 

# FROM THE PRINCIPLE OF PROJECTIVITY AND PARTIAL ORDERINGS TO SOPHISTICATED LINEARIZATION CONSTRAINTS

**Vladimir Pericliev**

*Mathematical Linguistics Department*
*Institute of Mathematics & Informatics, bl.8*
*Bulgarian Academy of Sciences, 1113 Sofia, Bulgaria*
*email: peri@banmatpc.math.acad.bg*

Abstract: Some problems with the projectivity principle and partial orderings in dependency grammar are discussed, and a linearization component overcoming these problems is proposed. This component enriches the inventory of atomic ordering relations, which can further be combined in complex logical expressions. The domain of application of linearization rules is extended to non-local trees so that non-projective structures can also be described.

Keywords: dependency grammar, word order, parsing

## 1. INTRODUCTION

Natural languages normally exhibit intricate word order regularities, and in certain cases, this applies even to languages with proclaimed rigid word order (English) or total scrambling (Warlpiri, Latin). Hence the need for a sophisticated linearization mechanism in a grammatical formalism.

Dependency Grammar (DG) usually employs two ordering devices: so-called "Projectivity Principle", and partial orderings in local trees, showing the relative order of the head word and its immediate dependents (e.g. "subject precedes verb"). The Projectivity Principle by itself is inadequate since a grammar respecting it will both overgenerate and undegenerate,

ISBN: 0 08 043 438X

 

i.e. it is both insufficiently and overly constrained. The addition of partial orderings also does not guarantee the needed expressive power as regards correct linearizations.

In this contribution, I will show some problems with projectivity and partial orderings, and will then outline a linearization component for a DG. Briefly, the proposal amounts to enriching the inventory of atomic ordering relations, allowing complex logical expressions to be formed from these ordering relations, and finally, allowing these rules to apply to non-local trees in order to capture correct linearizations in non-projective structures.

## 2. PROBLEMS WITH THE PRINCIPLE OF PROJECTIVITY AND PARTIAL ORDERINGS

A sentence is said to be "projective" just in case no intersection occurs in its dependency tree diagram; otherwise the sentence is said to be projective. The Principle of Projectivity (also familiar as the "Adjacency Principle") bans the generation of non-projective structures (analogously to the banning of discontinuous structures in constituency systems).

Consider a simple example from Bulgarian, comprising a verb, an object noun and its possessive pronoun modifier:

| *Izmij (verb)* | *si (pron)* | *liceto (noun)* |
|---|---|---|
| Wash | your | face |

(meaning: Wash your face!)

The dependencies will be obvious: the verb is the root of the tree, (immediately) dominating the noun, which in turn, (immediately) dominates its modifier.

The following list gives all theoretically admissible permutations of this string (3! = 3x2x1 = 6), together with an assessment of their grammaticality and projectivity (assuming the above dependencies):

| | | Grammaticality | Projectivity |
|---|---|---|---|
| 1. *Izmij si liceto* | (verb pron noun) | + | + |
| 2. *Izmij liceto si* | (verb noun pron) | + | + |
| 3. *Liceto izmij si* | (noun verb pron) | + | - |
| 4. *Liceto si izmij* | (noun pron verb) | + | + |
| 5. *Si liceto izmij* | (pron noun verb) | - | + |
| 6. *Si izmij liceto* | (pron verb noun) | - | - |

ISBN: 0 08 043 438X

 

The list clearly shows that a DG which uses the Projectivity Principle (i.e. blocks all non-projective structures) will be over-constrained since it will exclude the non-projective, but grammatical sentence (3). A DG which does *not* use the Projectivity Principle (and generates also all non-projective structures) will be under-constrained since it will admit the ungrammatical non-projective sentence (6). We may call this the PROJECTIVITY PROBLEM of DGs.

Therefore, if a DG is to be empirically adequate (i.e. if it is to generate all and only the grammatical strings of a language), it should not a priori either exclude or generate all non-projective structures. Rather, it should generate some non-projective structures, viz. those with correct orderings (in our case, (3)), and exclude others, viz. those with incorrect ones (in our case, (6)). We may say, in this sense, that the Projectivity Principle is both too strong and too weak a constraint on DGs (despite the fact that, as a general tendency, it quite nicely captures our intuitions about ordering, to the effect that words that go together are positioned near to one another).

The distinction projectivity/non-projectivity, as is well known, has a rather asymmetrical effect: non-projectivity in fact much better accounts for what is a misordering in a language than projectivity does for what is a correct ordering. Hence a DG would normally use--in addition to the projectivity principle, filtering non-projective sentences--partial orderings of the type A < B ("node A precedes node B", where a node may be a complex feature structure), to impose a correct linearization on projective structures. The domain of application of these rules are "local trees", i.e. the head and its immediate dependents. The question is: Would this apparatus suffice to handle correctly our data set?

Now, we are allowed to order:

verb (head) and noun (dependent)

noun (head) and pron (dependent)

It will be clear that it is impossible to generate all and only the grammatical sentences (1) through (4). Thus, in these sentences the verb either precedes the noun (exs. (1) and (2)) or follows it (exs. (3) and (4)), and an analogous situation holds for the other pair we may impose an ordering on, viz. noun and pronoun. If we allow free permutation in both pairs, this will result in the generation of the ungrammatical sentence (5), i.e. overgeneration. Imposing any precedence on either pair of nodes will, of necessity, lead to undergeneration, e.g. declaring verb < noun will exclude the grammatical exs. (3) and (4); declaring noun < pronoun--in order to exclude the ungrammatical (5)--results in blocking the grammatical (1), etc.

