A MetaMorpho magyar-angol gépi fordító rendszer igei vonzatkereteit működtető nyelvtan

Merényi Csaba

MorphoLogic kft., 1126 Budapest, Orbánhegyi út 5
merenyi@morphologic.hu

Kivonat: A Morphologic kft., a Szegedi Tudományegyetem és az MTA Nyelvtudományi Intézete által közösen fejlesztett magyar-angol gépi fordító rendszer nyelvtanának az igei vonzatkeretek kezelését végző részét mutatjuk be. A magyar nyelv a környezetfüggetlen nyelvtanok számára általában nehezen kezelhetőnek tartott jelenségek többségét mutatja, ilyenek a szabad szórend, a megszakított összetevők, vagy az üres kategóriák. A vonzatkeretek azonosításához ezekkel a problémákkal mindenképpen meg kell birkózni. A MorphoLogic MetaMorpho rendszerének különleges eszközeivel, és az alkalmazott kiterjesztett argumentum modell segítségével mindezen problémákra megoldást kínálunk.

1 Bevezetés

Jelen cikkben a MorphoLogic kft., a Szegedi Tudományegyetem és az MTA Nyelvtudományi Intézete által közösen fejlesztett magyar-angol gépi fordító rendszer jelenlegi állásáról számolunk be. Az eddigi felesztés során elsődleges feladatunknak azt tekintettük, hogy a Nyeltudományi Intézetben készülő igeivonzatkeret-leírások működőképessé tételéhez megteremtsük a technikai hátteret, illetve kidolgozzuk a kezelhetőnek tekintett szabálytípusokat. Ennek során elkészült a MorphoLogic MetaMorpho rendszerében egy mondatnyelvtan és egy szabálykonverter, melyek már alkalmasak arra, hogy a lefordított vonzatkereteket egyszerű főnévi csoportokat tartalmazó mondatokban működtessék.

Az alábbiakban röviden összefoglaljuk a MetaMorpho rendszer alapjait, majd rátérünk az igeivonzatkeret-leírások bemutatására, és betekintést nyújtunk abba, hogy a rendszer magnyelvtana hogyan képes ezeket az egyszerű, tömör, magasszintű leírásokat felhasználva az adott vonzatkeret tényleges megjelenései közül gyakorlatilag bármelyiket felismerni és lefordítani.

2 A MetaMorpho rendszer általános bemutatása

Ebben a fejezetben a MetaMorpho rendszer alapjait mutatjuk be a teljesség igénye nélkül, csupán tájékoztató jelleggel. A leírás célja, hogy a cikk következő fejezeteiben a rendszer tulajdonságaira tett utalások érthetőek legyenek az olvasó számára. A MetaMorpho rendszerről általában szólnak, illetve egyes részleteket illetően pontosabb leírásokat tartalmaznak [1], [2] és [3].

2.1 mmo: az alacsony szintű alapformalizmus

A MorphoLogic MetaMorpho rendszerének alapja az *mmo* formalizmus. Egy *mmo* szabály egy ún. *elemzősorból* és egy vagy több ún. *generálósorból* áll. Egy MetaMorpho nyelvtan szabályainak elemzősorai együttesen egy a forrásnyelvet leíró környezetfüggetlen frázisstruktúra-nyelvtant alkotnak, míg a generálósorok az elemzősorban leírt nyelvi egységnek megfelelő célnyelvi alakot vagy alakokat adják meg. Álljon itt egy leegyszerűsített példa, mely csak a bemutatást szolgálja:

```
*NP:123456789-01
HU.NP[Head<-NM] = DET(lex="a") + NM(type!=PROPER)
EN.NP = DET[dettype=DEF] + NM[NP.num]
```

A fenti szabály legelső sora csupán egy szabályazonosító. A második sor az elemzősor, melyben egy determinánst és NM nevű kategóriát vonunk össze NP-vé. A harmadik sor a generálósor, melyben azt írjuk le, hogy az adott NP-t hogyan kell a fordítás előállítása során kifejteni. A formlizmus alapvetően kétfajta kategóriát kezel: terminális és nemterminális szimbólumokat. Magától értetődő módon az elemzősorban a terminális szimbólumok közvetlenül a szintaktikai modul inputját képező tokeneknek felelnek meg, míg a nemterminálisok a más szabályok által létrehozott egységekre utalhatnak. A generálósorban szabadon csak terminális szimbólumok szerepelhetnek, ezek mellett csak olyan nemterminálisok állhatnak, melyek az elemzősorban előfordultak[24]. A terminálisok generálása már a szintaktikai modul után következő programelemek feladata, míg a nemterminálisok kifejtése az elemzősorbeli párjukat létrehozó szabály generálósorai által történik.

