes time and effort, not every possible expressions can be found in the database, and it is not really suitable for intelligent applications. Rule-based systems require much more time and effort to compile enough data, but do not have the above mentioned limitations. A further advantage of the rule-based approach is the re-usability of the systems: e.g. programs developed for parsing can be applied to question-answering or machine translation. The advantages of shallow (chunk-based) parsing are that it is highly robust, and (due to the lower complexity of analysis) highly efficient, but it is less precise and accurate. Deep (full) parsing is highly precise, but less robust, and (due to the higher complexity of analysis) less efficient (Frank 2003). The most successful applications are the hybrid ones, because linguistic knowledge (and full parsing) is obviously needed if accuracy is important, but for effectiveness and robustness statistical methods and corpora (and sometimes even shallow parsing) also seem inevitable.

3. Lexicalist grammars

In the last decades, *lexicalism* became an important issue in generative linguistics. It has always been admitted that a grammar needs a lexicon, where the words can be found with some of their properties. Later the importance of this lexicon has increased: more and more features became part of the lexicon, thus less and less rules the syntactic component had to contain.

There are several advantages of the lexicalist approach. Most importantly, lexicalist grammars are *declarative*, which means that they are *constraint-based* as opposed to derivation-based (transformational) theories. Properties and requirements are stored in the lexicon for every lexical item, which have to be unified during the analysis. A sentence is grammatical if all the requirements are met, that is, when *unification* is successful. Another positive feature is that lexicalist grammars are *monostratal*, which means that all the constraints take effect simultaneously; the order in which they apply is irrelevant. Finally, in a lexicalist theory assigning *semantic representation* can be achieved more easily. The best known lexicalist theories are LFG (Lexical Functional Grammar), HPSG (Head-Driven Phrase Structure Grammar), LTAG (Lexicalized Tree Adjoining Grammar), Categorial Grammar, and Construction Grammar.

Lexicalism proved to be successful not only in theory but in the field of language technology as well. “The main advantage of unification and constraint-based grammars is the simplification of the rules (and hence the computational processes) of analysis, transformation, and generation. Instead of a series of complex multi-level representations there are mono-stratal representations and/or simple lexical transfer; and the syntactic orientation which characterized previous transfer systems is replaced by lexicalist solutions. In addition, grammars are in principle *reversible*; the same formalism can in theory be applied in both analysis and synthesis.” (Mitkov 2003: 509)

Parsers based on lexicalist grammars can provide more detailed analysis, they can handle languages with rich morphology and free word-order as well, and the outputs of these analyses can be parallel, thus machine translation can be achieved more easily. Previously

existed parsers did not turn out to be sufficient enough for intelligent applications such as question answering, text summarization or good-quality machine translation. Deep-linguistic methods seemed to be indispensable for completing tasks like these. Grammars using only phrase-structure usually have difficulties with languages like Hungarian (rich morphology, almost free word-order), and they sometimes have too complicated and very different rule-systems for various languages. Lexicalist approaches seem to avoid these problems. Furthermore, it is widely accepted that a system can only be suitable for intelligent applications if a semantic representation is assigned to a sentence: which some unification-based programs can accomplish. Coverage has been a secondary issue (many of these applications are still in experimental phase), but some of these parsers has actually reached the coverage of parsers using shallow techniques and statistical methods.

Two of these systems are certainly worth mentioning. The Parallel Grammar project (Butt et al. 2002) uses LFG as a framework for "parallel" descriptions of various languages for multilingual NLP tasks. Large-scale LFG grammars have been developed for e.g. English, French, German, Japanese, or Urdu, and grammars of several other languages are being worked out (Hungarian as well)[54]. The HPSG-based DELPH-IN (Deep Linguistic Processing with HPSG Initiative, e.g. Bond et al. 2005) uses a combination of linguistic and statistical processing methods to get the meaning of texts[55]: grammars like ERG (English Resource Grammar, which is the largest HPSG-based grammar for English, Copestake–Flickinger 2000), or JACY (Japanese grammar, Siegel–Bender 2004), development environments like the LKB (Linguistic Knowledge Building, Copestake 2002), or the PET System, a "starter-kit" to make grammar-writing easier (Grammar Matrix, Bender et al. 2002), etc. Some of these systems also contain a semantic component, using MRS (Minimal Recursion Semantics, Copestake et al. 2005), which was first developed for HPSG, but later on other lexicalist grammars started integrating it into their system, such as LFG or LTAG.

Machine translation (as the "ultimate goal" of most computational linguistic projects) is aimed within these projects as well: partly to prove the universality of their formalism, and partly for practical reasons: to create good-quality translations, which has not really been an issue earlier. The results are promising, though most of these systems have a rather small database (so far). There is an application which uses both LFG and HPSG formalisms, the Norwegian–English semantic transfer-based machine translation system: LOGON (Lønning et al. 2004). It has been developed for tasks where good-quality translation is more important than coverage. The first step is the (syntactic and semantic) analysis of the Norwegian sentence by NorGram (member of the LFG-based ParGram), the output is a language-specific semantic representation (an MRS), then these representations are transferred into an English-like MRS, and finally the English sentence is generated by an HPSG-based parser (generator), using ERG and LKB.

4. GeLexi project

Total lexicalism – starting point

Among lexicalist approaches there are some where the lexicon has gained so much importance that there are only very few syntactic rules, and phrase-structure does not really carry any information. For instance, Karttunen (1986) introduces *radical lexicalism* by using a *unificational categorial grammar* (UCG). In this grammar the only syntactic operation is function application, most of the information is stored in the lexicon, and grammaticality of

[54] http://www2.parc.com/isl/groups/nltt/pargram/
[55] http://www.delph-in.net/

sentences can be decided by means of unification. According to Karttunen (1986: 19) UCG trees look like PS trees but they are only "analysis trees"; and he adds "all that matters is the resulting [morphological] feature set." This grammar is especially suitable for phenomena like nonlocal dependencies and languages with free word order. (Phrase-structure grammars usually have difficulties with both of them.)

Alberti (1999) takes this idea even further, and defines a *totally lexicalist* grammar (GASG), which is a modified unificational categorial grammar. From this grammar even function application is omitted, thus *unification* remains the only operation. This yields to a lexicon richer than any earlier one, where all the information is stored in the descriptions of lexical items, and there is no need for language-specific phrase-structure rules.[56]

The motivation for elaborating this grammar lies in the need for fulfilling the compositionality principle. Finding a compositional semantics compatible with the Chomskyan syntax could not really succeed (e.g. lambda-operation seemed to be too strong a tool). Alberti (1999) tried the other way: to create a syntax which is compositional with a well-functioning semantic theory, DRT (van Eijck–Kamp 1997); a syntax which lacks hierarchy just like its semantic counterpart.

The success of lexicalist approaches encourages us to keep trying out the "extreme" possibility of total lexicalism: can a grammar be developed (in theory and in practice) if only lexicon exists, syntax (PS trees) does not? Schneider (2005) raises the idea of reducing constituent-structure from LFG-representations as much as possible in order to make the parsing simpler (this seems to be especially useful in the case of languages with free word order), since the necessary information is fully encoded in the functional structure. The only reason for keeping constituent-structure is its context-freeness. In GASG constituent-structure is completely eliminated, only dependency relations remain. However, since there are restrictions on word order, the parsing algorithm still can stay polynomial (contrary to "pure" dependency grammars).

The aim of our research team has been to legitimate this grammar, and to try out whether total lexicalism is worth applying to computational linguistic tasks. Karttunen (1986) proved that using a unificational categorial grammar (in theory) is very efficient in the case of agglutinative languages. This could predict that an implementation of a totally lexicalist grammar would work well in the case of Hungarian. Nevertheless, we had to prove that the idea can be applied to completely different languages as well; this is why we have also added English lexical items to the database.

The background grammar is GASG (Generative/Generalized Argument Structure Grammar, Alberti 1999), a homogeneous, monostratal, unification-based theory. Monostratal grammars are considered to be more effective at the parsing process: the unification of two elements takes more time and effort, but false solutions are excluded sooner. GASG is not only monostratal but homogeneous as well. One of our goals is to try out whether this property increases or decreases efficiency.

In GASG descriptions of lexical items have four components: the own features of the element (e.g. its part of speech category), its requirements in a sentence (properties of possible arguments), semantic description (a proto-DRS), and the connection between syntax and semantics. A sentence is grammatical if all the requirements of the given lexical items are met, that is, when unification is successful.

Without building phrase-structure trees GASG could be regarded as a dependency grammar, which is not effective computationally: it is proved that without restrictions to word order, the parsing algorithm is exponential. But this does not stand in the case of

[56] This intention coincides with two mottoes of Joshi's (Joshi 2003): "Complicate Locally, Simplify Globally", and "Grammar ≈ Lexicon".

GASG, because there is a special requirement which is in charge of word order, namely *rank parameters*. We assume that every word wants to be next to every other word if they are in a syntactic or semantic relation in a sentence. Obviously, all these requirements cannot be met; this is why they should be ranked (the strongest wins). This approach can easily explain word order differences *between* languages as well: ranks are different. For example, in Hungarian free adverbs can appear before, after or in-between the arguments (scrambling), while in English only at the beginning or at the end of a sentence. The explanation is that in Hungarian the ranks of arguments and free adverbs are equal, but in English the rank of a free adverb is weaker then the rank of any of the arguments.

We have made the implementation in Prolog, which is suitable for computational linguistic tasks if the database is not required to be large. Hence our goal has been to try out lexicalist methods, and not to produce a software; we did not need a huge lexicon, only a few hundred entries. Our program can parse Hungarian and English sentences, and can provide morphological, syntactic and semantic representations. We have also elaborated a totally lexicalist approach to machine translation, which is achieved by the two-way application of the program (parsing and generating).

During the implementation we found that instead of the original view, that lexical items are fully inflected words, a better alternative is to assume that every morpheme is a separate lexical item with all their properties. There are two main reasons for that: first, it is much more effective in the case of languages with rich morphology like Hungarian (the lexicon has to contain much less entries), and second, in these languages it is more natural to assume that some syntactic relations are established by affixes (which are often stems in other languages, e.g. the causative morpheme is a suffix in Hungarian, but a stem in English). So the implementation has had effects on the theory as well, resulting in an even more homogeneous grammar.

Expectations

Our starting aim has been to prove that GASG is an exact, strictly formalized grammar, and we assumed that a working implementation could be the best evidence for that. While implementing the grammar in Prolog, we found that totally lexicalist methods can be very useful in language technology, especially in the case of languages like Hungarian, which is an agglutinative language with (almost) free word order. But we have also added English lexical items to the database to prove that the methods work for other kinds of languages as well.

