GeLexi project: Sentence Parsing Based on a GEnerative LEXIcon*

Gábor Alberti*, Judit Kleiber*, and Anita Viszket†

Abstract

The principal aim of our research team[1], called GeLexi, is to legitimate a new sort of generative grammar via verifying its computational implementability. This grammar is more radically "lexicalist" than any earlier one: no phrase structure trees are generated, but word order is accounted for by means of ranked parameters. Another novelty is the extension of "total lexicalism" to morphology: lexical items are assigned not to words but to morphemes. Our parser, in accordance with the basic task of every generative grammar, decides whether a sentence is grammatical, and if it is, then provides a morphophonological analysis, a compilation of grammatical relations, and two kinds of semantic representations. At the end we show some examples to demonstrate our procedures, among them a sentence containing the conjunction *és* 'and', which is our latest development.

1 Introduction

Our general aim is to verify that computational linguistics is worth returning from the nowadays wide-spread attitude characterized by "shallow parsing" (which is held to save expenses) to the pure theoretical (generative) linguistic basis. A formal (generative) grammar can be elaborated [4] showing the distribution of capacity advantageous in modern computer science: "minimal processing - maximum database".[2]

The starting aim of our team (section 2) is the same, approaching from a theoretical point of view: we would like to legitimate a new sort of generative grammar

*Presented at the 1st Conference on Hungarian Computational Linguistics, December 10–11, 2003, Szeged.

†University of Pécs, Faculty of Humanities, Linguistics Department, H7624 Pécs, Ifjúság útja 6., Hungary, e-mail: `gelexi@btk.pte.hu`

[1]We are grateful to the Hungarian National Scientific Research Found (OTKA T038386). Further thanks to Kata Balogh, who used to be an active member of the team.

[2]In harmony with the chance available now: to use a significantly greater number of huge patterns than earlier due to the immense increase in memory capacity. In the meantime generative linguistics, which used to be chiefly "process-oriented" (i.e. syntax-centered) in its first period [16], took a sweeping lexicalist turn [14, 15, 17, 19, 20]. The current attitude can be characterized by two mottoes of Joshi's [19], the father of mildly context-sensitive grammars [22]: "Complicate Locally, Simplify Globally", and "Grammar $\approx$ Lexicon".

(GASG: *Generative Argument Structure Grammar*) by working out its computational implementation (because a successful implementation is the best evidence for the exactness and consistency of a formal system). This grammar is more radically lexicalist than any earlier one, since no phrase structure trees are built, yet word order can be accounted for by means of ranked parameters for requirements concerning *immediate precedence* relations between words.

In section 3 our four-year work is looked over according to the assumption that the principal aim of the first national conference on computational linguistics is the introduction of the teams working on this field.

Section 4 is denoted to the demonstration of some examples in order to elucidate how our parser works, concentrating on the conjunction *és* 'and' in the spirit of the strategy that at all conferences we present a new achievement beyond summarizing our general ideas.

2 Starting aim

Due to generative linguistics [16], there is a formal frame to express the old recognition that the meaning of a sentence comes from two sources: lexical items and the structure they form. Initially the main question was studying the combinatorial possibilities of these structures by mathematical means (see the Chomskyan hierarchy of grammars, especially context free and context sensitive grammars [16] [22]). From these studies two fields started developing: generative language description and computational linguistics. In the generative paradigm the indispensability of transformation rules (moving constituents) could not be theoretically proved [22], consequently from the 70s several new (generative) approaches could be established [14] [15] [18] [19] which dispense with transformation (Partee et al [22] show all the variants of these *"mildly" context free* grammars). In connection with this, in these new approaches the *lexicon* plays the crucial role (instead of syntax, as earlier) in describing the linguistic phenomena, and there are only highly general phrase structure rules in *syntax*. The fact that in the 90s even the previously strictly transformational Chomskyan "main line" took a similar turn (the Minimalist Program [17]) shows that the lexicalist tendency is extremely strong nowadays.

GASG is a totally lexicalist grammar, which accomplishes Karttunen's "radical lexicalism" [20]: it is a grammar which dispenses with not only transformational rules but phrase structure trees as well (it is similar to dependency grammars [24] in this respect); yet it fulfills the basic generative task, namely that the set of grammatical sentences can be defined, and (through the analysis) a syntactic and semantic representation can be assigned to the sentence in question. Grammatical knowledge is built only in the descriptions of lexical items: each morpheme declares what kind of "environmental requirements" a sentence containing the given lexical item has to fulfill. Obviously, this is not the only information that these lexical units have: their own features are also registered in these descriptions, since other morphemes need this information when they "intend" to form sentences with them.

