Word Order and Discontinuities in a Dependency Grammar for Hungarian

Csongor Barta[1], Ricarda Dormeyer[2], and Ingrid Fischer[2]

[1] Computer Science Department, University of Szeged, 6000, Kecskemét, Katona Zsigmond u.22., Hungary
barta.csongor@stud.u-szeged.hu
[2] Lehrstuhl für Informatik 2, Universität Erlangen-Nürnberg, Martensstr. 3, 91058 Erlangen, Germany
{Ricarda.Dormeyer, Ingrid.Fischer}@informatik.uni-erlangen.de

Abstract. Natural languages are always difficult to parse. Two phenomena that constantly pose problems for different formalisms are word order—what part of a sentence has to be placed where—and discontinuities—words that belong together but are not placed into the same phrase. Dependency grammar, a linguistic formalism based on binary relations between words, is very adequate for handling both problems. A parser for dependency grammar together with its grammar writing formalism is described in this paper. Word order and discontinuities in Hungarian are handled based on this formalism.

1 Introduction

At the Computer Science Department of the Friedrich-Alexander University Erlangen-Nuremberg a lot of different parsers have been developed over the years. One of the newest developments, a parser for dependency grammar [4], is currently tested for several different languages. Grammar fragments for English, German and Latin have been written [4]. These languages differ in their word order. English has a fixed word order, e.g. the subject has to come first in a declarative sentence. In German, the word order is semi-free. For noun phrases, it is fixed; on the sentence level, the verb has to be in the second position in a declarative sentence. Other elements can take the other positions, no restrictions are given here. In Latin there are even less word order restrictions. Words can be placed nearly everywhere.

But not only word order is of special interest. Another problem are words that belong together but are not placed next to each other in the sentence. This phenomenon can be found in all languages analyzed. An English example is fronting as in *Ann John told me he had seen.* In German and Hungarian verb prefixes can be separated from the verb and move to another position.

Now non-indo-European languages are researched. Currently grammars for Japanese [17] and Hungarian are developed. For Japanese, a lot of different

Fig. 1. A dependency tree for *János keresi Marit*

dependency grammar implementations exist. This is not the case for Hungarian. Only one international publication could be found containing a dependency grammar for Hungarian [18]. The grammar described in [18] differs a lot from our approach. In [18] first all prefixes and suffixes are separated from word stems. The resulting string is the input for the dependency parser. In our approach this separation does not take place, a sentence is analyzed in its original writing.

In the sequel, our dependency parser and the Hungarian grammar developed up to now are described. In section 2 the basics of dependency grammar are introduced. After this linguistic introduction, our dependency parser is specified in section 3. Special features of our Hungarian grammar are given in section 4. We end with a conclusion.

2 A Short Overview on Dependency Grammar

When taking a look in the standard literature [9] on computational linguistics, long introductions in phrase structure grammars invented by Chomsky can be found. They have been in the focus for nearly forty years now. Phrase structure grammars turned out to be a helpful method when modeling English; quite a lot of parsers can be found together with extensive grammars. But it also turned out that they are not useful when it comes to languages with free or semi-free word order. Discontinuous constituents and long distance dependencies pose difficulties, too. Several work arounds and extensions have been invented to overcome these problems. Some of these extensions and new developments, e.g. *Head-Driven Phrase Structure Grammar* [13], are similar to dependency grammar. Dependency grammar, invented by Lucien Tesnière [14] [15], is popular in Europe and Japan. In this theory binary relations between the words of a sentence are used as the basic construct. The most important part of a sentence is the verb, it opens several slots for other parts of the sentence. Taking the verb *keresi*, it opens two slots, one for a noun in the nominative case, which is the subject, and one for a noun in the accusative case, the object. In the sentence *János keresi Marit* these slots are occupied by *János*, the subject, and *Marit*, the object. Normally the relations are visualized with the help of trees. A tree for the running example is given in Figure 1. Please note, that every combination of the three words results in the same dependency tree: *János Marit keresi*, *Marit keresi János*, ...

The subject and object can also open new slots. A simple noun opens a slot for e.g. one determiner and any number including zero of adjectives. This means that several different kinds of slots are necessary. First slots are used that must

Fig. 2. A dependency tree for *Meg kell, hogy beszéljük a problémákat*

be filled, with one element. If the element is missing, the corresponding sentence is grammatically not correct. In Figure 1 exactly one object is needed. In English each verb needs exactly one subject. Then there are slots that are optional, they can be filled but do not have to be filled. Time and place are optional for most verbs. They can be added, but they can also be left out. Another example for this is the subject in Hungarian. Finally there are slots that can be filled several times, e.g. a noun can take several adjectives. A word opening a slot is also called the head, the word filling the slot will be called the dependent in the rest of the paper.