Our practical aim has been to develop a parser, which can decide whether a sentence is grammatical or not, and (in the case of grammatical sentences) can print out various representations: morphological (the relevant lexical items), syntactic (relations among them) and semantic (a DRT). The most important component is semantics. This is not just because few computational systems contain semantic representations (mainly lexicalist parsers), but because this semantic output can be regarded as a machine-aided translation. Learning to read these DRT-like representations (which are in English) takes only about an hour, while learning, say, Hungarian takes years.

Finally, we aimed to work out a mechanism for machine translation based on total lexicalism, and we used the built-in generating function of Prolog for this task. We assumed that this can be done through the so-called copredicative network (which is a level between syntax and semantics) by means of the two-way application of our program: parsing source language sentences, then generating target language sentences (which includes their parsing as well – checking grammaticality).

So our main purpose has been to prove that GASG is an exact, strictly formalized grammar, and that totally lexicalist methods are worth applying in language technology. We

intended to accomplish this task by making an implementation in Prolog. We chose using smaller database but adding various kinds of lexical items in Hungarian and English. We have not aimed at making a huge lexicon and so a marketable software so far, enlarging the size of the database could be our next step.

Achievements

The present parser is in Prolog, and can carry out three tasks. It (1) decides (on a small Hungarian and English corpus) whether a sentence is grammatical or not, (2) assigns various types of representations to grammatical sentences, and (3) translates these sentences from Hungarian into English and vice versa.

The input of the parser is a series of words. The program first checks whether the words are well-formed. To accomplish this task, it has to segment the words into morphemes (lexical items) and check all the (morpho)phonological requirements these elements have. If unification is successful, the list of the relevant lexical items is printed out.

In Hungarian, words (especially suffixes) can appear in several surface forms, so (morpho)phonological requirements can be very complicated. For example, the accusative suffix has five allomorphs: *-t*, *-ot*, *-at*, *-et*, and *-öt*. It depends on several factors: the sound right before the suffix, the frontness of the stem, the roundness of the previous morpheme, and a so-called lowering property (which is a lexical property, so cannot be calculated on the basis of the phonological form of the morpheme). This is why the *own word* of a lexical item (how it appears in a sentence) often contains variables.

Let us see a simple example, first in Hungarian. If the grammaticality of (1) is asked, the parser finds it correct, and prints out (2), the list of the relevant lexical items.

(1) Péter énekel-tet-het-i Mari-t.
Peter sing-CAUSE-MAY-3SG Mary-ACC
'Peter may make Mary sing.'

(2) LEXICAL ITEMS:

```
Péter: n(1,1,li(m("","Péter",""),labstem("Peter",phonfst(1,2,0,2),1,[])))

énekel: n(2,1,li(m("","énekel",""),labstem("sing",phonfst(1,2,2,2),2,[["NOM"]])))
tet: n(2,2,li(m("t","A","t"),labder("cause",phonfsu(2,2,0.2,2),2, ac(-1,0,1))))
het: n(2,3,li(m("h","A","t"),labsuff("may",phonfsu(1,1,1,2),2,1)))
i: n(2,4,li(m("","i",""),labsuff("sg3obj+def",phonfsu(1,3,1,3),2,3)))

Mari: n(3,1,li(m("","Mari",""),labstem("Mary",phonfst(2,2,0,2),1,[])))
t: n(3,2,li(m("V","t",""),labsuff("ACC",phonfsu(1,1,1,3),1,4)))
```

Each lexical item gets a numbering in the sentence, which makes parsing simpler. For example, in the case of the allomorph *-het* it is (2,3), which means that *-het* is the third morpheme of the second word. After this numbering the own word can be seen with variables (capital letters), which is divided into three parts for technical reasons.

Finally a label can be seen, which is different in the case of stems (*labstem*), derivative elements (*labder*), and other kinds of suffixes (*labsuff*). The reason for that is the difference between the relevant properties we need to store. For instance, with stems the important phonological features are the frontness and roundness of the item, and whether it is a lowering stem or not. In the case of suffixes, the question is whether the suffix causes various stem alternations: lengthening, shortening, vowel-zero alternation or lowering (1 stands for yes, 2 stands for no, 3 stands for irrelevant). Besides, we need to store part-of-speech categories (with each type), argument structures (with stems and derivative element), and rank parameters (with suffixes for the right morpheme order within words).

The next step is syntactic analysis. (3) shows the second output of the program (syntactic relations), first in a more detailed form, where all the relations (established by the morphemes) can be seen, and then a simplified representation is given, where only the relations between the *words* are listed. Figure 1 shows the results in a more readable format.

```
(3) SYNTAX:

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

    regent-noun-subj: énekeltetheti-Péter
    regent-det-subj: énekeltetheti-Péter
    regent-noun-obj: énekeltetheti-Marit
    regent-det-obj: énekeltetheti-Marit
```

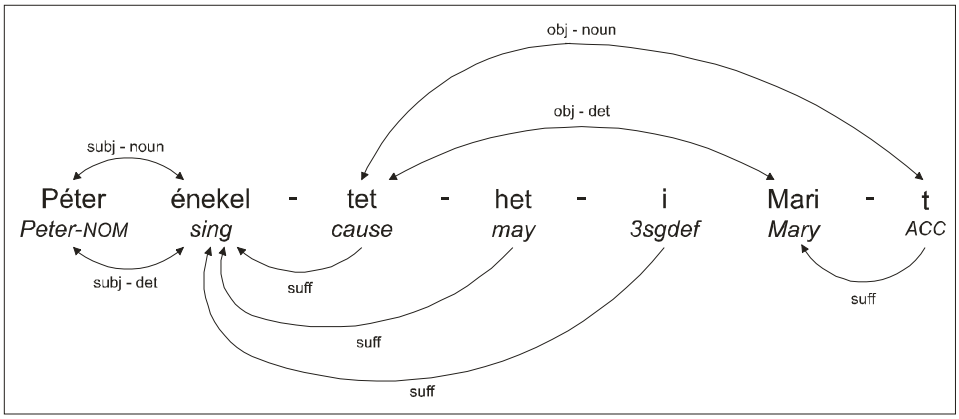


Figure 1: Syntactic relations

Arrows with two heads show the two-way relations (predicate–argument pairs); arrows with one head represent one-way (free) relations. Predicates search for their arguments in two pillars: a nominative and a determining one. In this example arguments are proper names, so these two pillars coincide in the case of the subject. The object of the sentence is found in the accusative suffix *-t*, and the causative *-tAt* needs it in the sentence, because the verb stem is intransitive.

Finally, a semantic representation is printed out. It is a kind of DRS (Discourse Representation Structure, van Eijck–Kamp 1997), but it has additional condition rows as well, which are responsible for building the actual DRS into wider context (LDRS, Alberti 2000). (4) shows the output of the parser, and a DRS-like "box" is put below it, which can help to understand the formalisms.

```
(4) SEMANTICS:
    provref("fixpoint",[e(2,3,1)])
    provref("old",[r(1,1,1)])
    pred("Peter",1,[r(1,1,1)])
    provref("new",[e(2,1,1)])
    pred("sing",2,[e(2,1,1),r(3,1,1)])
    provref("new",[e(2,2,1)])
    provref("=",[e(2,2,1),e(2,1,1)])
```

```
pred("cause",2,[e(2,2,1),r(1,1,1),e(2,1,1)])
provref("new",[e(2,3,1)])
provref("<",[e(2,3,1),e(2,2,1)])
pred("may",2,[e(2,3,1),e(2,2,1)])
provref("old",[r(3,1,1)])
pred("Mary",3,[r(3,1,1)])
```

r1, r3, e21, e22, e23
Peter(r1) Mary(r3) e23: may(e22) e22: cause(r1,e21) e21: sing(r3) ...

It can be seen that there are two entities: r1 and r3 (the numbers refer to the positions of the words in the sentence), one of them is Peter (r1), and the other is Mary (r3); and there are three events: e21, e22, and e23, where e23 is that e22 *may* be true, e22 is that r1 (Peter) causes e21, and e21 is that r3 (Mary) sings. There are other statements in the representation, e.g. that e23 is the fixpoint (tensed element) of the sentence (which is useful e.g. for showing the difference between *The Hungarian girl is pretty*, and *The pretty girl is Hungarian*), and that r1 and r3 are "old" referents, which means that Peter and Mary have to be known in the context (since they are proper names).

When the grammaticality of the English version of this sentence is asked, very similar representations are printed out, only the "names" of the referents (which are given on the basis of the numberings of the morphemes) are different (5).

(5)
```
LEXICAL ITEMS:

Peter:  n(1,1,li(m("","Peter",""),labsteme("Peter",1,[["0"]])))
may:  n(2,1,li(m("","may",""),labsteme("may",2,[["VERB"]])))
make:  n(3,1,li(m("","make",""),labsteme("cause",2,[["NOM","VERB"]])))
Mary:  n(4,1,li(m("","Mary",""),labsteme("Mary",1,[["0"]])))
sing:  n(5,1,li(m("","sing",""),labsteme("sing",2,[["NOM"]])))

SYNTAX:

gr("noun","regent","subj",1,1,3,1)
gr("det","regent","_",1,1,3,1)
gr("regent","verb","arg",2,1,3,1)
gr("regent","noun","subj",3,1,1,1)
gr("regent","det","subj",3,1,1,1)
gr("regent","verb","arg",3,1,5,1)
gr("noun","regent","subj",4,1,5,1)
gr("det","regent","_",4,1,5,1)
gr("regent","noun","subj",5,1,4,1)
gr("regent","det","subj",5,1,4,1)

regent-verb-arg: may-make
regent-noun-subj: make-Peter
regent-det-subj: make-Peter
regent-verb-arg: make-sing
regent-noun-subj: sing-Mary
regent-det-subj: sing-Mary

SEMANTICS:

provref("fixpoint",[e(2,1,1)])
provref("old",[r(1,1,1)])
pred("Peter",1,[r(1,1,1)])
provref("new",[e(2,1,1)])
provref("<",[e(2,1,1),e(3,1,1)])
pred("may",2,[e(2,1,1),e(3,1,1)])
```

```
provref("new",[e(3,1,1)])
provref("=",[e(3,1,1),e(5,1,1)])
pred("cause",3,[e(3,1,1),r(1,1,1),e(5,1,1)])
provref("old",[r(4,1,1)])
pred("Mary",4,[r(4,1,1)])
provref("new",[e(5,1,1)])
pred("sing",5,[e(5,1,1),r(4,1,1)])
```

It is an inherent feature of Prolog that a program can generate anything which it can analyze. Thus machine translation can be achieved by the two-way application of the program: parsing source language sentences and generating target language sentences. The question is which level of representation is the most suitable for being the starting point of the generation. Since we found the syntactic output too language-specific, and the semantic output sometimes does not contain all the information, we created another level, the so-called *copredicative network*, whose purpose is to show which predicates make statements about the same element. Copredicative network is a level between syntax and semantics, where (because of totally lexicalist morphology) most of the differences between languages are already neutralized. (If the translation cannot be done through this level, it can be done through semantics.) (6) and (7) show the copredicative network of the Hungarian and the English version of (1), respectively. It can be seen that the two representations are the same, except for the order of the rows, and the numberings of the morphemes. This is why in this approach, translating from and into completely different languages (like English and Hungarian) is not more difficult than it would be in the case of similar languages.