Generative linguistic theories are important to be verified by computational implementations, for having working algorithms is the best evidence for the exactness and consistency of a formal system. Our implementation does fulfill the basic generative task: it decides whether the input sentence is grammatical or not, and if it is, then it provides a syntactic and semantic representation.

We will turn back to other properties of our grammar and parser later on, but now let us point out a further advantage of this (lexicalist) approach. These days a significantly greater number of huge patterns can be used than earlier due to the immense increase in memory capacity [23]. It can be favorable to combine this technical development with the lexicalist approach, especially with a homogeneous grammar. Further on, GASG can legitimate the heuristic procedures (based on pattern matching), which were previously used intuitively, by improving them to theoretically correct systems.

3 Previous work

The idea of developing a computational implementation based on GASG was first published in 1998 at an international conference in Debrecen [1]. The first version of our parser was done in 2001 which could decide whether a Hungarian sentence was grammatical or not [10, 3]. It could account for regent-argument relations, agreement, and word order, but we had neither semantic nor morphological component: lexical items were assigned to words, not morphemes like now.

Then we started elaborating the semantic representation of GASG. The theory serving as the starting point was a developed version [2] of Kamp's DRT (Discourse Representation Theory) [18]. The main argument for choosing this approach was that GASG could be the *compositional* [22] grammar for this discourse semantic theory, which is much more advanced than the Montagovian semantic systems [22]. Our concepts until then were demonstrated in details in the Proceedings of a conference about principles, models, and rules published in Szeged [4]. Morphology is not discussed there, because until then lexical items were (morphologically complex) words, and were claimed to be arranged in a multiple lexical inheritance network put together by means of well-tried regular devices [21].

Later it became evident that for achieving the required complete homogeneity we should regard morphology in a totally lexicalist way as well. Therefore lexical items are not associated with words but with morphemes (stems and affixes) in this new approach. It is also important that these lexical items have all the information needed on each linguistic level to form sentences, while it is also decided which morphemes are put together, and which morphemes form independent words [5, 11, 12], which is not (necessarily) the same in different languages, e.g. in Hungarian *olvas-hat* 'read-CAN' (one word), but in English *can read* (two independent words). Another fundamental point of GASG and our parser is what technique the phrase structure (which is responsible for word order) is replaced with, namely checking the ranked *immediate precedence* requirements the lexical items have in their descriptions. These requirements can be fulfilled directly (by being next to the given morpheme) or indirectly, when requirements with higher ranks are met.

This is how between words belonging together (e.g. *the girl*) other words can be inserted (*the proud Hungarian girl*), in Hungarian even with their further dependants (*a* két fiára *büszke magyar lány* – 'the *two of her sons* proud Hungarian girl'. In English in this case the adjective has to be after the noun: *the Hungarian girl proud of her two boys*). Morphotax can also be entrusted to rank parameters with the important difference that morphemes cannot bring further dependants (which correlates with the regularity of morphology [21]).

The idea of a totally lexicalist morphology can make the differences between language types irrelevant on the abstract level of *copredicative network* [8], since it does not matter whether the (English) words *I may wait for you* or the (Hungarian) morphemes *vár-hat-l-ak* 'wait-CAN-2SGobj-1SGsubj' are looking for each other. We have started developing a machine translating system through this level [7]. The first results were demonstrated at the 2004 (Maltese) workshop of the European Association for Machine Translation (EAMT'04) [9].

The morpheme-based semantic representation was introduced at a conference in Mexico [6], where we also took the opportunity to publish the mathematical definition of GASG grammar type.

4 The present parser

In what follows, we introduce the components of our parser from morphophonology until semantics presenting some examples as well, first to show how the components work, and then to demonstrate how the conjunction *és* 'and' has been worked out.

4.1 Morphophonology

The input of this level (and the whole parser as well) is a simple string, a series of words. The first task the program has to fulfill is *segmentation*. The relevant lexical items are to be identified on the basis of the database, which has been found in the same program so far, but we are planning to use a relational database instead, for storing lexical items, which we have already started developing[3].

Lexical items consist of two parts, the *own word* and a label. The former shows how the lexical items appear in different environments. Sometimes it is only one form which is possible (e.g. *Mari*, meaning *Mary*), but sometimes more than one variants exist (e.g. *bokor* or *bokr-*, meaning *bush*; *-t/-at/-et/-ot/-t* for accusative case). In the latter case variables are used (*bokOr*, *-Vt*). The different features of the lexical items are stored in their label (first the phonological, then morphological and syntactic properties), but first the English "name" of the predicate is stored[4].