Mint az a példából is kiderül, az egyes szimbólumok különféle tulajdonságokkal, pontosabban jegyekkel rendelkezhetnek, ennyiben több a rendszer egy egyszerű környezetfüggetlen nyelvtannál. Ezek a jegyek tetszőleges nyelvi (lexikális, morfológiai, szintaktikai), vagy pusztán a rendszer belső működése szempontjából fontos technikai jellegű információt hordozhatnak. Jegyeink három típus valamelyikébe tartoznak. A szimbolikus jegyek egy előre definiált véges értékkészletből vehetnek fel egy értéket. Ilyen a fenti példában az NM *type* jegye, melyet most a PROPER értékkel hasonlítottunk össze. A sztring típus értelemszerűen egy karakterlánc tárolására alkalmas, példánkban a DET *lex* jegye ilyen. A harmadik, redkívüli jelentőségű jegytípus a poin-

[24] Nem teljesen igaz az, hogy minden nemterminálisnak van megfelelője az elemző sorban. Lásd később az ún. Brahma-szabályokat, valamint a pointerből való generálás lehetőségét.

ter, amely részelemzések hatékony tárolására alkalmas. Fent az NP *Head* nevű pointer típusú jegyében tároltuk azt a részfát, melyre az NM nevű szimbólum utal.

Bármelyik típusba tartozó tulajdonságok szabadon beállíthatók, örökölhetők, és tetszőleges kikötést lehet rájuk tenni. A pointer jegyben tárolt szimbólum tulajdonságaira is lehet hivatkozni. A rendszer legújabb változatában a szimbolikus értékekből szabadon definiálható operátorok segítségével kiszámított értéket is tovább lehet adni. A jegyérték-ellenőrzésnek is van egy új módja, mely nem egyszerűen egyenlőséget vagy nemegyenlőséget vizsgál, hanem szintén szabadon definiálható módon egyfajta „kompatibilitást", amely különösképpen mondatrészek közötti egyeztetések, illetve szemantikai megötések ellenőrzésekor hasznos.

A pointerek továbbadhatósága és a generálás során a pointerből való létrehozás lehetősége együttesen megoldják azt a problémát, hogy bizonyos összetevőket ne azon a ponton generáljunk, ahol az elemzésbe bekerültek. Ilyen „mozgatások" segítségével továbbra is jól strukturált, könnyen karbantartható nyelvtant írhatunk olyan nyelvi jelenségek kezelésére is, melyeknél a magyar illetve az angol szórend eltérő, sőt valamelyik nyelvben az összetevő egyes részei akár egymástól távol is megjelenhetnek.

2.2 mmc: Brahma szabályok

Korábban említettük, hogy a generálósorban csak olyan nemterminális szimbólumok szerepelhetnek, amelyeknek van az elemzősorban párja. Ez alól a szabály alól két kivétel is van. Az egyik az a fent említett lehetőség, hogy egy nemterminális szimbólumot egy pointerben továbbadott (másik szabályban szereplő) elemzősori szimbólumból hozunk létre. A másik lehetőség első hallásra ennél jóval meglepőbb: a semmiből is előteremthetjük. Ha a rendszer egy olyan nemterminálist talál generáló oldalon, melynek nincs párja az elemzősorban, akkor mielőtt hibát jelezne, egy különleges szabályhalmazban, az ún. *Brahma fájlban* megnézi, hogy a nyelvtanírók nem rendelkeztek-e az adott kategória elemzősori megfelelő nélküli létrehozásának miként jéről. Ha az adott kategória szerepel ebben a fájlban, akkor innen folytatódik a generálás; a csak generálósorokból álló *Brahma szabály* a jegyértékek alapján hozza létre a megfelelő alakot. Ezek a speciális elemzősor nélküli *mmo* szabályok másnéven az *mmc* szabályok.

Az *mmc* szabályok bementetét képező jegyértékek között természetesen szerepelhetnek pointerek is, így a Brahma szabályok nem minden esetben a „semmiből teremtenek", néha csak bonyolult feltételrendszerek többszöri leírását spóroljuk meg velük. Egy pointerben tárolt szimbólum különböző körülmények közötti más-más formában való generálását több *mmo* szabályból is kényelmesen mintegy „szubrutinszerűen meghívhatjuk" így. Akárcsak a programozásnál, a többször előforduló összetett feladatokat célszerű ilyen jól karbantartható egységes módon megoldani.