(6) COPREDICATIVE NETWORK (Hungarian Version):

```
copr("sing",2,1,"Mary",3,1,1,1,"arg")
copr("sing",2,1,"Mary",3,1,1,0,"arg")
copr("cause",2,2,"Peter",1,1,1,1,"arg")
copr("cause",2,2,"Peter",1,1,1,0,"arg")
copr("cause",2,2,"sing",2,1,2,0,"arg")
copr("may",2,3,"cause",2,2,1,0,"arg")
```

(7) COPREDICATIVE NETWORK (English Version):

```
copr("may",2,1,"cause",3,1,1,0,"arg")
copr("cause",3,1,"Peter",1,1,1,1,"arg")
copr("cause",3,1,"Peter",1,1,1,0,"arg")
copr("cause",3,1,"sing",5,1,2,0,"arg")
copr("sing",5,1,"Mary",4,1,1,1,"arg")
copr("sing",5,1,"Mary",4,1,1,0,"arg")
```

For generating the target-language sentence, the relevant lexical items have to be selected. The meaningful morphemes (which have semantic content) can be found in the basis of the copredicative network, but for a grammatical sentence, other kinds of (language-specific) lexical items have to be present as well: the ones responsible for case marking and agreement. There are universal restrictions for the types and places of these elements, so the program has to search only for a limited number of extra morphemes (technically it assumes variables), and only at very specific template positions. Then (to get the grammatical sentence) the program should check different permutations of the lexical items (we use filters to avoid trivially impossible variants) until it finds (the) one compatible with all the requirements for the relevant elements in the target language (target-language parsing).

(8) shows the result of translating (1) into English, and (9) is the translation of the English version of (1) into Hungarian. Because the program stores lexical items, not words, it is not problematic that in the Hungarian version the sentence consists of three words, while the English version contains five.

```
(8) translate_Hun_Eng("Péter énekeltetheti Marit.").
    In English: Peter may make Mary sing.
    yes

(9) translate_Eng_Hun("Peter may make Mary sing.").
    In Hungarian: Péter énekeltetheti Marit.
    yes
```

A more extreme case can be when the subject and the object are not present in the Hungarian sentence either, which is possible, since Hungarian is a pro-drop language. In this case verbal suffixes show the person and the number of the missing elements. This could be even harder for a translator using traditional lexicons. But our parser can assign a semantic representation to a sentence like this as well, so translating it would not be more difficult than translating sentences with spelled-out arguments. In (11) the translation of the simple sentence (10) can be seen, together with the morphological, syntactic and semantic representations.

(10) Szeret-l-ek.
love-2SG.OBJ-1SG.SUBJ
'I love you.'

```
(11)  translate_Hun_Eng_print("Szeretlek.").

    LEXICAL ITEMS:

    szeret:n(1,1,li(m("","szeret",""),labstem("love",phonfst(1,2,2,2),2,[["NOM","ACC"]])))
    l: n(1,2,li(m("","l",""),labsuff("objperson2",phonfsu(3,2,1,1),2,2.5)))
    ek: n(1,3,li(m("V","k",""),labsuff("sg1",phonfsu(1,1,2,3),2,3)))

    SYNTAX:

    gr("suff","stem","free",1,2,1,1)
    gr("suff","stem","free",1,3,1,1)

    COPREDICATIVE NETWORK:

    copr("love",1,1,"sg1",1,3,1,1,"arg")
    copr("love",1,1,"objperson2",1,2,2,1,"arg")

    SEMANTICS:

    provref("fixpoint",[e(1,1,1)])
    provref("new",[e(1,1,1)])
    pred("love",1,[e(1,1,1),r(0,1,1),r(0,1,2)])[57]

    In English: I love you.

    LEXICAL ITEMS:

    I:  n(1,1,li(m("","I",""),labsteme("I",1,[["0","sg","1","NOM"]])))

    love:  n(2,1,li(m("","love",""),labsteme("love",2,[["NOM","ACC"]])))

    you:  n(3,1,li(m("","you",""),labsteme("you",1,[["0","_","2","_"]])))

    SYNTAX:

    gr("noun","regent","subj",1,1,2,1)
    gr("det","regent","_",1,1,2,1)
    gr("regent","noun","subj",2,1,1,1)
    gr("regent","det","subj",2,1,1,1)
```

[57] r011 means "I": a built-in referent (0 shows that) singular (1) and first person (2). r012 means "you": built-in referent, singular, second person.

```
gr("regent","noun","obj",2,1,3,1)
gr("regent","det","obj",2,1,3,1)
gr("noun","regent","obj",3,1,2,1)
gr("det","regent","_",3,1,2,1)

regent-noun-subj: love-I
regent-det-subj: love-I
regent-noun-obj: love-you
regent-det-obj: love-you

COPREDICATIVE NETWORK:

copr("love",2,1,"I",1,1,1,1,"arg")
copr("love",2,1,"I",1,1,1,0,"arg")
copr("love",2,1,"you",3,1,2,1,"arg")
copr("love",2,1,"you",3,1,2,0,"arg")

SEMANTICS:

provref("fixpoint",[e(2,1,1)])
provref("old",[r(1,1,1)])
pred("=",1,[r(1,1,1),r(0,1,1)])
provref("new",[e(2,1,1)])
pred("love",2,[e(2,1,1),r(1,1,1),r(3,1,1)])
provref("old",[r(3,1,1)])
pred("you",3,[r(3,1,1)])

yes
```

So we have made a parser for Hungarian on the basis of GASG, a totally lexicalist grammar. We also added English lexical items to prove that the mechanisms work not only for this particular language. Our program can decide (on a small corpus) whether a sentence is grammatical or not, and can produce various types of representations, among which the most important is semantics. Using the generating function of Prolog, we can also translate from Hungarian into English and vice versa. The program can parse sentences containing various linguistic phenomena, such as morphophonological alternations, ambiguity, causativization, zero pronouns, verbal prefixes, adjectives with arguments, and conjunction.

5. Limitations – LiLe project

While making the implementation in Prolog we came across several difficulties. For instance, sometimes the database needed to be modified, when we found extra properties which should be stored. This could not be an easy task in Prolog. Another disadvantage of this programming language is that its output cannot be easily read.

This is why we decided to rebuild the lexicon as a relational (SQL) database, and so our lexicon has become compatible with the XML-format as well. Because of the new structure, our lexicon has been able to be used in other fields, too. For instance, it can be regarded as a "dynamic corpus". The expression means that this lexicon would not contain the existing words (sentences), like a regular corpus, but the possible ones which could be generated. Furthermore, users could look up not only words (sentences) but elements with particular features as well (e.g. Hungarian lowering stems). Another application could be helping education: teaching foreign language (Hungarian), or grammar.

Another disadvantage of this approach could be the fact, that it is competence-based. More and more linguists think that corpus-based approaches are more promising; or – if a system is competence-based – the rules should be more flexible to be able to handle a text with mistakes as well (Prószéky 2005).

The solution to this problem could be using a special feature which we have already tried in the new system (SQL database). Because of the locality of our approach ("rules" are assigned to lexical items, not the whole language), the grammar is flexible. We can easily

"switch off" any property at any lexical item, so that the set of grammatical sentences would be just a little different.

Finally, the idea of total lexicalism may have a disadvantage as well. Treating general syntactic rules could be problematic, e.g. the rule in English that every sentence has a subject. The question is where to store this feature. Putting it into the description of every verb stem would not be very effective. The solution to this problem can be to find one particular morpheme which the feature should be assigned to.

6. Future work – ReALIS project

Considering these difficulties we would like to make some changes in the future. We plan to improve the semantic component on the basis of Alberti (2005), to be able to handle texts, not only sentences, and to get a more detailed analysis, including a semantic representation more sophisticated than any earlier one. Our ultimate goal is to achieve good-quality machine translation which would also account for rhetorical relations, discourse-functions (topic, focus), or aspect. We believe that this semantic representation is detailed enough to serve as an interlingua, which could make it easier to achieve language-independent machine translation. (Lexicalist approaches like LFG and HPSG usually use transfer-based machine translation, which needs different transfer lexicons for every language pair.)

Furthermore, we plan to switch to a more effective programming language and enlarge the size of the database. Meanwhile, we would like to achieve goals which do not need a large corpus, such as helping education. And finally, we plan to keep working on elaborating various phenomena (derivative system of Hungarian, argument structures, etc.) to prove that total lexicalism can be an effective tool in language technology.

7. Conclusions

The aim of our research team has been to prove that a totally lexicalist grammar can be a well working system in theory and in practice alike. To achieve this goal, we have made a parser in Prolog which can decide the grammaticality of a sentence and can provide morphological, syntactic and semantic representations. Our small database consists of Hungarian and English stems and affixes, and can also translate from Hungarian into English and vice versa.

In the past four years we tried out several linguistic ideas. We experimented with phenomena computational linguistics usually do not do research into. We have equally studied details (e.g. the behavior of Hungarian articles), and larger issues (e.g. translation). We could afford to do so because our aim has not been to produce a marketable software as soon as possible. Our parser obviously needs to be extended, but our results are promising. The significance of our project is that we have showed that morphology-based total lexicalism and representational discourse semantics are worth applying in language technology. We intend to strengthen this view by further research in the future.

Hivatkozások

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

Alberti Gábor (2000): Lifelong Discourse Representation Structures. In: *Gothenburg Papers in Computational Linguistics* 00–5, Sweden, 13–20.

Alberti Gábor (2004): LDRT: Az „ideális hallgató” tudásának reprezentációja. In: László János, Kállai János, Bereczkei Tamás (szerk.): *A reprezentáció szintjei* I., Gondolat, Budapest, 365–382.

Alberti Gábor (2005a): *ReALIS*. Akadémiai doktori értekezés.