[3]The members of the team working on this project (called *LiLe project*, where 'LiLe' stands for Linguistic Lexicon) are Anita Viszket, Éva Szilágyi, Zoltán Bódis and Judit Kleiber (University of Pécs, Linguistics Department).

[4]It is often asked why we do not use existing encoding systems or even existing morphological parsers. The answer is that we need much more (and sometimes quite different) information than these systems store, because we aim at producing more detailed analyses.

After identifying the relevant lexical items the program checks the well-formedness of the words (it can account for such linguistic phenomena as vowel harmony, shortening, lenghtening, epenthesis, lowering, etc.), and the morpheme order. The output of this component is a list containing these lexical items. If there are more than one possible segmentation of a word, the program can print out each of them. Let us demonstrate this through an example.

```
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)))
```

Asking the well-formedness of the word *dobom*, the program provides two solutions, a verb (*throw*) in 1SG definite conjunction, and a noun (*drum*) with the possessive 1SG. So we can account for ambiguities on this linguistic level.

Finally we quote the morphological output of the parser for a more complex sentence. At the beginning of each line the proper morph can be seen, and the numbers after that (and before the own word) mean that the given morpheme is (1) in which word and (2) which morpheme within that word.

```
gramm("Péter keresteti Marit a magyar rendörséggel.").
Peter look-for - cause - 3sg Mary-Acc the Hungarian police-Instr
'Peter makes the police look for Mary.'

LEXICAL ITEMS:
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,[])))
magyar: n(5,1,li(m("","magyar",""),labstem("Hungarian",phonfst(2,2,1,1),4,[])))
rendörség: n(6,1,li(m("","rendörség",""),labstem("police",phonfst(1,2,3,2),1,[])))
gel: n(6,2,li(m("S","A","l"),labsuff("INSTR",phonfsu(1,2,2,3),1,4)))
```

4.2 Syntax

The input of this level is the list of numbered lexical items (see above). The task this component has to fulfill is to satisfy all the requirements these lexical units have in their descriptions concerning syntax.

The most important requirements are regent (predicate) - argument relations. The verb *vár* 'wait', for instance, has two argument structures in the program (Nom+Acc; Nom+Sublat), so the parser has to find two nouns with these case marking morphemes (the first two or the second two) in order to fulfill one of its requirements. This searching has to be mutual, so these nouns have to find the same verb when they are looking for their regent. This is to exclude sentences like **Péter* *vár Marira Julira* *'Peter waits for Mary Julie.', where two nouns in sublative case are looking for the same verb, so one of their searching cannot be mutual. There is another type of relation (free relations) where the seeking is one-way, e.g. the relation between an adjective and a noun. The adjective needs a noun, but not the other way round; that is why a noun can have more adjectives.

After finding these (mutual or one-way) relations the program checks word order by means of ranked *immediate precedence* requirements, which we have already mentioned. With this technique we can also explain, for example, why free adverbs and arguments can be freely mixed in Hungarian but not in English: suppose regent-argument relation has an *immprec* rank α, free adverb-verb relation has a rank β, and in Hungarian $\alpha = \beta$, but in English $\alpha < \beta$.

If the parser can fulfill all the requirements, the syntactic relations are printed out in two different formats, the first one is more detailed than the much simplified second one. Let us demonstrate this through the previous example.

```
1. gramm("Péter keresteti Marit a magyar rendörséggel.").

2. SYNTAX:
3. gr("noun","regent","subj",1,1,2,1)
4. gr("det","regent","_",1,1,2,1)
5. gr("regent","noun","subj",2,1,1,1)
6. gr("regent","det","subj",2,1,1,1)
7. gr("regent","noun","obj",2,1,3,2)
8. gr("regent","det","obj",2,1,3,1)
9. gr("suff","stem","free",2,2,2,1)
10. gr("regent","noun","obl",2,2,6,2)
11. gr("regent","det","obl",2,2,4,1)
12. gr("suff","stem","free",2,3,2,1)
13. gr("det","regent","_",3,1,2,1)
14. gr("suff","stem","free",3,2,3,1)
15. gr("noun","regent","obj",3,2,2,1)
16. gr("det","noun","free",4,1,6,1)
17. gr("det","regent","_",4,1,2,1)
18. gr("adj","noun","free",5,1,6,1)
19. gr("suff","stem","free",6,2,6,1)
20. gr("noun","regent","obl",6,2,2,1)
21. regent-noun-subj: keresteti-Péter
22. regent-det-subj: keresteti-Péter
23. regent-noun-obj: keresteti-Marit
24. regent-det-obj: keresteti-Marit
25. regent-noun-obl: keresteti-rendörséggel
26. regent-det-obl: keresteti-a
27. det-noun: a-rendörséggel
28. adj-noun: magyar-rendörséggel
```

