Natural Language Processing Using Generative Indirect Dependency Grammar

Stefan Diaconescu

SOFTWIN Str. Fabrica de Glucoza Nr. 5, Sect.2, 020331 Bucharest, ROMANIA
sdiaconescu@softwin.ro

Abstract. Abstract. This paper presents an extension of the Generative Dependency Grammar (GDG) and GDG with Feature Structure (FS). The extension consist of the indirect dependency notion introduction. The direct dependencies allows to expres some relations between elements (for exemples non-terminals) belonging to the the right side of a single grammar rule. The indirect relations will allow to express relations between elements belonging to the right side of diferent grammar rules. This feature allows to write shorter grammar rules and therefore more compact grammars. This will facilitate the writing of grammar rules, thing that is not a trivial one.

1 Introduction

Generative Dependencies Grammars (GDG) and Generative Dependencies Grammars with Features (GDGF) [4] merge advantages of Generative Grammars (GG) [2] and Dependencies Grammars (DG) [12] [5] [6] [7] [11] [8] i.e. they merge the representation of the phrasal categories from GG with the handling of discontinuous structures like gaps and non projective constructions from DG. GDG keeps from GG the idea of sequence of words that can be considered an expression of the syntax and from DG the idea of relation between words or grammar categories that can be considered an expression of the semantics. Each grammar rule from GDG contains in the right side two sequences: a sequence of non-terminals, terminals, pseudo-terminals or direct actions (*ntpa*) - this is the "generative" part of the rule - and a sequence of different types of dependencies between *ntpa* - this is the "dependency" part of the rule. GDG and GDGF have the limitation that the dependencies from dependency part of the rules can express relations between the *ntpa* belonging to the generative part of the same rule. We will name this kind of relations direct relations and we will rename GDG as GDDG. In this paper we will describe a new kind of GDG grammars that allows to express relations between *ntpa* that belong to the generative part of different grammar rules. We will name this kind of relations indirect relations and the corresponding grammars Generative Indirect Dependency Grammar (GIDG) and respectively Generative Indirect Dependencies Grammar with Feature (GIDGF).

2 The Notion of Indirect Dependency

Let us take a simplified grammar fragment (in BNF):

1.<rule>::=<non-terminal$_0$><subordinate$_1$>

2.<non-terminal$_0$>::=<non-terminal$_1$><subordinate$_2$>

3.<subordinate$_1$>::=<non-terminal$_2$><subordinate$_3$>

We suppose that the dependency relations that we want to represent are as follows:

a) The non-terminal <subordinate$_1$> from the rule 1 is subordinate to the non-terminal < non-terminal$_0$> from the same rule 1.

b) The non-terminal <subordinate$_2$> from the rule 2 is subordinate to the non-terminal <subordinate$_1$> from the rule 1.

c) The non-terminal <subordinate$_3$> from the rule 3 is subordinate to the non-terminal <non-terminal$_0$> from the rule 1.

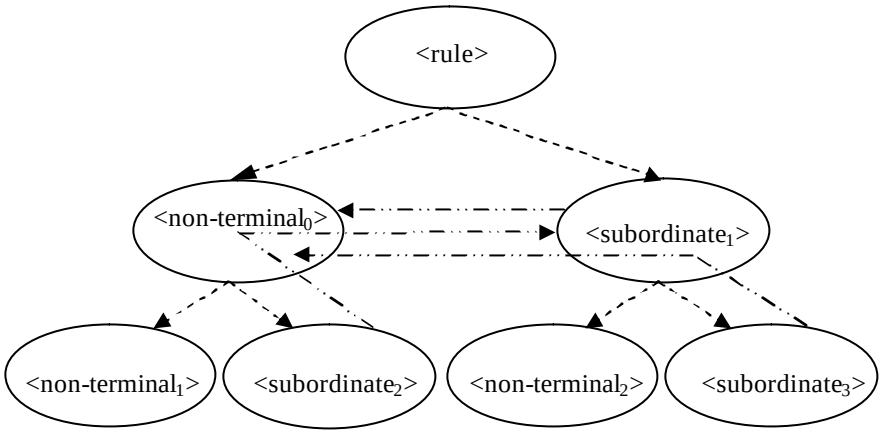

Fig. 1. Indirect Dependencies

In order to realize the request from the point (a) we must introduce a "direct" subordinate relation @relation1@ between <subordinate$_1$> and <non-terminal$_0$>. In order to realize the requests from points (b) and (c) we must introduce some "indirect" or external relations that link non-terminals from the right part of different rules. We will define in the rule 1 two relations @@relation$_2$@@ and @@relation$_3$@@. These relations will be carried out in the rules that expands the non-terminals from the rule 1. The relation @relation$_2$@ will arrive in the rule 2 and the relation @relation$_3$ will arrive in the rule 3. In the rules 2 and 3 will be introduced some "indirect" external relation @@relation$_2$@ and @@relation$_3$@@ that will be targets for the relations with the same name from the rule 1.

In the next sections we will define a grammar that can express such relations.

3 The Dependency Tree

The GIDG is based on Dependency Trees (DT) so we will define first of all the DT. Let us have the following sets:

- N is the set of non-terminals n having a name written between <...>.