Alberti Gábor (2005b): Accessible Referents in “Opaque” Belief Contexts. In: Andreas Herzig, Hans van Ditmarsch (eds.): *Belief Revision and Dynamic Logic*, ESSLLI 2005 (workshop), Edinburgh, 1–8.

Alberti Gábor (2006a): *Matematika a természetes nyelvek leírásában*, Tinta Könyvkiadó, Budapest.

Alberti Gábor (2006b): A szóképzéssel együttjáró vonzatszerkezet-változások rendszere. In: *Nyelvtudományi Közlemények* 103, 75–105.

Alberti Gábor, Balogh Kata (2003): Az eltűnt névelő nyomában. In: Büky László (szerk.): *A mai magyar nyelv leírásának újabb módszerei* VI., SZTE, Szeged, 9–31.

Alberti Gábor, Balogh Kata, Kleiber Judit (2002a): GeLexi Project: Prolog Implementation of a Totally Lexicalist Grammar. In: Dick de Jongh, Marie Nilsenova, Henk Zeevat (eds.): *Proceedings of the Third and Fourth Tbilisi Symposium on Language, Logic and Computation*, ILLC, Amsterdam, and Univ. of Tbilisi.

Alberti Gábor, Balogh Kata, Kleiber Judit, Viszket Anita (2002b): A totális lexikalizmus elve és a GASG nyelvtan-modell. In: Maleczki Márta (szerk.): *A mai magyar nyelv leírásának újabb módszerei* V., SZTE, Szeged, 193–218.

Alberti Gábor, Balogh Kata, Kleiber Judit, Viszket Anita (2003): Total Lexicalism and GASGrammars: A Direct Way to Semantics. In: Alexander Gelbukh (eds.): *Proceedings of CICLing2003*, Springer-Verlag, 37–48.

Alberti Gábor, Balogh Kata, Kleiber Judit, Viszket Anita (2005): Towards a Totally Lexicalist Morphology. In: Kenesei István, Christopher Piñón (eds.): *Approaches to Hungarian* 9, 9–33.

Alberti Gábor, Balogh Kata, Kleiber Judit, Viszket Anita (2007a): A fordítás totálisan lexikalista megközelítése. In: Fóris Ágota és Tóth Szergej (szerk.): *Terminologia et Corpora – Supplementum: Ezerarcú lexikon*, Szombathely, 143–152.

Alberti Gábor, Dóla Mónika, Kántor Gyöngyi, Kleiber Judit, Ohnmacht Magdolna (2007b): ReALIS: a „reális” interpretációs rendszer. In: Alberti Gábor, Fóris Ágota (szerk.): *A mai magyar formális nyelvtudomány műhelyei*, Nemzeti Tankönyvkiadó, Budapest, 139–156.

Alberti Gábor, Kleiber Judit (2001): Világok között az Életfogytiglani DRS-ben. In: Kabán Annamária (szerk.): *Funkcionális mondatperspektíva és szövegszerkesztési stratégia*, Miskolci Egyetemi Kiadó, 35–46.

Alberti Gábor, Kleiber Judit (2003): Extraction of Discourse-Semantic Information from Hungarian Sentences by means of a Totally Lexicalist Grammar. In: Elena Paskaleva, Galia Angelova, Hamish Cunningham, Kalina Bontcheva (eds.): *Information Extraction for Slavonic and Other Central and Eastern European Languages*, Borovets, Bulgaria, 63–69.

Alberti Gábor, Kleiber Judit (2004): The GeLexi MT Project. In: John Hutchins (eds.): *Proceedings of EAMT 2004 Workshop*, Valletta: Univ. of Malta, 1–10.

Alberti Gábor, Kleiber Judit, Viszket Anita (2004a): GeLexi project: Sentence Parsing Based on a GEnerative LEXIcon. In: *Acta Cybernetica* 16, 587–600.

Alberti Gábor, Kleiber Judit, Viszket Anita (2004b): GeLexi projekt: Fordítás totálisan lexikalista alapokon. In: Alexin Zoltán, Csendes Dóra (szerk): *II. Magyar Számítógépes Nyelvészeti Konferencia MSZNY 2004*, Juhász Nyomda, Szeged, 73–80.

Alexin Zoltán, Gyimóthy Tibor, Csirik János (2004): Programcsomag információkinyerési kutatások támogatására. In: Alexin Zoltán, Csendes Dóra (szerk): *II. Magyar Számítógépes Nyelvészeti Konferencia MSZNY 2004*, Juhász Nyomda, Szeged, 41–48.

Babarczy Anna, Gábor Bálint, Hamp Gábor, Kárpáti András, Rung András, Szakadát István (2005): Hunpars: mondattani elemző alkalmazás. In: Alexin Zoltán, Csendes Dóra (szerk): *III. Magyar Számítógépes Nyelvészeti Konferencia MSZNY 2005*, Juhász Nyomda, Szeged, 20–28.

Babarczy Anna, Gábor Bálint, Hamp Gábor, Rung András (2006): Argumentumstruktúrák gépi azonosítása (Szemantikai modul a Hunpars elemzőhöz). In: Alexin Zoltán, Csendes Dóra (szerk): *IV. Magyar Számítógépes Nyelvészeti Konferencia MSZNY 2006*, Juhász Nyomda, Szeged, 139–147.

Balogh Kata, Kleiber Judit (2003): Computational Benefits of a Totally Lexicalist Grammar. In: Pavel Mautner, Vaclav Matousek (eds.): *Text, Speech and Dialogue, Proceedings of TSD2003*, Springer-Verlag, Berlin Heidelberg New York, 114–119.

Bánréti Zoltán (1992): *A mellérendelés.* In: Kiefer Ferenc (szerk.): Strukturális magyar nyelvtan I. Mondattan, Akadémiai Kiadó, Budapest.

Beavers, John (2003): *Documentation: A CCG Implementation for the LKB*. Elérhető: http://citeseer.ist.psu.edu/cache/papers/cs/32253/http:zSzzSzlingo.stanford.eduzSzpubszSzWP-2002-08.pdf/beavers02documentation.pdf

Bender, Emily M., Dan Flickinger, and Stephan Oepen (2002): The Grammar Matrix: An Open-Source Starter-Kit for the Rapid Development of Cross-Linguistically Consistent Broad-Coverage Precision Grammars. In: *Proceedings of COLING 2002 Workshop on Grammar Engineering and Evaluation*, Taipei.

Bender, Emily M., Laurie Poulson, Scott Drellishak, Chris Evans (2007): Validation and Regression Testing for a Cross-linguisic Grammar Resource. In: *Proceedings of the ACL 2007 Workshop on Deep Linguistic Processing*, Prague, 136–143.

Bódis Zoltán, Kleiber Judit, Szilágyi Éva, Viszket Anita (2003): *Nyelvészeti lexikon – oktatási és kutatási adatbázis fejlesztése (LILE projekt)*. Előadás a 'Multimédia az oktatásban' c. konferencián. Elérhető: http://lingua.btk.pte.hu/lile.asp

Bódis Zoltán, Kleiber Judit, Szilágyi Éva, Viszket Anita (2004): LiLe projekt: Adatbázis mint „dinamikus korpusz". In: Alexin Zoltán, Csendes Dóra (szerk): *II. Magyar Számítógépes Nyelvészeti Konferencia MSZNY 2004*, Juhász Nyomda, Szeged, 11–18.

Bond, Francis, Stephan Oepen, Melanie Siegel, Ann Copestake, Dan Flickinger (2005): Open source machine translation with DELPH-IN. In: *Proceedings of the Open-Source Machine Translation Workshop at the 10th Machine Translation Summit*, Phuket, Thailand, 15–22.

Butt, Miriam, Helge Dyvik, Tracy Holloway King, Hiroshi Masuichi, and Christian Rohrer (2002): The Parallel Grammar project. In: *Proceedings of COLING 2002 Workshop on Grammar Engineering and Evaluation*, Taipei.

Cahill, Aoife, Mairéad McCarthy, Josef van Genabith, Andy Way (2002): Parsing with PCFGs and Automatic F-Structure Annotation. In: Miriam Butt and Tracy Holloway King (eds): *Proceedings of the LFG02 Conference*, National Technical University of Athens, CSLI Publications, 76–95.

Cahill, Aoife, Tracy Holloway King, John T. Maxwell III (2007): Pruning the Search Space of a Hand-Crafted Parsing System with a Probabilistic Parser. In: *Proceedings of the ACL 2007 Workshop on Deep Linguistic Processing*, Prague, 65–72.

Clark, Stephen, Julia Hockenmaier, Mark Steedman (2002): Building Deep Dependency Structures with a Wide-Coverage CCG Parser. In: *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, Philadephia, PA, 327–334.

Clément, Lionel, Alexandra Kinyon (2001): XLFG – an LFG parsing scheme for French. In: Miriam Butt and Tracy Holloway King (eds): *Proceeding of the LFG01 Conference*. University of Hong Kong, CSLI Publications, 47–65.

Copestake, Ann (2002): *Implementing Typed Feature Structure Grammars*. Stanford, CA: CSLI Publications.

Copestake, Ann, Dan Flickinger (2000): An open-source grammar development environment and broad-coverage English grammar using HPSG. In: *Proceedings of the Second conference on Language Resources and Evaluation (LREC-2000)*, Athens, Greece.

Copestake, Ann, Dan Flickinger, Ivan Sag, Carl Pollard (2005): Minimal Recursion Semantics: An introduction. In: *Research in Language and Computation* 3(2–3), 281–332.

Copestake, Ann (2007): Semantic composition with (Robust) Minimal Recursion Semantics. In: *Proceedings of the ACL 2007 Workshop on Deep Linguistic Processing*, 73–80.

Czap László, Mátyás János (2003): Beszélő fej. In: Alexin Zoltán, Csendes Dóra (szerk): *Magyar Számítógépes Nyelvészeti Konferencia MSZNY 2003*, Szegedi Tudományegyetem, Egyetemi nyomda, Szeged, 196–201.

Csendes Dóra, Csirik János, Gyimóthy Tibor, Kocsor András (2005): The Szeged Treebank. In: *LNCS series* Vol. 3658, Springer, Berlin, 123–131.

Csendes Dóra, Hatvani Csaba, Alexin Zoltán, Csirik János, Gyimóthy Tibor, Prószéky Gábor, Váradi Tamás (2003): Kézzel annotált magyar nyelvi korpusz: a Szeged Korpusz. In: Alexin Zoltán, Csendes Dóra (szerk): *Magyar Számítógépes Nyelvészeti Konferencia MSZNY 2003*, Szegedi Tudományegyetem, Egyetemi nyomda, Szeged, 238–245.