In the first part three types of free relations and several types of regent-argument relations can be distinguished. First the names of the relations can be seen, and then the numberings of the morphemes which establish the given relation. Free relation types are *suff-stem-free*, when a suffix is in relation with its stem (lines 9, 12, 14, 19), *det-noun-free*, when a determiner found its noun (line 16), and *adj-noun-free*, when an adjective is in relation with its noun (line 18). The regent-argument relation types are *regent-noun-subj/obj/obl*, when a predicate found one of its nominal arguments (lines 5, 7, 10), *regent-subj/obj/obl*, when the regent is in relation with a determinative element, because we claim that the arguments have two pillars, a noun and a determiner[5] (lines 6, 8, 11), or – because regent-argument relations are mutual – *noun-regent-subj/obj/obl*, when the arguments found their regents (lines 3, 15, 20), and *det-regent-_*, when the determiner pillar of the argument is in relation with the regent (lines 4, 13, 17), where *'_'* stands for either *subj*, *obj* or *obl* (it cannot be decided, because – in Hungarian – there are no markings on the determiner pillar concerning syntactic roles). In the second part the same information can be found, but mutual relations appear only once, and the numberings are replaced by the words themselves, so it can be read much easier than the first representation.

It is an interesting point of our approach that not necessarily words (stems) look for each other, but affixes as well. E.g. in the sentence *Énekeltetem Marit* (*sing-CAUSE-1SG Mary-ACC*) 'I make Mary sing' the verb itself do not even require a (human) object, consequently it cannot be the verb stem that searches for the object. The morpheme which needs (thus legitimates) the object in this sentence is the causative morpheme (derivative suffix) *-tAt*, and the morpheme it looks for is not the whole *Marit* or the stem *Mari*, but the accusative case marking suffix *-Vt*, because it is this element that shows that *Marit* is an object in this sentence.

4.3 Semantics

The input of this level is all the information we have at this point (list of lexical items, syntactic relations). The task this component has to fulfill is to provide a DRS (Discourse Representation Structure) putting it together from the proto-DRSs the lexical items give.

The information necessary for producing these proto-DRSs can be found in the descriptions of lexical items. Determiners provide referents (*provref*), and predicates make statements (*pred*). In the sentence *The boy loves a girl*, say, there are two referents *r1* and *r4* (the first and the fourth words provide them), and three statements, that *boy(r1)*, *girl(r4)*, and *love(r1,r4)*. Further on, it is also known that *r1* is an old referent (it has been provided earlier), but *r4* is new. This is the point where different kinds of filters can be built in, e.g. that the first argument of the predicate *love* must be human.

If the program can produce the proto-DRSs and put them together to a final DRS, it is printed out. Let us turn back to the previous example again, and see the discourse semantic output of the parser when the grammaticality of the sentence *Péter keresteti Marit a magyar rendőrséggel* (*Peter look-for-CAUSE-3SG Mary-ACC the Hungarian police-INSTR*) 'Peter makes the Hungarian police look for Mary' is asked.

[5]Sometimes this nominal pillar and the determiner pillar are within one morpheme, e.g. in the case of proper names; and there are also exceptions, e.g. *Péter moziba megy* (*Peter cinema-ILLAT go-3SG*) 'Peter goes to the cinema', which requires a different approach.

```
1. gramm("Péter keresteti Marit a magyar rendörséggel.").

2. DISCOURSE SEMANTICS:
3. provref("fixpoint",[e(2,2,1)])
4. provref("old",[r(1,1,1)])
5. pred("Peter",1,[r(1,1,1)])
6. provref("new",[e(2,1,1)])
7. pred("look-for",2,[e(2,1,1),r(4,1,1),r(3,1,1)])
8. provref("new",[e(2,2,1)])
9. provref("=",[e(2,2,1),e(2,1,1)])
10. pred("cause",2,[e(2,2,1),r(1,1,1),e(2,1,1)])
11. provref("old",[r(3,1,1)])
12. pred("Mary",3,[r(3,1,1)])
13. provref("old",[r(4,1,1)])
14. provref("<or=",[r(4,1,1),e(2,2,1)])
15. pred("Hungarian",5,[r(4,1,1)])
16. pred("police",6,[r(4,1,1)])
```