A more complicated sentence as *Meg kell, hogy beszéljük a problémákat* leads to the tree in Figure 2. This sentence contains a discontinuous constituent, another popular problem when analyzing natural languages. The word *meg* is followed by a word of the main clause, but syntactically it belongs to the verb of the subordinate clause *beszéljük*. In the tree in Figure 2 *meg* is drawn as a dependent of *beszéljük*. The linear structure of the words in this sentence cannot be reconstructed from Figure 2. In the tree only the syntactic structure is given. In phrase structure grammar linear and syntactic structure are combined in one tree leading to crossing edges in the phrase structure tree for this sentence. Trees with crossing edges cannot be constructed with context-free grammars.

3 A Parser for Dependency Grammar

Our parser [16] is based on three concepts. The parsing algorithm itself is similar to the well-known Cocke-Kasami-Younger algorithm for context-free phrase structure grammars [9]. The first dependency parser based on this idea was presented in [12]. Words are described by feature structures [9] enriched by a few symbols necessary for dependency grammars. Feature structures are combined with the help of graph unification. Our handling of discontinuous constituents and word order restrictions differs from [12].

3.1 Word Order

In Tesnière's original approach, word order was unimportant for syntactic description. Any order was allowed. This is not useful for parsers, too many wrong

sentences would be accepted. The order between head and dependent must be considered as well as the order between the different dependents of one head. Also the number of elements following or preceding a word can be important. E.g. in German the verb in a declarative sentence must fill the second position. In our approach each word has a position list where positions of the word itself and other words are described. This includes as a minimum a position for the word itself. Then each dependent of a word can have a fixed position in this position list. Free positions within the position list are also possible, a free position can take every dependent that is not marked as fixed. The position list makes parsing easier: When a fixed position is following according to this list, the parser has to check for just one element. If the next element is free, only free elements have to be checked.

3.2 Discontinuous Constituents

Linguistically, a discontinuous dependency can be regarded as a ternary relation, i.e. a relation between a dependent, its *syntactic head* and its *positional head* or linear head [1]. The syntactic head is the word containing a slot for the discontinuous dependent. constituent. But because the syntactic head's constituent is discontinuous, the dependent is positioned in the position list of the linear head. In the example sentence, the syntactic head for *meg* is *beszéljük* and its linear head is *kell*. The dependent fills a slot of its syntactic head, but occupies a position of the linear head's position list. Because of this, the processing of discontinuous dependencies starts with the linear head. When the parser finds the syntactic head, it attaches its unfilled slots to the linear head; when it finds the dependent, it allows the dependent to occupy a position in the linear head's position list while not yet filling a slot. At the end of the parsing of the linear head's position list, the open slots are filled by these dependents. A similar approach to parsing dependency structures can be found in [19].

3.3 Other Approaches

Over the years several other parsers for dependency grammar have been proposed. In [1] linear order and syntactic order are strictly separated, something we tried to avoid in our approach, because it makes grammar writing less intuitive and parsing less efficient. Fraser's parser [5] is based on backtracking and uses a parsing stack. Covington's approach [2] is cited very often. He invented a simple backtracking algorithm for free word order. But his approach is not well suited for semi-free word order phenomena. Finally there are several dependency parsers based on constraint resolution, a completely different approach [3].

4 Parsing Hungarian

In this section two Hungarian sentences are analyzed. With these examples our grammar description language is described.

```
Word "Angéla" <"Name"> [
    lexeme: Angéla;
    gender: fem;
    case: nom;]

Word "olvas"   <"VerbPres"> [
    lexeme: olvas;
    mood: declarative;
    number: sing;
    person: 3;]

Template "Name" [
    category: noun;
    special: propername;
    number: sing;
    person: 3;]
```

```
Template "VerbPres" [
    category: verb;
    form: finite;
    tense: present;
    sentence: declarative;
    subj:  oslot [
        category: noun;
        cont: +;
        case: nom;
    ];
    order: (%1 %2 i %3);
]
%1 = slot  [cont: +;];
%2 = mslot [cont: +;];
%3 = mslot [cont: +;];
```

Fig. 3. Simple grammar with templates for *Angéla olvas.*

4.1 Grammar Description Language and Free Word Order

The grammar description language is important, as it must be easy to learn and to handle for the linguist writing grammars for the parser. We will introduce it with the help of the easy example *Angéla olvas.* In Figure 3, the corresponding grammar is given. Please note that due to space our example grammars are not complete.