2.3 mmd: a könnyen kezelhető magasszintű leírás

A MetaMorpho rendszer egyik nagy erényének tartjuk az egységes leírási módot. Legyen szó mondatszerkezeti szabályról, lexikálisan csak részben specifikált igei mintáról, vagy a legegyszerűbb szótári tételről, gyakorlatilag minden nyelvi informá-

ció *mmo* formalizmusban van tárolva. Ugyanakkor az *mmo* olyan alacson szintű leírás, amelynek minden technikai részletet tartalmaznia kell. Különösképpen az adott esetben rendkívül nagy számban előforduló jegyértéköröklések teljesen olvashatatlanná tehetik a rendszer számára használható *mmo* szabályt. A lexikális erőforrások hatékony előállítása érdekében kidolgoztuk azt a technológiát, amely lehetővé teszi, hogy a szabályírók egy egyszerűsített formalizmusban dolgozhassanak, és csak a nem redundáns információt kelljen látniuk, illetve szerkeszteniük. Ezt az egyszerűsített formalizmust nevezzük *mmd*-nek. Az *mmd* szabályokból egy konverterprogram állítja elő a megfelelő *mmo* szabályt (vagy szabályokat) a rendszer igényeinek megfelelően. A konverzió a programba előre bekódolt algoritmusok alapján történik, de nem teljesen rugalmatlanul, ugyanis az *mmd* formalizmus, szemben az *mmo*-val, tartalmaz ún. metajegyeket is, amelyek segítségével a szabályírók magasszintű utasításokkal befolyásolhatják, hogy pontosan milyen szabályok jöjjenek létre. Jó példa erre az igei vonzatkeretek leírásánál használt :*opt* metajegy, amely azt mondja meg, hogy egy adott vonzat opcionális-e. Ha igen, akkor a konverter két *mmo* szabályt is előállít az egy *mmd* szabályból: az egyikben szerepel az adott vonzat, míg a másikból kimarad.

3 Magyar-angol vonzatkeret-leírások

Az igei vonzatkeretek leírásánál természetesen a magasszintű *mmd* nyelvet használjuk. Pillanatnyilag a nyelvtan több mint 17000 VP-s szabályt tartalmaz, és a hiányok pótlása, valamint a lexikális és szemantikai megkötések alapján történő további finomítás után ennek többszöröse is lehet a végleges szám. Ilyen nagy munkánál mindenképpen törekedni kell arra, hogy a leírás a lehető legegyszerűbb és legtömörebb legyen.

Magyar nyelvű mondatok elemzésekor a vonzatkeretek azonosítását számos a nyelvre jellemző jelenség nehezíti meg. Első helyen említendő ezek közül a szabad szórend. Figyelembe véve a topikalizáció és a fókuszba emelés lehetőségeit, a vonzatok gyakorlatilag bármilyen sorrendben előfordulhatnak, illetve az ige és a szabad módosítók ezek között kevés megkötéssel bárhol elhelyezkedhetnek. További gondot okoz az, hogy bizonyos vonzatok nem egy megszakítatlan összetevőként jelennek meg a mondatban. A birtokos szerkezetekből kiemelhető a birtokos, 'hogy'-os alárendelt mondatok extrapozíció után elválhatnak az utalószótól, a vonatkozó mellékmondatok szintén gyakran az igei csoport végére mozgatva jelennek meg. A vonzatkeret azonosítása és a fordítás egyaránt megkövetelik, hogy ezeket a távoli összetevőket egymáshoz rendeljük és egy egységként kezeljük. Végül megoldást kellett találni arra is, hogy egyes névmási vonzatok elhagyhatók, pontosabban a felszínen nem jelennek meg, de a vonzatkeretek azonosításához, illetve fordításukhoz fel kell tételeznünk jelenlétüket.