Referents are provided for Peter (line 4), Mary (11), and the police (13), each of them are old referents, and it is predicated that *r111* is Peter (5), *r311* is Mary (12), and *r411* is the police (16), which is Hungarian (15). In this approach other kinds of referents are provided, such as Davidsonian (eventuality) referents for the events (lines 6, 8), where *e211* is that *r411* is looking for *r311* (line 7), and that *r111* causes the event *e211* (line 10)[6].

In comparison with earlier approaches there are two important changes in this representation. First, in this version, *morphemes* provide the proto-DRSs, so that we can give a more precise interpretation, and, second, we try to give a formulation, which easily fits in the structure of a Lifelong DRS [2], which goes beyond sentence parsing. This method could enable our program to parse whole texts, not only sentences. LDRT provides a partial ordering between possible worlds, this ordering can be seen in lines 9 and 14.

Hungarian is a pro-drop language, which means that personal pronouns do not have to appear in the sentence. Several approaches assume empty elements in this case, but we do not. In GASG (and our parser) in a sentence like *Szeretlek.* 'I love you.' syntax is very simple, it consists of only *suff-stem-free* relations. What is more interesting is semantics, where we have to account for the first and the second argument of the predicate. In our approach if (pro)nouns are not present in the sentence, then the conjugation (agreement suffixes on the verb) will be responsible for showing the subject and the object.

```
gramm("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)))
```

[6]If we have a similar sentence, but without a noun in instrumental case, the first argument of the predicate *'look-for'* will be an unknown referent *r000*.

```
SYNTAX:
gr("suff","stem","free",1,2,1,1)
gr("suff","stem","free",1,3,1,1)

DISCOURSE 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)])
```

It can be seen that the information that who loves is *me* (*r011*, where *0* means that this is an inbuilt referent, and *11* means singular, first person), and who is loved is you-singular (*r012*), and of course this knowledge could not come from anywhere else than from the two suffixes of the verb.

4.4 Copredicative Network

This component does not belong strictly to the parsing mechanism but is an abstract level between syntax and semantics showing which propositions "co-predicate" and how, and is useful in (machine) translation, because it preserves something from the original structure of the sentence, but the predicate-argument relations also appear in it.

The input of this level is the list of the relevant lexical items and the syntactic relations, and the output is the list of copredications. Let us show it through the familiar example.

```
1. gramm("Péter keresteti Marit a magyar rendörséggel.").

2. COPREDICATIVE NETWORK:
3. copr("look-for",2,1,"Peter",1,1,1,1,"arg")
4. copr("look-for",2,1,"Peter",1,1,1,0,"arg")
5. copr("look-for",2,1,"Mary",3,1,2,1,"arg")
6. copr("look-for",2,1,"Mary",3,1,2,0,"arg")
7. copr("cause",2,2,"Peter",1,1,1,1,"arg")
8. copr("cause",2,2,"Peter",1,1,1,0,"arg")
9. copr("cause",2,2,"look-for",2,1,2,0,"arg")
10. copr("the",4,1,"police",6,1,0,1,"free")
11. copr("Hungarian",5,1,"police",6,1,1,1,"free")
```

In line 3 it can be seen that there is a copredication between the predicate *look-for* (second word, first morpheme) and the argument *Peter* (first word, first morpheme), namely that the first argument (1) of the former is the first argument (1) of the latter (which is, in the case of arguments means that *itself*). There is another first argument of this predicate (line 4), which is the zeroth argument of *Peter*, namely its determiner pillar (it has already been mentioned that predicates look for their arguments on two pillars). The same can be found in lines 5-6, now with the second argument of the predicate. Lines 7-8 show that it is Peter that causes something, which is in line 9, and the second argument of the predicate *cause* is the zeroth argument of *look-for*, namely the event itself. These all have been regent-argument relations (*"arg"*), and there are two free-relations in lines 10-11.

4.5 Conjunction

Up to this point we have introduced our parser in general, and now we show our latest development, *conjunction.* First we worked out the case when two nouns are conjoined, but we claim that other types of conjunctions can be treated similarly.