To shorten the grammar, templates as introduced by [6] are possible. Templates encode parts of the feature structures that are used very often. Within the entries for words template names are used instead of complete feature structures. As a first step, the lexicon is transformed before parsing. Template names are removed, the feature structure parts these names described are unified with the rest of the feature structure.

Feature structures are started by "[" and ended with "]", features and values are separated by ":" and feature-value-pairs are separated by ";". In Figure 3 two word entries and two templates are given. The lexical entry for *Angéla* contains a template named `Name`, which is also given. Feature structures for `Name` and *Angéla* are unified leading to an entry where agreement features as `number`, `gender`, `case` (not all possible cases are given) and `person` are described. Also the `lexeme` and `category` are shown. The verb *olvas* is also composed with the template for verbs in present tense. This template is more interesting. It contains an optional slot for a subject indicating that in Hungarian, the subject can be left out. At the end the special feature `order` indicates possible positions. As this position list is used for every word in present tense, it is more complicated than necessary for our small example. The symbol `i` stands for the position of the word itself, in this case the current verb. Before this verb at least one element must be placed. %1 must be a slot, this slot must be filled. %2 is marked `mslot` (multiple

```
Word "kell" [
    category: verb;
    lexeme: kell;
    subj: slot  [
        category: subjunction;
        lexeme: hogy;
        cont: +;];
    order: (%1 i %2);]
%1 = mslot;
%2 = mslot [cont: +;];

Word "beszéljük" [
    category: verb;
    lexeme: megbeszélni;
    subj: oslot [category: noun;];
    obj:  slot  [category: noun;];
    verbpart: slot [
        category: verbpart;
        lexeme: meg;];
    order: (%1 i %2);
]
%1 = mslot [cont: +;];
%2 = mslot [cont: +;];

Word "meg" [
    category: verbpart;
    lexeme: meg;]

Word "problémákat" [
    category: noun;
    lexeme: probléma;
    spec: %1 oslot [
        category: determiner;
        cont: +;];
    order: (%1 i);]

Word "a" [
    category: determiner;
    lexeme: a;]

Word "hogy" [
    category: subjunction;
    lexeme: hogy;
    prop: %1 slot [
        category: verb;
        cont: +;]
    order: (i %1);]
```

Fig. 4. Simple grammar for *Meg kell, hogy beszéljük a problémákat.*

slot). A multiple slot can be filled with an arbitrary number of elements but can also be empty. An arbitrary number of elements can also follow after the verb. It can also be described that the subject must go in the first position; in this case %1 has to be added to the subject slot. Please note that this is a lexicalised grammar, i.e. all information is stored in the lexicon, no extra grammar rules are needed.

4.2 An Example With a Discontinuity

Our treatment of free word order has already been introduced in the previous section. Now a more complicated example taken from [11] is used to show how discontinuities are handled. The dependency tree has already been shown in Figure 2. Discontinuity is quite common in Hungarian especially in the tenses and the verbal prefixes, e.g. *Szilvia el akar menni, Angéla akar elmenni.* In Figure 4 a short and not complete grammar for the sentence *Meg kell, hogy beszéljük a problémákat* is given. Templates are left out for simplicity. In the grammar, continuity and discontinuity are marked using special features. A feature `cont` is used to specify whether a dependency may be realized only continuously ("+"), only discontinuously ("-"), or both (not specified); a second feature `cont-const` is used to specify whether dependents may be extracted from the constituent

headed by a certain word[3]. Both features can be applied to lexical entries of words, to slots, or to positions in a position list. Specification of dependents, slots or positions as continuous will stop this process from taking place.

To parse the sentence for example 4, the parser first encounters the words *meg* and *kell*. *Meg* contains neither slots nor a position list and therefore cannot act as a head. The verb *kell*, on the other hand, has no slot for a `verbpart` dependent, but it has an `mslot` position to its left which may also be filled by a discontinuous dependent of one of its dependents. Therefore, a new analysis is generated where *meg* fills this position on the assumption that its syntactic head will turn out to be a word depending directly or indirectly on its positional head *kell*. The third word *hogy* is read but not yet linked to its head *kell*, since it has itself an obligatory continuous slot that must be filled first. The same applies to the next word *beszéljük*. Only after both *a* and *problémákat* have been read is the first slot filled: the specifier (`spec`) slot in *problémákat* can be filled by *a*. After that *problémákat* has no open slot or position left and can itself fill the object (`obj`) slot and part of the %2 position of the subordinate clause's verb. Now the verb has two slots left. The subject slot is optional, and the slot for the verbal prefix may be discontinuous. Therefore, the verb can be bound to its head *hogy*. After that *hogy* can be linked to its head. The `verbpart` slot is passed first to *hogy* and then to *kell*. Now, finally, the word *meg* that has been bound only positionally is assigned to its slot, namely to the discontinuous `verbpart` slot.