What we actually need is to declare that the possessive pronoun *si* 'your', which is a clitic in Bulgarian, never occurs clause-initially, whereas all other permutations are admissible; unfortunately, no such declarations are possible with the mentioned apparatus. We shall call the impossibility to state certain orderings the ORDERING PROBLEM of DGs.

ISBN: 0 08 043 438X

 

Therefore, the usual ordering devices in DG, the Projectivity Principle, and the partial ordering of pairs of nodes in local trees, are insufficient to handle the intricate orderings that may occur in natural languages. Hence a DG needs linearization rules with greater expressive power.

## 3. A LINEARIZATION COMPONENT

There are different ways of constructing a linearization component and incorporating it into a DG. My own proposal is that--analogously to other formalisms like GPSG or HPSG--a DG should have separate components for dpendency relations (which just states dependency and no ordering), and for linearization, where "Linear Precedence Rules" (e.g. adj < noun) hold "globally", that is throughout the whole grammar. (Not stating any precedence between two nodes means that these nodes may occur in any relative position.)

The formalism proposed introduces further word order restrictions in addition to "X precedes Y" (enabling it to avoid the ordering-problem), and also extends the domain of application of these restrictions to non-local trees (in order to handle the projectivity-problem).

In the dependency part of a DG, there are two types of nodes: non-contiguous ones (notated: #Node) and contiguous ones (notated just: Node). A contiguous node shows that this node (=head) and its daughters (=dependents) form a contiguous sequence, whereas a non-contiguous node shows non-contiguity, i.e. the occurrence, among its daughters, of nodes governed from outside of this local tree.

*Example 1*

Below is the grammar, expressed in this formalism, of the Latin sentence: "Puella bona puerum parvum amat" (good girl loves small boy), which is grammatical in all its permutations (5!=120), and, besides, having non-projectivity in the noun phrases. This is captured with the following dependency rules with *no* ordering restrictions, which allows free permutation of all 5 words:

s ==> #verb.

verb ==> #noun(nom), #noun(acc).

noun(nom) ==> adj(nom).

noun(acc) ==> adj(acc).

accompanied by the "dictionary" rules:

ISBN: 0 08 043 438X

 

verb ==> amat.

noun(nom) ==> puella.

noun(acc) ==> puerum.

adj(nom) ==> bona.

adj(acc) ==> parvum.

In order to solve the ordering-problem, the following additional atomic ordering constraints are introduced:

*Precedence constraints:*

- precedes (e.g. a < b)
- immediately precedes ( a << b)

*Adjacency constraints:*

- is adjacent (a <> b)

*Position constraints:*

- is positioned first/last (e.g. first(a,Node), where Node is a node; e.g. first(a,s) designates that "a" is sentence-initial.

Atomic word order constraints are allowed to combine into complex logical expressions, using the following operators with obvious semantics:

- Conjunction (notated: *and*)
- Disjunction (*or*)
- Negation (*not*)
- Implication (*if*, e.g. (a << b) if (a < c) )
- Equivalence (*iff*, e.g. (a << b) iff (a< c) )
- Ifthenelse (*ifthenelse*)

ISBN: 0 08 043 438X

 

This linearization language is, of course, partly logically redundant (e.g. immediately precedence may be expressed through precedence and adjacency, and so is the case with the last two of the operators, etc.). However, what is logically is not necessarily psychologically equivalent, and the goal has been to maintain a linguist-friendly notation.

*Example 2*

Let us look at the Bulgarian example in the previous section. Here are the dependency rules (omitting dictionary rules):

s ==> verb.

verb ==> #noun.

noun ==> pron.

(the notation "#noun", as explained above, means that between the head noun and its dependent pron(oun), (an)other word(s) may be embedded). To generate all and only the grammatical permutations, we need to state only that the pronoun cannot occur first, so the linearization rule:

not first(pron,s).

(i.e. "pron" is forbidden to occur sentence-initially).

*Example 3*

As another simple illustration of the expressive power of the linearization language, consider the word order Universal 20 (of Greenberg and Hawkins) to the effect that NPs comprising dem(onstrative), num(eral), adj(ective) and noun can appear in that order, or in its mirror-image. We can write a "universal" rule enforcing adjacent permutations of all nodes with the following three ordering rules:

noun ==> dem,num,adj.

dem <> num.

num <> adj.

adj <> noun.

ISBN: 0 08 043 438X

 

## 4. CONCLUSION

Different versions of DG may handle linearization in general, and specific linguistic phenomena in particular, applying different mechanisms suiting best their general framework. My proposal was for an explicit linearization component with a linguist-friendly notation, i.e. rules, possibly of complex structure, that closely approximate common linguistic parlance in stating word ordering.

The basic empirical motivation for this proposal is the study of projectivity and other phenomena in Bulgarian, specifically the ordering of clitics. The Bulgarian clitic system exhibits quite intricate regularities: clitics do not occur clause-initially (unlike, say, in Macedonian); the same word behaves as a proclitic under some conditions, and as an enclitic in others; clitics cluster in specific, often non-projective, sequences, etc. Similarly complex orderings are of course to be encountered in other languages as well, and this requires adequate rules for their expression.

The proposed notation is implemented into a parsing system which, in its original presentation (Pericliev and Grigorov, 1994), accepts constituency, rather than dependency, grammars as input. A modification of this parser allows the system to be used for dependency systems as well. Working both as a parser and generator, the system is a convenient tool for the development and testing of grammars involving complex word order regularities.

*Acknowledgement.* This work was partly supported by contract I-526 with the Bulgarian Ministery of Education, Science and Technology.

## REFERENCE

Pericliev, V. and A. Grigorov (1994). Parsing a flexible word order language. In: COLING, *Proceedings of the 15th International Conference on Computational Linguistics.* Kyoto, Japan.

ISBN: 0 08 043 438X