Dang, Hoa Trang, Karin Kipper, Martha Palmer (2000): Integrating Compositional Semantics into a Verb Lexicon. In: *COLING 2000*, 1011–1015.

Farkas Richárd (2007): Részben felügyelt tanulási módszerek a tulajdonnév felismerében. In: Tanács Attila, Csendes Dóra (szerk): *V. Magyar Számítógépes Nyelvészeti Konferencia MSZNY 2007*, Juhász Nyomda, Szeged, 166–176.

Farkas Richárd, Konczer Kinga, Szarvas György (2004): Szemantikuskeret-illesztés és az IE rendszer automatikus kiértékelése. In: Alexin Zoltán, Csendes Dóra (szerk): *II. Magyar Számítógépes Nyelvészeti Konferencia MSZNY 2004*, Juhász Nyomda: Szeged, 49–53.

Farkas Richárd, Szarvas György (2004): Statisztikai alapú tulajdonnév-felismerő magyar nyelvre. In: Alexin Zoltán, Csendes Dóra (szerk): *II. Magyar Számítógépes Nyelvészeti Konferencia MSZNY 2004*, Juhász Nyomda, Szeged, 136–140.

Farkas Richárd, Szarvas György (2006): Nyelvfüggetlen tulajdonnév-felismerő rendszer, és alkalmazása különböző domainekre. In: Alexin Zoltán, Csendes Dóra (szerk): *IV. Magyar Számítógépes Nyelvészeti Konferencia MSZNY 2006*, Juhász Nyomda: Szeged, 22–31.

Flickinger, Dan and Emily M. Bender (2003): Compositional Semantics in a Multilingual Grammar Resource. In: *Proceedings of the Workshop on Ideas and Strategies for Multilingual Grammar Development, ESSLLI 2003*, Vienna, Austria, 33–42.

Frank, Anette (1999): From parallel grammar development towards machine translation. In: *Proceeding of MT Summit* VII, 134–142.

Frank, Anette (2003): Projecting LFG F-structures from Chunks or (Non-)Configurationality from a different Viewpoint. In: Miriam Butt, Tracy Holloway King (eds.): *The Proceedings of the LFG'03 Conference*, University at Albany, State University of New York, CSLI Publications, 217–237.

Forst, Martin, Jonas Kuhn, Christian Rohrer (2005): Corpus-Based Learning of OT Constraint Rankings for Large-Scale LFG Grammars. In: Miriam Butt, Tracy Holloway King (eds.): *Proceedings of the LFG'05 Conference*, University of Bergen, CSLI Publications, 154–165.

Fujita, Sanae, Francis Bond, Stephan Oepen, Takaaki Tanaka (2007): Exploiting Semantic Information for HPSG Parse Selection. In: *Proceedings of the ACL 2007 Workshop on Deep Linguistic Processing*, 25–32.

Gábor Kata, Héja Enikő (2005): Vonzatok és szabad határozók szabályalapú kezelése. In: Alexin Zoltán, Csendes Dóra (szerk.): *III. Magyar Számítógépes Nyelvészeti Konferencia*, Juhász Nyomda, Szeged, 245–256.

Gábor Kata, Héja Enikő (2006a): Szemantikai igeosztályok tesztelése az MNSZ-ben. In: Alexin Zoltán, Csendes Dóra (szerk): *IV. Magyar Számítógépes Nyelvészeti Konferencia MSZNY 2006*, Juhász Nyomda, Szeged, 147–156.

Gábor Kata, Héja Enikő (2006b): Predikátumok és szabad határozók. In: Kálmán László (szerk.): *KB 120 A titkos kötet*, MTA Nyelvtudományi Intézet, Tinta Könyvkiadó, Budapest, 134–152.

Gábor Kata, Héja Enikő (2007): Igék szemantikai klaszterezése bővítménykereteik alapján. In: Tanács Attila, Csendes Dóra (szerk): *V. Magyar Számítógépes Nyelvészeti Konferencia MSZNY 2007*, Juhász Nyomda, Szeged, 129–137.

Gábor Kata, Héja Enikő, Mészáros Ágnes (2004): Információkinyerés igeneves szerkezetekből. In: Alexin Zoltán, Csendes Dóra (szerk): *II. Magyar Számítógépes Nyelvészeti Konferencia MSZNY 2004*, Juhász Nyomda, Szeged, 54–62.

Gambäck, Björn (2005): Semantic Morphology. In: *Inquiries into Words, Constraints and Contexts: Festschrift in the Honour of Kimmo Koskenniemi on his 60th Birthday,* CSLI Studies in Computational Linguistics, CSLI Publications, Stanford, California, 204–213.

Gyarmati Ágnes, Almási Attila, Szauter Dóra (2006): A melléknevek beillesztése a Magyar WordNetbe. In: Alexin Zoltán, Csendes Dóra (szerk): *IV. Magyar Számítógépes Nyelvészeti Konferencia MSZNY 2006*, Juhász Nyomda, Szeged, 117–126.

Halassy Béla (1994): *Az adatbázis-tervezés alapjai és titkai. Avagy az út az adattól az adatbázison át az információig;* IDG, Budapest.

Halácsy Péter, Kornai András, Németh László, Rung András, Szakadát István, Trón Viktor, Varga Dániel (2004): Hunglish: nyílt statisztikai magyar–angol gépi nyersfordító. In: Alexin Zoltán, Csendes Dóra (szerk): *II. Magyar Számítógépes Nyelvészeti Konferencia MSZNY 2004*, Juhász Nyomda, Szeged, 81–84.

Halácsy Péter, Kornai András, Németh László, Sass Bálint, Varga Dániel, Váradi Tamás, Vonyó Attila (2005): A hunglish korpusz és szótár. In: Alexin Zoltán, Csendes Dóra (szerk.): *III. Magyar Számítógépes Nyelvészeti Konferencia*, Juhász Nyomda, Szeged, 134–143.

Hócza András (2004): Teljes mondatszintaxis tanulása és felismerése. In: Alexin Zoltán, Csendes Dóra (szerk): *II. Magyar Számítógépes Nyelvészeti Konferencia MSZNY 2004*, Juhász Nyomda, Szeged, 127–135.

Hodász Gábor, Gőbler Tamás (2003): Nyelvi tudásra épülő fordítómemória. In: Alexin Zoltán, Csendes Dóra (szerk): *Magyar Számítógépes Nyelvészeti Konferencia MSZNY 2003*, Szegedi Tudományegyetem, Egyetemi nyomda, Szeged, 261–266.

Homola, Petr, Vladislav Kuboň (2004): A Translation Model for Languages of Acceding Countries. In: John Hutchins (ed.): *Broadening Horizons of Machine Translation and its Applications. Proceedings of the Ninth EAMT workshop*, Foundation for International Studies, University of Malta, Valletta, 90–97.

Hopkins, Mark, Jonas Kuhn (2007): Deep Grammars in a Tree Labeling Approach to Syntax-based Statistical Machine Translation. In: *Proceedings of the ACL 2007 Workshop on Deep Linguistic Processing*, 33–40.

Joshi, Aravind K. (2003): Starting with complex primitives pays off. In: Alexander Gelbukh (ed.): *Proceedings of CICLing2003*, Springer-Verlag. 1–10.

Kálmán László, Balázs László, Erdélyi Szabó Miklós (2003): Tudásalapú természetesnyelv-feldolgozás. In: Alexin Zoltán, Csendes Dóra (szerk): *Magyar Számítógépes Nyelvészeti Konferencia MSZNY 2003*, Szegedi Tudományegyetem, Egyetemi nyomda, Szeged, 109–114.

Kálmán László, Rádai Gábor (2001): *Dinamikus szemantika*. Osiris, Budapest.

Kálmán László, Trón Viktor, Varasdi Károly (szerk.) (2002): *Lexikalista elméletek a nyelvészetben*. Tinta Könyvkiadó, Budapest.

Kaplan, Ronald M., Joan Bresnan (1982): Lexical-functional grammar: A formal system for grammatical representation. In Joan Bresnan (ed.): *The Mental Representation of Grammatical Relations*, MIT Press, Cambridge MA, 173–281.

Kaplan, Ronald M., Tracy Holloway King (2003): Low-Level Mark-Up and Large-scale LFG Grammar Processing. In: Miriam Butt, Tracy Holloway King (eds.): *The Proceedings of the LFG'03 Conference*, University at Albany, State University of New York, CSLI Publications, 238–249.

Karttunen, Lauri (1986): *Radical Lexicalism,* Report No. CSLI 86–68, Stanford.

Kis Balázs, Naszódy Mátyás, Prószéky Gábor (2003): Komplex (magyar) szintaktikai elemző rendszer mint beágyazott rendszer. In: Alexin Zoltán, Csendes Dóra (szerk): *Magyar Számítógépes Nyelvészeti Konferencia MSZNY 2003*, Szegedi Tudományegyetem, Egyetemi nyomda, Szeged, 145–150.

Kleiber Judit (2005): Across world(let)s in a representationist interpretation system. In: Gervain Judit (ed.): *Proceedings of the Tenth ESSLLI Student Session*.

Kleiber Judit (2006): Szótár totálisan lexikalista alapokon. In: Kassai Ilona (szerk.): *Szakszó, szaknyelv, szakmai kommunikáció*, Nyelvészeti Doktorandusz Füzetek 3, 18–31.

Kleiber Judit (2007a): Total Lexicalism in Language Technology. In: Ville Nurmi, Dmitry Sustretov (eds.): *Proceedings of the Twelfth ESSLLI Student Session*, 149–160.

Kleiber Judit (2007b): Számítógépes nyelvészet Pécsen. In: Alberti Gábor, Fóris Ágota (szerk.): *A mai magyar formális nyelvtudomány műhelyei*, Nemzeti Tankönyvkiadó, Budapest, 170–188.

Komlósy András (2001): *A lexikai-funkcionális grammatika mondattanának alapfogalmai*. Tinta Könyvkiadó, Budapest.

Kuti Judit, Vajda Péter, Varasdi Károly (2005): Javaslat a magyar igei WordNet kialakítására. In: Alexin Zoltán, Csendes Dóra (szerk): *III. Magyar Számítógépes Nyelvészeti Konferencia MSZNY 2005*, Juhász Nyomda, Szeged, 79–87.

Kuti Judit, Varasdi Károly, Cziczelszki Judit, Gyarmati Ágnes, Nagy Anikó, Tóth Marianna, Vajda Péter (2006): Igei wordnet és igei eseményszerkezet ábrázolása. In: Alexin Zoltán, Csendes Dóra (szerk): *IV. Magyar Számítógépes Nyelvészeti Konferencia MSZNY 2006*, Juhász Nyomda, Szeged, 97–108.