To show how this works, we finally quote the whole output of our parser, when the grammaticality of the input string (1) is questioned, in which a conjunction (*és* 'and') can be found.

```
1. Mari és Juli kereshetik Pétert.
   Mary and Julie look-for - may - 3pl Peter-Acc
   'Mary and Julie may look for Peter.'

2. LEXICAL ITEMS:
3. Mari: n(1,1,li(m("","Mari",""),labstem("Mary",phonfst(2,2,0,2),1,[])))
4. és: n(2,1,li(m("","s",""),labstem("and",phonfsu(1,1,1,1),5,[])))
5. Juli: n(3,1,li(m("","Juli",""),labstem("Julie",phonfst(2,2,0,2),1,[])))
6. keres: n(4,1,li(m("","keres",""),
         labstem("look-for",phonfst(1,2,2,2),2,[["NOM","ACC"]])))
7. het: n(4,2,li(m("h","A","t"),labsuff("may",phonfsu(1,1,1,2),2,1)))
8. ik: n(4,3,li(m("","ik",""),labsuff("pl3def",phonfsu(2,3,1,3),2,3)))
9. Péter: n(5,1,li(m("","Pter",""),labstem("Peter",phonfst(1,2,0,2),1,[])))
10. t: n(5,2,li(m("V","t",""),labsuff("ACC",phonfsu(1,1,1,3),1,4)))

11. SYNTAX:
12. gr("noun","regent","conj",1,1,2,1)
13. gr("det","regent","conj",1,1,2,1)
14. gr("regent","noun","conj",2,1,1,1)
15. gr("regent","det","conj",2,1,1,1)
16. gr("noun","regent","subj",2,1,4,1)
17. gr("regent","noun","conj",2,1,3,1)
18. gr("regent","det","conj",2,1,3,1)
19. gr("det","regent","subj",2,1,4,1)
20. gr("noun","regent","conj",3,1,2,1)
21. gr("det","regent","conj",3,1,2,1)
22. gr("regent","noun","subj",4,1,2,1)
23. gr("regent","det","subj",4,1,2,1)
24. gr("regent","noun","obj",4,1,5,2)
25. gr("regent","det","obj",4,1,5,1)
26. gr("suff","stem","free",4,2,4,1)
27. gr("suff","stem","free",4,3,4,1)
28. gr("det","regent","__",5,1,4,1)
29. gr("suff","stem","free",5,2,5,1)
30. gr("noun","regent","obj",5,2,4,1)
31. regent-noun-conj: és-Mari
32. regent-det-conj: és-Mari
33. regent-noun-conj: és-Juli
34. regent-det-conj: és-Juli
35. regent-noun-subj: kereshetik-és
36. regent-det-subj: kereshetik-és
37. regent-noun-obj: kereshetik-Pétert
38. regent-det-obj: kereshetik-Pétert

39. COPREDICATIVE NETWORK:
40. copr("and",2,1,"Mary",1,1,1,0,"arg")
41. copr("and",2,1,"Mary",1,1,1,1,"arg")
42. copr("and",2,1,"Julie",3,1,2,0,"arg")
43. copr("and",2,1,"Julie",3,1,2,1,"arg")
```

```
44. copr("look-for",4,1,"and",2,1,1,1,"arg")
45. copr("look-for",4,1,"and",2,1,1,0,"arg")
46. copr("look-for",4,1,"Peter",5,1,2,1,"arg")
47. copr("look-for",4,1,"Peter",5,1,2,0,"arg")
48. copr("may",4,2,"look-for",4,1,2,0,"arg")
49. SEMANTICS:
50. provref("fixpoint",[e(4,2,1)])
51. provref("old",[r(1,1,1)])
52. pred("Mary",1,[r(1,1,1)])
53. provref("new",[r(2,1,1)])
54. provref("<or=",[r(2,1,1),e(4,1,1)])
55. pred("element",2,[r(1,1,1),r(2,1,1)])
56. pred("element",2,[r(3,1,1),r(2,1,1)])
57. provref("old",[r(3,1,1)])
58. pred("Julie",3,[r(3,1,1)])
59. provref("new",[e(4,1,1)])
60. pred("look-for",4,[e(4,1,1),r(2,1,1),r(5,1,1)])
61. provref("new",[e(4,2,1)])
62. provref("<",[e(4,2,1),e(4,1,1)])
63. pred("may",4,[e(4,2,1),e(4,1,1)])
64. provref("old",[r(5,1,1)])
65. pred("Peter",5,[r(5,1,1)])
66. yes
```

To start at the end, line 66 shows that the sentence is grammatical, which means that the program could identify the relevant lexical items (stems and affixes, lines 2-10) in an appropriate morphophonological environment [5], and the relevant syntactic relations, which can be seen in lines 11-38. Looking at the simplified representation (31-38), it can be read that there is a relation between the conjunction and Mary (31-32), and between the conjunction and Julie (33-34), and this conjunction is the one that establishes relation with the verb (lines 35-36). Lines 37-38 show that the object of the verb is Peter. The same is expressed in lines 39-48 but in a more abstract way, the copredicative network shows the relations between the (form-independent) semantic units.