Az általunk alkalmazott leírás a fenti bonyodalmak ellenére az egyszerűség, tömörség és kezelhetőség követelményeinek jól megfelel. A szabályírónak gyakorlatilag csak annyi a feladata, hogy egy semleges magyar szórendben felsorolja a vonzatokat és az igét, és azok azonosító jegyeit (pl. eset, névutó, lexikális alak) leírja. A vonzatok között esetleg fennálló kapcsolatokra magasszintű meta-jegyekkel tehetnek megkötést. A fordítást egy semleges szórendű angol vonzatkeret formájában kell megadni, melynek elemei a magyar vonzatoknak felelnek meg. A használható szimbólumok a magnyelvtan által egyébként nem használt, intuitív kategóriákba tartoz-

nak: SUBJ, OBJ, COMPL. Példaként tekintsük a „vmi beleharap vmibe” vonzatkeret leírását:

```
*VP=bele|harap:7
HU.VP = SUBJ(animate=YES) + TV(:lex="bele|harap") +
COMPL#1(pos=N, case=ILL)
EN.VP = SUBJ + TV[lex="bite"] + COMPL#1[prep="into"]
```

Ennek az egy mmd szabálynak a segítségével a rendszer képes lefordítani az alábbi mondatok mindegyikét (ezek valódi példák a működő programból):

Moose>a mókus beleharapott a körtébe. (2)
1: [the squirrel bit into the pear.]

Moose>beleharapott a mókus a körtébe? (3)
1: [did the squirrel bite into the pear?]

Moose>a körtébe a mókus harapott bele. (4)
1: [the squirrel bit into the pear.]

Moose>beleharaptam a körtébe. (5)
1: [I bit into the pear.]

Moose>beleharaptak? (6)
1: [did they bite into it?]

Moose>belém harapott a mókus. (7)
1: [the squirrel bit into me.]

Moose>a mókusnak ki harapott bele a körtéjébe? (8)
1: [who bit into the squirrel's pear?]

Amint azt (1)-(3) mutatják, az ige, az igekötő és a vonzatok tényleges megjelenési sorrendjétől függetlenül azonosítani tudjuk a vonzatkeretet. (4) és (5) azt illusztrálják, hogy egy vagy több vonzat is megvalósulhat zéró felszíni alakú névmásként. Ezeket az eseteket is kezelni tudjuk a fenti *mmd* szabállyal, annak ellenére, hogy abban mind az alany, mind az *ILLATIVUS* esetű vonzat kötelező vonzatként szerepelnek. (6) a vonzatkeret „ragozott igekötős” változatát mutatja be, amelynek létezése kikövetkeztethető a „vmi beleharap vmibe” minta létezéséből, és amelyet ezért a szabályíróknak nem kell külön lekódolniuk. Végül (7) egy olyan esetet mutat be, amelyben az *mmd* szabályban egy COMPL#1 nevű szimbólummal jelölt vonzatot valójában két különböző helyen megjelenő nyelvi egység – egy elmozgatott birtokos és a birtok – alkotja.

A következő fejezetben betekintést adunk annak technikai részleteibe, hogy az egyszerű *mmd* leírásból a konverter által előállított jóval bonyolultabb *mmo* szabályok hogyan tudják a magnyelvtan segítségével gazdaságosan és áttekinthető módon lefedni a példákban bemutatott jelenségeket.

4 Magnyelvtan

4.1 A Kiterjesztett Argumentum fogalma

Az ige vonzatai a felszínen sok különböző formában jelenhetnek meg, de sok szempontból azonos módon kell őket kezelni. Ezért célszerű volt mindenféle vonzatot egy közös kategória, az ARG alá rendelni. Azonban egy környezetfüggetlen nyelvtanban egy egyszerű nemterminális kategória önmagában nem elegendő nem összefüggő vagy a felszínen nem megjelenő összetevők ábrázolásához. Emiatt a magnyelvtanban a vonzatokat egy absztrakt kategóriának tekintjük, ezt nevezzük *Kiterjesztett Argumentum*nak vagy röviden *xARG*-nak. A kiterjesztett argumentum tényleges ábrázolása nem egy nemterminális szimbólum, hanem egy jegyhalmaz, amely a teljes vonzatkeretet reprezentáló nemterminális szimbólum (a VPP) jegyei között szerepel.