Lascarides, Alex, Nicholas Asher (1993): Temporal Interpretation, Discourse Relations, and Common Sense Entailment. In: *Linguistics and Philosophy* 16, 437–493.

Lønning, Jan Tore, Stephan Oepen, Dorothee Beermann, Lars Hellan, John Carroll, Helge Dyvik, Dan Flickinger, Janne Bondi Johannessen, Paul Meurer, Torbjørn Nordgård, Victoria Rosén, Erik Velldal (2004): LOGON. A Norwegian MT effort. In: *Proceedings of the Workshop in Recent Advances in Scandinavian Machine Translation*, Uppsala, Sweden.

Marimon, Montserrat, Núria Bel, Sergio Espeja, Natalia Seghezzi (2007): The Spanish Resource Grammar: pre-processing strategy and lexical acquisition. In: *Proceedings of the ACL 2007 Workshop on Deep Linguistic Processing,* 105–111.

Mártonfi Attila (2003): Próbák és példák a Magyar értelmező kéziszótár (2. kiadás, 2003) rejtett információinak feltárására. In: Alexin Zoltán, Csendes Dóra (szerk): *Magyar Számítógépes Nyelvészeti Konferencia MSZNY 2003*, Szegedi Tudományegyetem, Egyetemi nyomda, Szeged, 8–13.

Merényi Csaba (2005): A MetaMorpho magyar–angol gépi fordító rendszer igei vonzatkereteit működtető nyelvtan. In: Alexin Zoltán, Csendes Dóra (szerk): *III. Magyar Számítógépes Nyelvészeti Konferencia MSZNY 2005*, Juhász Nyomda, Szeged, 108–115.

Mihálcz Márton (2003): Magyar főnévi WordNet-ontológia létrehozása automatikus módszerekkel. In: Alexin Zoltán, Csendes Dóra (szerk): *Magyar Számítógépes Nyelvészeti Konferencia MSZNY 2003*, Szegedi Tudományegyetem, Egyetemi nyomda, Szeged, 153–158.

Mihálcz Márton (2004): Angol–magyar gépi fordítórendszer támogatása jelentés-egyértelműsítő modullal. In: Alexin Zoltán, Csendes Dóra (szerk): *II. Magyar Számítógépes Nyelvészeti Konferencia MSZNY 2004*, Juhász Nyomda, Szeged, 92–99.

Mihálcz Márton (2005): Magyar EuroWordNet projekt: bemutatás és helyzetjelentés. In: Alexin Zoltán, Csendes Dóra (szerk): *III. Magyar Számítógépes Nyelvészeti Konferencia MSZNY 2005*, Juhász Nyomda, Szeged, 68–78.

Mihálcz Márton, Naszódi Mátyás, Vajda Péter, Varasdi Károly (2007): NP-koreferenciák feloldása magyar szövegekben a Magyar WordNet ontológia segítségével. In: Tanács Attila, Csendes Dóra (szerk): *V. Magyar Számítógépes Nyelvészeti Konferencia MSZNY 2007*, Juhász Nyomda, Szeged, 138–146.

Mitkov, Ruslan (szerk.) (2003): *The Oxford Handbook of Computational Linguistics*. Oxford University Press, New York.

Montague, Richard (1970): English as a formal language. In Richmond H. Thomason (ed.): *Formal Philosophy: Selected Papers of Richard Montague*, New Haven and London, Yale Univ. Press, 1974.

Németh László, Halácsy Péter, Kornai András, Trón Viktor (2004): Nyílt forráskódú morfológiai elemző. In: Alexin Zoltán, Csendes Dóra (szerk): *II. Magyar Számítógépes Nyelvészeti Konferencia MSZNY 2004*, Juhász Nyomda, Szeged, 163–171.

Novák Attila (2003): Milyen a jó Humor? In: Alexin Zoltán, Csendes Dóra (szerk): *Magyar Számítógépes Nyelvészeti Konferencia MSZNY 2003*, Szegedi Tudományegyetem, Egyetemi nyomda, Szeged, 138–143.

Novák Attila, Nagy Viktor, Oravecz Csaba (2003): Magyar ismeretlenszó-elemző program fejlesztése. In: Alexin Zoltán, Csendes Dóra (szerk): *Magyar Számítógépes Nyelvészeti Konferencia MSZNY 2003*, Szegedi Tudományegyetem, Egyetemi nyomda, Szeged, 45–54.

Oravecz Csaba, Varasdi Károly, Nagy Viktor (2004): Többszavas kifejezések számítógépes kezelése. In: Alexin Zoltán, Csendes Dóra (szerk): *II. Magyar Számítógépes Nyelvészeti Konferencia MSZNY 2004*, Juhász Nyomda, Szeged, 141–154.

Orosz Kata (2006): Főnevek szemantikai jegyei és kódolásuk a MetaMorpho projektben. In: Alexin Zoltán, Csendes Dóra (szerk): *IV. Magyar Számítógépes Nyelvészeti Konferencia MSZNY 2006*, Juhász Nyomda, Szeged, 157–166.

Pajzs Júlia (2003): A készülő Akadémiai nagyszótár számítógépes vonatkozásai. In: Alexin Zoltán, Csendes Dóra (szerk): *Magyar Számítógépes Nyelvészeti Konferencia MSZNY 2003*, Szegedi Tudományegyetem, Egyetemi nyomda, Szeged, 203–210.

Partee, Barbara H., Alice ter Meulen, Robert E. Wall (1990): *Mathematical Methods in Linguistics*. Kluwer Academic Publishers, Dordrecht, The Netherlands.

Pinker, Stephen (1999): *Words and Rules: The Ingredients of Language*. New York, HarperCollins.

Pohl Gábor (2003): Szövegszinkronizációs módszerek, hibrid bekezdés- és mondatszinkronizációs megoldás. In: Alexin Zoltán, Csendes Dóra (szerk): *Magyar Számítógépes Nyelvészeti Konferencia MSZNY 2003*, Szegedi Tudományegyetem, Egyetemi nyomda, Szeged, 254–259.

Pohl Gábor (2004): Iteratív bekezdés- és mondatszinkronizáló. In: Alexin Zoltán, Csendes Dóra (szerk): *II. Magyar Számítógépes Nyelvészeti Konferencia MSZNY 2004*, Juhász Nyomda, Szeged, 117–123.

Pohl Gábor (2006): A Morpho TM főnévicsoport-szinkronizáló módszereinek továbbfejleszése és vizsgálata. In: Alexin Zoltán, Csendes Dóra (szerk): *IV. Magyar Számítógépes Nyelvészeti Konferencia MSZNY 2006*, Juhász Nyomda, Szeged, 190–201.

Pohl Gábor, Ugray Gábor (2004): Angol címek felismerése. In: Alexin Zoltán, Csendes Dóra (szerk): *II. Magyar Számítógépes Nyelvészeti Konferencia MSZNY 2004*, Juhász Nyomda, Szeged, 155–160.

Prószéky Gábor (2003a): A természetes nyelvek formális modelljeiről. In: Alexin Zoltán, Csendes Dóra (szerk): *Magyar Számítógépes Nyelvészeti Konferencia MSZNY 2003*, Szegedi Tudományegyetem, Egyetemi nyomda, Szeged, 124–129.

Prószéky Gábor (2003b): Automatikus információszerzés gazdasági rövidhírekből. In: Alexin Zoltán, Csendes Dóra (szerk): *Magyar Számítógépes Nyelvészeti Konferencia MSZNY 2003*, Szegedi Tudományegyetem, Egyetemi nyomda, Szeged, 161–166.

Prószéky Gábor (2005): A világháló nyelvi vizsgálata. In: Alexin Zoltán, Csendes Dóra (szerk): *III. Magyar Számítógépes Nyelvészeti Konferencia MSZNY 2005*, Juhász Nyomda, Szeged, 3–12.

Prószéky Gábor, Kis Balázs (1999): *Számítógéppel – emberi nyelven. Intelligens szövegkezelés számítógéppel.* Szak Kiadó, Bicske.

Prószéky Gábor, Tihanyi László, Ugray Gábor (2004): Moose: a robust high-performance parser and generator. In: John Hutchins (ed.): *Broadening Horizons of Machine Translation and its Applications. Proceedings of the Ninth EAMT workshop*, Foundation for International Studies, University of Malta, Valletta, 138–142.

Romero, Maribel, Laura Kallmeyer (2005): Scope and Situation Binding in LTAG using Semantic Unification. In: *Proceedings of the Sixth International Workshop on Computational Semantics*, Tilburg, 247–258.

Rosén, Victoria, Paul Meurer and Koenraad de Smedt (2005): Constructing a Parsed Corpus with a Large LFG Grammar. In: Miriam Butt and Tracy Holloway King (eds): *Proceedings of the LFG'05 Conference*, University of Bergen, CSLI Publications, 371–387.

Sass Bálint (2005): Vonzatkeretek a Magyar Nemzeti Szövegtárban. In: Alexin Zoltán, Csendes Dóra (szerk): *III. Magyar Számítógépes Nyelvészeti Konferencia MSZNY 2005*, Juhász Nyomda, Szeged, 257–264.

Sass Bálint (2006): Igei vonzatkeretek az MNSZ tagmondataiban. In: Alexin Zoltán, Csendes Dóra (szerk): *IV. Magyar Számítógépes Nyelvészeti Konferencia MSZNY 2006*, Juhász Nyomda, Szeged, 15–21.

Sass Bálint (2007): Élő vagy élettelen? In: Tanács Attila, Csendes Dóra (szerk): *V. Magyar Számítógépes Nyelvészeti Konferencia MSZNY 2007*, Juhász Nyomda, Szeged, 195–203.

Schneider, Gerold (2005): A Broad-Coverage, Representationally Minimal LFG Parser: Chunks and F-structures are Sufficient. In: Miriam Butt and Tracy Holloway King (eds): *Proceedings of the LFG'05 Conference*, University of Bergen, CSLI Publications, 388–407.

Siegel, Melanie, Emily M. Bender (2004): Head-Initial Constructions in Japanese. In: Stefan Müller (ed.): *Proceedings of the HPSG04 Conference*, Center for Computational Linguistics, Katholieke Universiteit Leuven, CSLI Publications, 244–260.

Steedman, Mark, Jason Baldridge (2007): Combinatory Categorial Grammar. In: Robert D. Borsley, Kersti Borjars (eds.): *Non-Transformational Syntax*, Blackwell.

Szilágyi Éva (2005): *Lexikai egységek számítógépes kezelése: képzők, adatbázisban*. Szakdolgozat a PTE BTK Nyelvtudományi Tanszékén.