Discourse semantics can be seen in lines 49-65. The main points are that a situation (*e411*) is probable (line 63), and this situation is that *r211* is looking for *r511* (60), where *r211* is a group consisting of *r111* (55) and *r311* (56), *r111* is Mary (52), *r311* is Julie (58), and *r511* is Peter (65).

In the case of verbs with *groups* as subjects, there arises a problem of agreement. To solve it, we have (partially) adopted Bánréti's suggestion [13] for our parser, namely that in these cases plural conjugation must be used with the "highest" ranked person ($1 > 2 > 3$) among all the co-ordinated nouns' persons. For example in the sentences *Én és te/ti/Péter kereshetjük Marit* (*I and you$_{sg}$/you$_{pl}$/Peter look-for-CAN-1PLdef*) 'I and you$_{sg}$/you$_{pl}$/Peter may look for Mary', the verb is in 1PL.

Another interesting problem is to interpret collective actions. The approach behind the present version of the parser is that collective reading illustrated above is generally sufficient in the semantic representation: it can be decided later if we need the referents one by one. There is only one case in which we assume a distributive reading, when the verb and the co-ordinated subjects are all in 3SG, for example in the sentence *Péter és János keresi Marit* (*'Peter and John* *look-for-3SGdef Mary-ACC*) 'Peter and John are looking for Mary'. In this case Peter and John are looking for Mary *separately*, while in the former example (with a verb in plural) they were doing it together. The semantic representation can show the difference.

```
1. DISCOURSE SEMANTICS:
2. provref("fixpoint",[e(4,1,1)])
3. provref("old",[r(1,1,1)])
4. pred("Peter",1,[r(1,1,1)])
5. provref("new",[r(2,1,1)])
6. provref("<or=",[r(2,1,1),e(4,1,1)])
7. pred("element",2,[r(1,1,1),r(2,1,1)])
8. pred("element",2,[r(3,1,1),r(2,1,1)])
9. provref("old",[r(3,1,1)])
10. pred("John",3,[r(3,1,1)])
11. provref("new",[e(4,1,1)])
12. pred("look-for",4,[e(4,1,1),r(1,1,1),r(5,1,1)])
13. provref("new",[e(4,1,2)])
14. pred("look-for",4,[e(4,1,2),r(3,1,1),r(5,1,1)])
15. provref("old",[r(5,1,1)])
16. pred("Mary",5,[r(5,1,1)])
```

Compared this distributive interpretation to the previous one, we can see that two actions are concerned instead of one (lines 11-14), the two subject are looking for *r511* separately. Nevertheless, of course, they form a group in this case as well (lines 7-8), because we can refer to their common referent.

5 Conclusion and future work

We hope that even this brief demonstration could prove that our program can provide much wider parsing than usual on a non-trivial lingustic fragment. We are planning to enlarge this fragment, extend our system to other languages, and develop further "intelligent" applications[7]. The guarantee for realizability is the theoretically "pure" and practically simplified processing. Growing out of the experimental phase, we are about to switch over to a more efficient programming language and query analizer, and we are seeking the opportunity to use some kind of corpus to simulate knowledge bases.

References

[1] Alberti, G.: GASG: Minimal Syntax, Maximal Lexicon and PROLOG. Paper read at ALLC/ACH '98, July 9. In Hunyadi, L. (ed.): ALLC/ACH '98. KLTE, Debrecen (1998) 81–83.

[7]Since the time of the conference (MSZNY2003), we have worked out the theoretical basis of machine translation, and now our parser can translate a particular group of sentences from Hungarian into English and vice versa (EAMT'04).

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

[3] Alberti, G., Balogh, K., Kleiber, J.: GeLexi Project: Prolog Implementation of a Totally Lexicalist Grammar. In de Jongh, Zeevat, Nilsenova (eds.): Proc. of the Third and Fourth Tbilisi Symp. on Language, Logic and Computation. ILLC, Amsterdam, and Univ. Tbilisi (2002).