Az xARG-ot számos jegy alkotja, amelyek a vonzat lexikális, morfológiai és szintaktikai tulajdonságait kódolják, valamint több pointer típusú jegyből is áll. Ezek a pointerek tárolhatják a fejet tartalmazó ARG összetevőt (ha volt ilyen a felszínen), valamint a kimozgatott elemeket, melyek egységesen az ARG-hoz hasonló közös kategória, a LINK alatt jelennek meg. LINK lehet például az NP birtokosa, vagy az NP-t módosító vonatkozó mellékmondat. A pointerek szerepe az elemzés során gyakorlatilag elhanyagolható (néha azokon keresztül hivatkozunk olyan tulajdonságokra, melyek nincsenek kivezetve az xARG-ba), a vonzatot alapvetően az egyéb jegyértékek reprezentálják. Generálásnál viszont, amennyiben nem zéró névmásról volt szó, az ARG és a LINK pointerek segítségével állíthatjuk elő a fordítást.

A kiterjesztett argumentumok véglegesen csak a teljes vonzatkerettel együtt állnak össze. A vonzatkereteket reprezentáló VPP kategória egy rekurzív szabályhalmaz segítségével jön létre, amely az ige előtt és után álló minden mondatrészt egyenként bekebelez, amíg azok el nem fogynak. Az egyes beelemzett mondatrészeket és tulajdonságaikat ezek a rekurzív VPP szabályok a megfelelő xARG jegyekben helyezik el. A VPP öt xARG számára tartalmaz jegyeket, amelyeket a vonzatok a beelemzés sorrendjében foglalnak el. Ezekre a jegyhalmazokra a továbbiakban xARG1...xARG5 néven fogunk utalni.

A konverter által létrehozott *mmo* szabályok tehát valójában nem maguk vonják össze az *mmd*-ben leírt mondatrészeket egy VP-vé, hanem a magnyelvtan fent vázolt mechanizmusa által már összerakott VPP-k fölötti szűrőként működnek olymódon, hogy azok xARG-jaira tesznek megkötéseket. A továbbiakban azt fogjuk felvázolni, hogy pontosabban hogyan működnek ezek a szűrők, illetve milyen szűrőkre van szükség a szabad szórend, a zéró névmások, illetve más korábban említett jelenségek kezeléséhez.

4.2 A szabad szórend kezelése

A fent elmondottak szerint a VPP az argumentumokat az xARG1 ... xARG5 jegyhalmazokban olyan sorrendben tartalmazza, amilyen sorrendben azokat beelemezte. Az igei vonzatkeretet azonosító szűrők két kölünböző módon ismerhetik fel az adott igei mintát tetszőleges szórenddel. A legkézenfekvőbb eljárás az, hogy minden lehet-

séges sorrendhez előállít a konverter egy külön *mmo* szabályt. Pl. a korábban bemutatott „vmi beleharap vmibe” vonzatkerethez lehetne két *mmo*-t generálni, melyek elemző sorai vázlatosan így nézhetnének ki:

```
HU.VP = VPP(lex=”harap”, ik=”bele”, argnum=ARG2, Arg1-
>case=NOM, Arg2->case=ILL, ...)

HU.VP = VPP(lex=”harap”, ik=”bele”, argnum=ARG2, Arg1-
>case=ILL, Arg2->case=NOM, ...)
```

A fejlesztés kezdeti szakaszában ténylegesen ezt az utat választottuk, mert a C++ nyelven írt konverter program egyszerűbben elő tudta állítani a vonzatok összes permutációját. Ennek a megközelítésnek azonban az a hátránya, hogy feleslegesen megsokszorozza az előállítandó *mmo* szabályok számát. Mivel egyéb okoknál fogva is szükséges lehet a szabályok többszörözése, a létrehozott nyelvtan mérete akár a rendszer teljesítményét veszélyeztető módon is felduzzadhatott.

Szerencsére az operátorokkal kiterjesztett *mmo* formalizmus elég erős eszköznek bizonyult ahhoz, hogy a magnyelvtanban hatékonyan megoldjuk az összegyűjtött xARG-ok permutálását. Így akármilyen sorrendben is szerepeljenek eredetileg a vonzatok, az összes lehetséges permutációt tartalmazó VPP-k előállítása után már egyetlen szűrő is elgendő a vonzatkeret azonosításához. A felesleges VPP-k nem terhelik a rendszert, mert bár ezen az elemzési ponton kissé felszaporodik a létrejött elemzési tények száma, azok közül csak egynek lesznek felmenői, amelyek tovább kombinálódhatnak más részelemzésekkel. Így kombinatorikus robbanástól nem kell tartanunk.