Szilágyi Éva, Kleiber Judit, Alberti Gábor (2007): A totálisan lexikalista szintaxis rangja(i). In: Tanács Attila, Csendes Dóra (szerk.): *V. Magyar Számítógépes Nyelvészeti Konferencia MSZNY 2007*, Juhász Nyomda, Szeged, 284–287.

Tatai Gábor, Laufer László (2003): Nyelvi elemek érzelmi töltetének vizsgálata és felhasználása természetes nyelvi dialógusrendszerben. In: Alexin Zoltán, Csendes Dóra (szerk): *Magyar Számítógépes Nyelvészeti Konferencia MSZNY 2003*, Szegedi Tudományegyetem, Egyetemi nyomda, Szeged, 102–107.

Tihanyi László (2003): A MetaMorpho projekt története. In: Alexin Zoltán, Csendes Dóra (szerk): *Magyar Számítógépes Nyelvészeti Konferencia MSZNY 2003*, Szegedi Tudományegyetem, Egyetemi nyomda, Szeged, 247–252.

Tihanyi László, Merényi Csaba (2006): A MetaMorpho fordítóprogram projekt 2006-ban. In: Alexin Zoltán, Csendes Dóra (szerk): *IV. Magyar Számítógépes Nyelvészeti Konferencia MSZNY 2006*, Juhász Nyomda, Szeged, 169–179.

Traat, Maarika, Johan Bos (2004): Unificational Combinatory Categorial Grammar: Combining Information Structure and Discourse Representations. In: *Proceedings of the 20th International Conference on Computational Linguistics (COLING '04)*, Geneva, Switzerland, 296–302.

Trón Viktor (2001): *Fejközpontú frázisstruktúra-nyelvtan*. Tinta Könyvkiadó, Budapest.

Trón Viktor, Halácsy Péter, Rebrus Péter, Rung András, Simon Eszter, Vajda Péter (2005): morphdb.hu: magyar morfológiai nyelvtan és szótári adatbázis. In: Alexin Zoltán, Csendes Dóra (szerk): *III. Magyar Számítógépes Nyelvészeti Konferencia MSZNY 2005*, Juhász Nyomda, Szeged, 169–179.

Vajda Péter, Nagy Viktor, Dancsecs Erzsébet (2004): A Ragozási szótártól a NooJ morfológiai moduljáig. In: Alexin Zoltán, Csendes Dóra (szerk): *II. Magyar Számítógépes Nyelvészeti Konferencia MSZNY 2004*, Juhász Nyomda, Szeged, 183–190.

van Eijck, Jan, Hans Kamp (1997): Representing Discourse in Context. In: Johan van Benthem, Alice ter Meulen (eds.): *Handbook of Logic and Language,* Elsevier, Amsterdam, MIT Press, Cambridge, Mass., 179–237.

Váradi Tamás (2003): Főnévi csoport annotációja a CLaRK rendszerrel. In: Alexin Zoltán, Csendes Dóra (szerk): *Magyar Számítógépes Nyelvészeti Konferencia MSZNY 2003*, Szegedi Tudományegyetem, Egyetemi nyomda, Szeged, 65–70.

Váradi Tamás (2006): Részleges gépi fordítás a NooJ rendszerben. In: Alexin Zoltán, Csendes Dóra (szerk): *IV. Magyar Számítógépes Nyelvészeti Konferencia MSZNY 2006*, Juhász Nyomda, Szeged, 202–210.

Váradi Tamás, Gábor Kata (2004): A magyar INTEX fejlesztéséről. In: Alexin Zoltán, Csendes Dóra (szerk): *II. Magyar Számítógépes Nyelvészeti Konferencia MSZNY 2004*, Juhász Nyomda, Szeged. 3–10.

Varasdi Károly, Gábor Kata (2003): Javaslat a magyar igék szemantikájának számítógépes implementációjára. In: Alexin Zoltán, Csendes Dóra (szerk): *Magyar Számítógépes Nyelvészeti Konferencia MSZNY 2003*, Szegedi Tudományegyetem, Egyetemi nyomda, Szeged, 93–98.

Vörös Ferenc, Trepák Mónika (2005): A MobiMouse Plus szótárak készítése közben szerzett tapasztalatokról. In: Alexin Zoltán, Csendes Dóra (szerk): *III. Magyar Számítógépes Nyelvészeti Konferencia MSZNY 2005*, Juhász Nyomda, Szeged, 143–154.

Zeevat, Henk (1987): Combining Categorial Grammar and Unification. In Uwe Reyle, Christian Rohrer (eds.): *Natural Language Parsing and Linguistic Theories*, Dordrecht.

Zhang, Yi, Valia Kordoni, Erin Fitzgerald (2007): Partial Parse Selection for Robust Deep Processing. In: *Proceedings of the ACL 2007 Workshop on Deep Linguistic Processing*, 128–135.

Mellékletek

1. sz. melléklet: A Prolog működése

Az első szöveges melléklet a Prolog működését mutatja be egy egyszerű példán. A téma a családi viszonyok (testvér, szülő, nagyszülő stb.). Többféle Prolog létezik, mi a programot Visual Prologban készítettük. Fontos jellemzője ennek a fejlesztőkörnyezetnek, hogy deklarálni kell a használt predikátumokról, hogy milyen típusú argumentumokat vesznek fel, például hogy a *szülője* predikátumnak két STRING típusú argumentuma van (ki, kinek). További fontos tulajdonsága, hogy a program három részre osztható: az első rész az adatbázis, ahol az szerepel, amit tudunk, vagyis a konstansokat tartalmazó predikátumok (a *database* címszó alatt a deklarációk, a *clauses* alatt az állítások); a második a parancsokat tartalmazó rész, vagyis hogy a további predikátumok argumentumait hogyan kaphatjuk meg (a *predicates* címszó alatt a deklarációk, a *clauses* alatt a parancsok); a harmadik rész pedig az, ahol a célállítás (*goal*) megfogalmazása történik.

Nézzük először az első részt: a *database* címszó alatt először deklaráljuk, hogy a *férfi* és a *nő* predikátumoknak egy, a *házastársa* és a *szülője* predikátumoknak két string típusú argumentuma lehet; majd a *clauses* címszó alatt felsoroljuk az összes igaz állítást az adott predikátumokkal.

```
database
  férfi(STRING)
  nő(STRING)
  házastársa(STRING,STRING)
  szülője(STRING,STRING)

clauses
  férfi("Péter").
  férfi("János").
  férfi("Feri").
  nő("Mari").
  nő("Juli").
  nő("Anna").
  nő("Bori").
  házastársa("Juli","János").
  házastársa("Anna","Feri").
  szülője("Juli","Péter").
  szülője("Juli","Mari").
  szülője("János","Péter").
  szülője("János","Mari").
  szülője("Feri","Juli").
  szülője("Anna","Juli").
  szülője("Bori","János").
```

Ezután a *predicates* címszó alatt deklaráljuk a további használni kívánt predikátumokat, megadjuk például, hogy a *testvére* predikátumnak két string típusú argumentuma lesz. Majd a *clauses* címszó alatt kondicionális típusú állításokat teszünk, a különbség a hagyományos logikai felírástól annyi, hogy a sorrend nem p→q (ha p akkor q), hanem q:–p (q akkor, ha p). Vagyis például az első állítás azt mondja, hogy akkor testvére A a B-nek, ha van egy C, aki mindkettőjükkel *szülője* viszonyban van, és ahol A és B nem azonos (senki sem testvére saját magának). Vagy a második azt mondja, hogy akkor nagymamája A a B-nek, ha van egy C, akinek szülője A, és aki szülője B-nek, illetve ha a *nő* predikátum igaz A-ra. A vessző tehát a logikai *és*-nek (konjunkciónak) felel meg, a logikai *vagy*-nak (alternációnak) pedig a

pontosvessző, ez látható az *unokája* predikátum esetében: akkor unokája A a B-nek, ha nagymamája B az A-nak, vagy ha nagypapája B az A-nak.

```
predicates
  nondeterm testvére(STRING,STRING)
  nondeterm nagymamája(STRING,STRING)
  nondeterm nagypapája(STRING,STRING)
  nondeterm anyósa(STRING,STRING)
  nondeterm apósa(STRING,STRING)
  nondeterm unokája(STRING,STRING)

clauses
  testvére(A,B):-
   szülője(C,A), szülője(C,B), not(A=B).

  nagymamája(A,B):-
   szülője(A,C), szülője(C,B), nő(A).

  nagypapája(A,B):-
   szülője(A,C), szülője(C,B), férfi(A).

  anyósa(A,B):-
   nő(A), szülője(A,C), házastársa(B,C).

  anyósa(A,B):-
   nő(A), szülője(A,C), házastársa(C,B).

  apósa(A,B):-
   férfi(A), szülője(A,C), házastársa(B,C).

  apósa(A,B):-
   férfi(A), szülője(A,C), házastársa(C,B).

  unokája(A,B):-
   nagymamája(B,A); nagypapája(B,A).
```

A *goal* címszó alá kell beírni azt, amit a programtól kérdezünk. Egyszerre csak egy célállítást futtathatunk, a többi elé %-et tettem, ami a kommentezés (egyik) módja ebben a Prologban. Idemásoltam azt is, mit válaszol a program az adott kérdésre. Az első célállítás arra kérdez rá, hogy testvére-e Péter Marinak, amire a `yes` választ kapjuk. Ha a célállításban változó szerepel (nagybetű és nincs idézőjelben), akkor a program nem *ellenőriz*, hanem *generál*: kiadja a lehetséges megoldásokat. Sikeresen kiszámolja, hogy Péternek két nagymamája van: Anna és Bori, hogy Jánosnak nincs nagypapája (a modellben), hogy az *anyósa* predikátum két párra igaz (Bori–Juli és Anna–János); majd egy újabb ellenőrzés következik, azt kérdezzük, hogy apósa-e Feri Marinak, amire a válasz: `no`, végül pedig kiíratjuk a programmal az összes nagyszülő–unoka párt (hat megoldást kapunk).

```
goal
  testvére("Péter","Mari").
%  yes

%  nagymamája(X,"Péter").
%  X=Anna
%  X=Bori
%  2 Solutions

%  nagypapája(X,"János").
%  No Solution
```

```
%  anyósa(X,Y).
%  X=Bori, Y=Juli
%  X=Anna, Y=János
%  2 Solutions

%  apósa("Feri","Mari").
%  no

%  unokája(X,Y).
%  X=Péter, Y=Anna
%  X=Mari, Y=Anna
%  X=Péter, Y=Bori
%  X=Mari, Y=Bori
%  X=Péter, Y=Feri
%  X=Mari, Y=Feri
%  6 Solutions
```

Ahogy a dolgozat fő részében már említettem, a Prolog mindehhez két műveletet használ: *unifikációt* és *visszalépést*. Megpróbál illeszteni a változók helyére értékeket (sorban halad a lehetséges értékek között), és ha elakad, akkor visszalép egyet, az utolsó „elágazásnál" más irányba indul el, más értékkel próbálkozik. Ha ez sem vezet eredményre, újra visszalép és így tovább egészen addig, amíg vagy nem tud mindent unifikálni, vagy el nem fogynak a lehetőségek.