5 Conclusion and Future Work

It is most important to add a Hungarian morphology to the parser. Up to now, the parser works with a full form lexicon for Hungarian. This might be a solution for other languages, but it does definitely not work for Hungarian due to the high number of possible word endings.

Word order phenomena and discontinuity in Hungarian were modeled for a dependency parser. It turned out that, as for the other languages tested, it was possible and easy to write down. Nevertheless there is still a lot of work to do. Up to now only special problems of Hungarian have been modeled as a proof of concept for the parser. Next a full-flexed Hungarian grammar should be developed.

References

1. Norbert Bröker, *Separating surface order and syntactic relations in a dependency grammar*, in Proceedings of the 36th Annual Meeting of the ACL and 17th International Conference on Computational Linguistics, Montreal, 1998.
2. Michael A. Covington, *Parsing discontinuous constituents in dependency grammar*, Computational Linguistics, 16(4):234-236, 1990.

[3] No example for `cont-const` is given.

3. Ralph Debusmann, Denys Duchier and Geert-Jan Kruijff *Extensible Dependency Grammar: A New Methodology*, Recent Advances in Dependency Grammar, COLING 2004.
4. Ricarda Dormeyer, *Syntaxanalyse auf der Basis der Dependenzgrammatik*, PhD Thesis, Computer Science, Friedrich-Alexander University Erlangen-Nuremberg, 2004.
5. Norman M. Fraser, *Parsing and dependency grammar*, UCL Working Papers in Linguistics, 1:296-319, 1989.
6. Peter Hellwig, *Chart parsing according to the slot and filler principle*, in Proceedings of the 12th International Conference on Computational Linguistics, pages 242-244, Budapest, 1988.
7. Richard Hudson, *Word Grammar*, Blackwell, Oxford, 1984.
8. Richard Hudson, *Towards a computer-testable word grammar of English*, UCL Working Papers in Linguistics, 1:321-338, 1989.
9. Daniel Jurafsky and James H. Martin, *Speech and Language Processing, An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, Prentice Hall, New Jersey, 2000.
10. László Keresztes, *Hungaro Lingua: Praktische ungarische Grammatik*, Debreceni Nyári Egyetem, 1999.
11. Katalin Kiss, *The Syntax of Hungarian*, Cambridge Syntax Guides, Cambridge University Press, 2002.
12. Michael C. McCord, *Slot grammar. A system for simpler construction of practical natural language grammars*, in Rudi Studer, editor, Natural Language and Logic, pages 118-145. Springer, Berlin, Heidelberg, 1990.
13. Ivan Sag, Thomas Wasow and Emily Bender: *Syntactic Theory. A Formal Introduction*, Second Edition, Stanford: Univ. of Chicago Press, 2000.
14. Lucien Tesnière, Esquisse d'une syntaxe structurale, Klincksieck, Paris, 1953.
15. Lucien Tesnière, Eléments de syntaxe structurale, Klincksieck, Paris, 1959.
16. Thomas Tröger, *Ein Chartparser für natürliche Sprache auf der Grundlage der Dependenzgrammatik*, Master Thesis, Computer Science, Friedrich-Alexander University Erlangen-Nuremberg, 2003.
17. Alexandra Pröll, *Eine Dependenzgrammatik für das Japanische*, Bachelor Thesis, Computer Science, Friedrich-Alexander University Erlangen-Nuremberg, 2004.
18. Gábor Prószéky, Ilona Koutny and Balázs Wacha, *A dependency syntax of Hungarian*, in Dan Maxwell and Klaus Schubert, eds., Metataxis in Practice, pages 151-182. Foris Publications, Dordrecht, 1989.
19. Markus Schulze, *Ein sprachunabhängiger Ansatz zur Entwicklung deklarativer, robuster LA-Grammatiken*, PhD Thesis, Computer Science, Friedrich-Alexander University Erlangen-Nuremberg, 2004.