4.3 A zéró névmások kezelése

A zéró névmások kezelése pillanatnyilag a konverter segítségével történik. Az *mmd* leírás szerint alanynak vagy tárgynak minősülő vonzatokról tudja a konverter, hogy azok a felszínen esetleg nem jelennek meg, ezért olyan szűrőket is generál, amelyek ezeket a vonzatokat nem kérik számon a VPP-n:

```
HU.VP = VPP(lex=”harap”, ik=”bele”, argnum=ARG2, Arg1-
>case=NOM, Arg2->case=ILL, ...)

HU.VP[xs=YES, ...] = VPP(lex=”harap”, ik=”bele”,
argnum=ARG1, Arg1->case=ILL, ...)
```

A VP-nek a vonzatokra vonatkozó jegyeit viszont úgy tölti ki a második szabály, hogy abban jelzi a zéró névmás jelenlétét, gyakorlatilag maga a szűrő létrehozza a megfelelő xARG-ot, és jegyeinek értékét többek között az ige ragozásából következteti ki. A generálósorban az ilyen vonzatot nem tudjuk a megfelelő xARG pointer típusú jegyeiből létrehozni, hiszen azok nem tartalmaznak semmit, ehelyett a 2.2 alatt említett Brahma szabályok segítségével generáljuk őket. A Brahma generáláshoz szükséges jegyértékeket a konverter tölti ki a konkrét szabály tulajdonságaitól függően. Például az élőre vagy élettelenre utaló névmás generálása közötti választást befo-

lyásolja az, hogy az mmd-ben volt-e, illetve milyen megkötés volt a megfelelő vonzat *human* jegyére.

4.4 A megszakított összetevők kezelése

A megszakított összetevők kezelésében a vonzatkeretet azonosító szűrőknek kevés szerep jut. Ezek a vonzat xARG-gal való modellezésének köszönhetően tulajdonképpen maguktól működnek. A VPP összeállítása során alkalmazott bonyolult, sok nyelvi tudást hordozó szabályok a szűrőknek átadott vonzatkeret-jelöltek xARG-jaiban már minden nyelvtanilag lehetséges módon egymáshoz rendelték az ARG-okat és a különféle LINK-eket. Így pédául az előző fejezetben leírt (7)-ben az *[a mókusnak]* távoli birtokos LINK-et és az *[a körtéjébe]* birtokos személyragot tartalmazó ARG-ot a személyrag meglétéből kiindulva, majd a szám és személy szerinti sikeres egyeztetés után, egyazon xARG-ban tárolva küldjük a szűrőknek, melyek csak közvetve szólhatnak bele a megszakított összetevők felismerésébe. Amennyiben több lehetséges egymáshoz rendelés is létezik, a szűrők az egyes vonzatokra, vagy azoknak például éppen a birtokosára tett megkötéseikkel egyértelműsíthetik az elemzést.

Az xARG generálásakor minden LINK-jét átadjuk neki, melyeket azok típusától függően saját szerkezetén belül a megfelelő helyre fog elhelyezni. Az adott példánál maradva, minden célnyelvi NP kategóriájú argumentum számít arra a lehetőségre, hogy kap felülről egy birtokos LINK-et, amelynek felhasználásával birtokos szerkezetet generál. Így lesz a magyar mondatban egymástól elválasztva megjelenő *[a mókusnak]* és *[a körtéje]* kifejezésekből az angolban egy NP: *[$_{ARG}$[$_{LINK}$the squirrel]'s pear].*

5 Összegzés

Az előző fejezetekben röviden vázoltuk, hogy a MorphoLogic MetaMorpho rendszere elsősorban a pointer típusú jegyek, a Brahma szabályok és a Kiterjesztett Argumentum modell segítségével hogyan tudja kezelni a környezetfüggetlen nyelvtanok számára általában nehézséget jelentő szabad szórendet, zéró elemeket, valamint a megszakított összetevőket. A fejlesztés alatt álló magyar-angol gépifordítórendszerünkben ezeket a technológiákat a gyakorlatban is sikerrel alkalmazzuk.

Bibliográfia

1. Prószéky, Gábor & Tihanyi, László: MetaMorpho: A Pattern-based Machine Translation Project. In: Proceedings of the 24th 'Translating and the Computer' Conference. London, United Kingdom, 19–24 (2002)
2. Tihanyi László: A MetaMorpho projekt története. Magyar Számítógépes Nyelvészeti Konferencia 2003, Szeged.
3. Gábor Prószéky, László Tihanyi, Gábor Ugray: Moose: A Robust High-Performance Parser and Generator. EAMT Workshop, Malta, 2004