Az első célállításra például hamar megtalálja a választ: megkapja bemenetnek, hogy testvére(Péter,Mari), és először megkeresi az ehhez a predikátumhoz tartozó sort: testvére(A,B), ahol A helyére *Pétert*, B helyére *Marit* illeszti. Majd megpróbálja teljesíteni a feltételeket: kell keresnie egy olyan C-t, amelyre a szülője(C,Péter) teljesül, ezt rögtön első próbálkozásra megtalálja (az első sor ezzel a predikátummal az, hogy szülője(Juli,Péter)), majd megnézi, hogy C=Juli helyettesítéssel teljesül-e a *testvére* következő feltétele, vagyis hogy szülője(Juli,Mari). Megint megpróbál illeszteni, az első *szülője* predikátumot tartalmazó sorra nem tud, de a másodikra már igen. Végül leellenőrzi, hogy fennáll-e, hogy A nem azonos B-vel, és mivel ez is teljesül, kiírja, hogy `yes`.

Még egy fontos tulajdonsága van a Prolognak, ami a nyelvnek is sajátja: a *rekurzivitás*. A listakezelést is így lehet megoldani: például annak ellenőrzése, hogy egy elem benne van-e egy listában, úgy történik, hogy megnézzük, hogy megegyezik-e a lista fejével, vagy benne van-e a lista maradékában. Azt, hogy benne van-e a lista maradékában, úgy ellenőrizzük, hogy a fejét levágjuk, és megnézzük, hogy az elem benne van-e az így nyert listában, vagyis hogy megegyezik-e az új lista fejével, vagy benne van-e a maradékában. A keresés akkor marad abba, amikor a lista feje már nem vágható le, vagyis elértünk az üres listáig (a lista fejét már annyiszor levágtuk, ahány eleme volt az eredeti listának):

```
in(X, [X|_]).
in(X, [_|T]):-
   in(X, T).
```

Számunkra azért is fontos ennyire a listakezelés, mert a programban gyakorlatilag mindent listaként kezeltünk: a mondatot alkotó szavakat, a szavakat alkotó morfémákat, a szintaktikai viszonyrendszert, vagy akár a proto-DRS-eket.

2. sz. melléklet: Egy hosszabb szöveg elemzése a ReALIS-ben

A második szöveges melléklet egy befektetési szöveg elemzését tartalmazza, amelyet a ReALIS dinamikus szemantikai keretben készítettem el (az elemzés részletes magyarázata Kleiber (2005)-ben olvasható). A példa annak érzékeltetésére szolgál, hogy az elmélet nem csupán mondatokhoz, hanem szövegekhez is képes nagyon alaposan kidolgozott szemantikai reprezentációt társítani, amely kellően részletes ahhoz, hogy akár a fordítás interlinguája is lehessen. Nem csupán a predikátumokat és argumentumaikat tartalmazza, hanem a különböző retorikai viszonyokat is; számot ad a többes számról és a szöveg feldolgozása közben létrejövő lehetséges világok struktúrájáról: számáról, egymásba ágyazódásáról, „címkéiről" (például feltételes állítások (restrictor és nucleus világ), vágy, gondolat, lehetőség), vagy a referensekre való visszautalási lehetőségekről stb.[58] Az elemzés továbbá érv a reprezentacionalizmus mellett is, vagyis hogy bizonyos mondatok megértéséhez elengedhetetlen, hogy azokat valamiféle struktúrában reprezentáljuk, mert egy szöveg által felépített lehetségesvilág-struktúra lehet annyira bonyolult, hogy a beszélőnek, illetve a hallgatónak muszáj valahogy megjegyeznie a hierarchiát ahhoz, hogy „ugrálhasson" a belső világok között (erre azért van szükség, mert a referensek tetszőleges mélységben be lehetnek ágyazva, és bármikor visszatérhetünk bármelyik fiktív világba; erre több példát is szolgáltat a szöveg)[59].

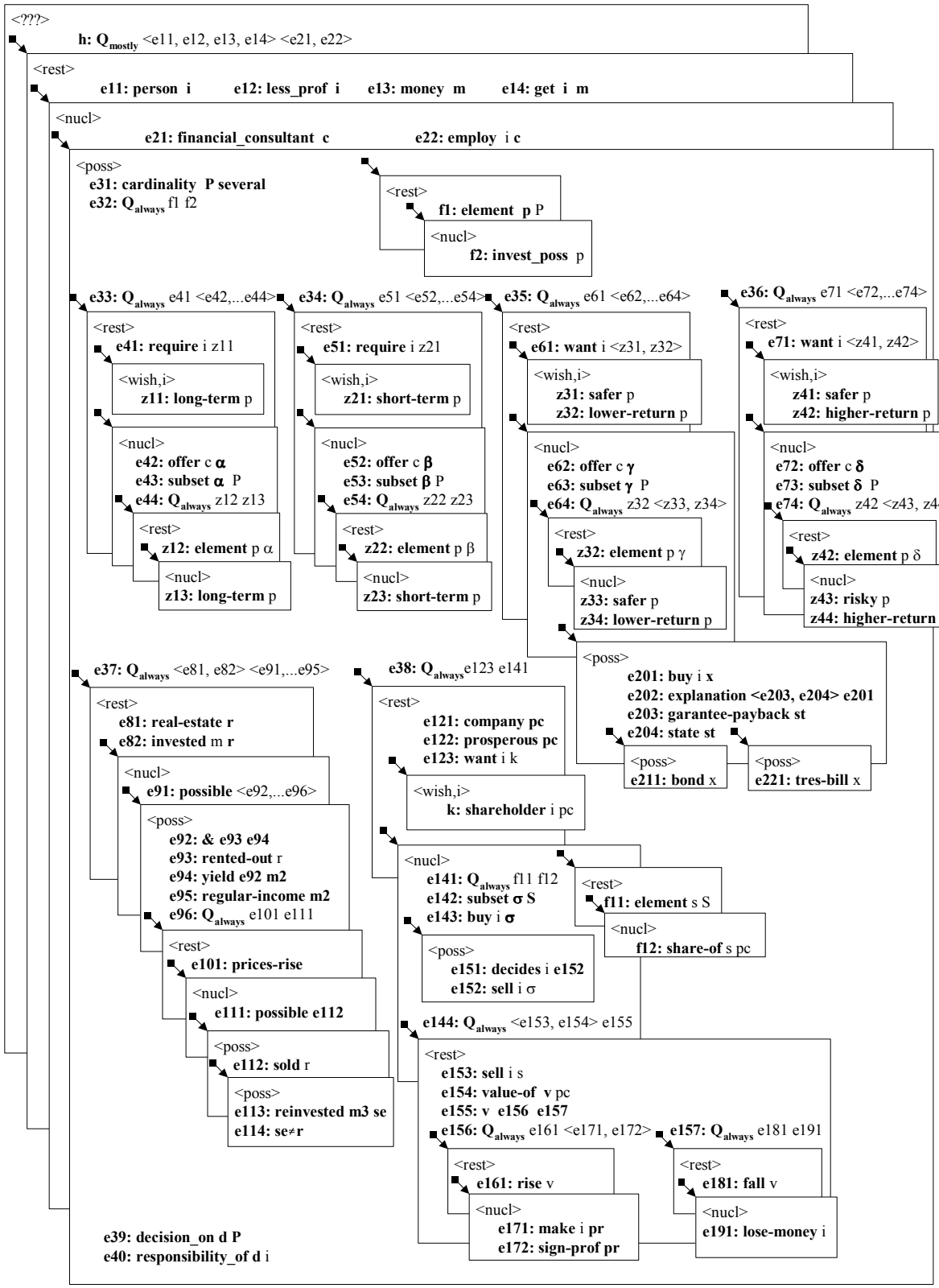
A szöveg:

> Egy pénzügyekben kevésbé jártas ember, ha jelentősebb pénzösszeghez jut, pénzügyi tanácsadóhoz fordul. A szakember többféle megtakarítási lehetőséget ajánlhat attól függően, hogy hosszú vagy rövid távú megtérülést kíván elérni a befektető, illetve, hogy mérsékelt hozamú biztonságos, vagy magasabb hozamú, de kockázatos befektetést szeretne. Ha ingatlant vásárol, bérbe adhatja, így folyamatos bevételre tesz szert, vagy az ingatlanárak emelkedése esetén értékesítheti, és az összeget újra befektetheti egy kedvezőbb üzletbe. Ha résztulajdonos szeretne lenni egy jól működő részvénytársaságban, akkor részvényeket vásárol, melyek alapján évente részesedik a cég nyereségéből. Ha mégis úgy dönt, hogy eladja őket, a vállalat értékének dinamikus növekedése révén jelentős haszonra tehet szert, de kedvezőtlen gazdasági folyamatok következtében befektetése veszíthet is az értékéből. Ha teljes biztonságra törekszik, és alacsonyabb hozammal is megelégszik, államkötvényt vagy kincstárjegyet vásárol, melyek visszafizetését az állam szavatolja. Természetesen a lehetőségek közüli választás a befektető döntése.

---

[58] Itt csak azokat soroltam, amelyekre ez a részleges elemzés is kitér. Az ábra nem tartalmazza az időre utaló jegyeket, a topik, illetve fókusz azonosításáért felelős kurzort stb.; mivel a hangsúlyt a különböző fiktív világok bonyolult hierarchiájára helyeztem.

[59] Egy ezzel analóg jelenség a véges automata és a veremautomata esete: a véges automata rögzített számú beágyazást képes kezelni, ha azonban korlátlan mennyiségű adatra kell emlékezni, akkor már veremautomatára van szükség, amelynek (a véges automatával szemben) már van „memóriája".



19. ábra: A szöveg (részleges) reprezentációja. Látható, hogy hatféle lehetőséget ajánlanak a hallgatónak: (1) hosszú távú befektetéseket, (2) rövid távú befektetéseket, (3) biztonságosabb és alacsonyabb megtérülésű befektetéseket, (4) kockázatosabb de magasabb megtérülésű befektetéseket, (5) ingatlanba fektetés lehetőségét, végül (6) részvények vásárlását.