[4] Alberti, G., Balogh, K., Kleiber, J.,Viszket, A.: A totális lexikalizmus elve és a GASG nyelvtan-modell [The Principle of Total Lexicalism and the GASG grammar model]. Maleczki, M. (szerk.): A mai magyar nyelv leírásának újabb módszerei V. Szegedi Tudományegyetem (2002) 193–218.

[5] Alberti, G., Balogh, K., Kleiber, J.,Viszket, A.: Towards a totally lexicalist morphology. Talk at 6th International Conference on the Structure of Hungarian (ICSH6), Düsseldorf, Germany (2002). To appear in Kenesei, I., Piñón, Ch. (eds.): Approaches to Hungarian 9.

[6] Alberti, G., Balogh, K., Kleiber, J.,Viszket, A.: Total Lexicalism and GAS-Grammars: A Direct Way to Semantics. In Gelbukh, A. (ed.): Proceedings of CICLing2003 (Mexico City). LNCS N2588. Springer-Verlag, Berlin Heidelberg New York (2003) 37–48.

[7] Alberti, G., Balogh, K., Kleiber, J.,Viszket, A.: A fordítás totálisan lexikalista megközelítése [The Totally Lexicalist Approach of Machine Translation]. MANYE, Számítógépes nyelvészeti szekció, Győr (2003).

[8] Alberti, G., Kleiber, J.: Extraction of Discourse-Semantic Information... In Cunningham, H., Paskaleva, E., Bontcheva, K., Angelova, G. (eds.): Information Extraction for Slavonic and Other Central and Eastern European Languages. Borovets, Bulgaria (2003) 63–69.

[9] Alberti, G., Kleiber, J.: The GeLexi MT Project. In J. Hutchins (ed.): Proceedings of EAMT 2004 Workshop, Valletta: Univ. of Malta (2004) 1–10.

[10] Balogh, K., Kleiber, J.: Egy lexikalista nyelvtan morfoszintaxisának PROLOG-implementációja [The PROLOG-implementation of the morphosyntax of a Totally Lexicalist Grammar]. JGYTF, Szeged (2001).

[11] Balogh, K., Kleiber, J.: Computational Benefits of a Totally Lexicalist Grammar. In Matouek, V., Mautner, P. (eds.): Text, Speech and Dialogue, Proceedings of TSD2003. Springer-Verlag, Berlin Heidelberg New York (2003) 114–119.

[12] Balogh, K., Kleiber, J.: A Morphology Driven Parser for Hungarian. Talk at the 5th Int. Tbilisi Symp. on Language, Logic and Computation. Org. by ILLC, Amsterdam, and U. Tbilisi (2003).

[13] Bánréti, Z.: A mellérendelés [Conjunction]. Kiefer, F. (szerk.) Strukturális magyar nyelvtan I. Mondattan. Akadémiai, Budapest (1992) 715–796.

[14] Borsley, R. D.: Modern Phrase Structure Grammar. Blackwell, Oxford Cambridge (1996).

[15] Bresnan, J.: Lexical Functional Syntax. Blackwell, Oxford (2000).

[16] Chomsky, N.: Syntactic Structures. The Hague, Mouton (1957).

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

[18] van Eijck, J., Kamp, H.: Representing discourse in context. In van Benthem, J., ter Meulen, A. (eds.): Handbook of Logic and Language. Elsevier, Amsterdam, The MIT Press, Cambridge, Mass. (1997).

[19] Joshi, A. K.: Starting with Complex Primitives Pays Off. In Gelbukh, A. (ed.): Proceedings of CICLing2003 (Mexico City). LNCS N2588. Springer-Verlag (2003) 1–10.

[20] Karttunen, L.: Radical Lexicalism. Report No. CSLI 86 68, Stanford (1986).

[21] Karttunen, L.: Computing with Realizational Morphology In Gelbukh, A. (ed.): Proceedings of CICLing2003 (Mexico City). LNCS N2588. Springer-Verlag (2003) 203–214.

[22] Partee, B., ter Meulen, G.B., Wall, R.P.: Mathematical Methods in Linguistics. Kluwer Academic Publ. (1990).

[23] Prószéky, G.: Megértéstámogatás és gépi fordítás: nyelvtechnológia a XXI. század elején [Text Understanding and Machine Translation: Language Technology in the Early Years of the 21st Century]. Eighth National Neumann Congress, Hungary (2003).

[24] Schubert, K.: Metataxis (Contractive Dependency Syntax for Machine Translation). Foris, Dordrecht (1987).

Received May, 2004