- T is the set of the terminals t written between "...".

- P is the set of pseudo-terminals p i.e. the set of the non-terminals that contain only terminals. The name of a pseudo-terminal will be written between %...%.

- A is the set of the procedural actions a i.e. the set of the routines that can be used to represent a certain portion of the text that we analyze. The name of a procedural action will be written between #...#.

- SR is the set of the subordinate relations sr i.e. the set of the relations between *N, T, P, A, CR*, respecting some rules. The links that enter in an sr come from one element that is considered to be subordinated to the elements that receive links that comes from this sr. The name of an sr will be written between @...@.

- CR is the set of the coordinate relations cr i.e. the set of the relations between *N, T, P, A, SR* or other CR respecting some rules. The links that enter in a cr come from some elements that are considered to be coordinated each other but also from some other elements. The links that come from the coordinated elements (usually 2) are named fixed entries. The other entries are named supplementary entries. The name of a cr will be also written between @...@.

Definition: A DT is a tree that has nodes from *N, T, P, A, SR, CR* and the oriented branches between nodes respects the following rules:

- An $ntpa$ (an element from N or T or P or A) can have zero or one output (i.e. a link oriented from the $ntpa$ to other element) that goes to an sr or to a cr on fixed entries and zero, one or many inputs (i.e. links from other elements that are oriented to the $ntpa$) that come from an sr. It will be noted by $ntpa(i_1, i_2, ...)$ or by $ntpa()$ if no entry is available.

- An sr has always an output that goes to an $ntpa$ or to a cr on a supplementary entry and one input that comes from an $ntpa$ or from a cr. It will be noted by $sr(i_1)$.

- A cr can have zero or one output that goes to an sr or to a cr on fixed entry and can have $m >= 2$ fixed inputs that come from $ntpa$ or from cr and zero, one or many supplementary inputs that come from sr. It will be noted by $cr(f_1, f_2, ... / s_1, s_2, ...)$ if it has supplementary inputs (where $f_1, f_2, ...$ are fixed entries and $s_1, s_2, ...$ are supplementary inputs) or by $cr(f_1, f_2, ...)$ if it has not supplementary inputs. We will consider $m = 2$ because this is used about correlative constructions that have two members.

An important feature of the DT is the head. A head in a DT is a node that has only inputs and has no output. An *ntpa* or *cr* node can be head but an *sr* relation cannot be head. A dependency tree has one and only one head. If a DT has an *ntpa* as head, it will be named *ntpa* head DT. If a DT has a coordinate dependency relation as head it will be named coordinate head DT.

A dependency tree that do not contains non-terminals will be named final tree.

4 The Generative Indirect Dependency Grammar

We will give now the following definition of the GIDG:

Definition: A Generative Indirect Dependency Grammar GIDG is an 8-tupple GIDG = {*N, T, P, A, SR, CR,* t_0*, R*} where *N, T, P, A, SR, CR* were defined above and:

- t_0 belongs to *N* and is named root symbol.

- *R* is a set of numbered rules of the form (i) n_i -> (s_i, q_i) where; n_i belongs to *N*; s_i is a sequence of *ntpa*, q_i is a sequence of one or more comma separated dependency trees having nodes from s_i (i = 1, 2, 3, ...). The first DT must be a non-terminal DT and the other DTs can be non-terminal DT or external DTs (see section 5). If there is not a first non-terminal DT, then q_i begins with two commas.

The next conditions must be respected:

- All *ntpa* from s_i must be found one time and only one in q_i.

- All *ntpa* from q_i must be found one time and only one in s_i.

- If s_i, and/or q_i contain the same *ntpa* , *sr* or *cr* many times, then the apparition of the same element will be differently labelled in order to distinguish them.

Example of GIDG:

Let us have the romanian phrase: "Pe fata barbatul a vazut-o sanatasa." A grammar that can generate this phrase is the following (we use romanian grammar categories and the following notations: gsc = complex subjective group, gpc = complex predicative group, sub = subject, pred = predicate, cd = direct complement, cd abv = abbreviated direct complement, eps = supplementary predicative element, s-p = subject - predicate relation, p-cd = predicate - direct complement relation, cd-eps = direct complement - supplementary predicative element relation):

(1) <phrase> -> (<gsc> <gpc>, <gsc>(@s-p@(<gpc>(@@p-cd@@()))))

(2) <gsc> -> (<cd> <sub>, <sub>(), @@p-cd@@(<cd>(@@cd-eps@@())))

(3) <gpc> -> (<pred> <cd abv> <eps>, <pred>(@p-cd@(<cd abv>())), @@cd-eps@@(<eps>()))

(4) <cd> -> ("pe fata", "pe fata"())

(5) <sub> -> ("barbatul", "barbatul"())

(6) <pred> -> ("a vazut","a vazut"())

(7) <cd abv> -> ("-o", "-o"())
(8) <eps> -> ("sanatoasa", "sanatoasa"())

5 Indirect Dependency Tree Substitution and Links

We will define the following external types of DT:

a) The external subordinate tail DT is a DT that contains a subordinate relation that has not output (it is a DT having as had a subordinate relation).

b) The external subordinate head DT is a DT that contains a subordinate relation that has no input.

c) The external coordinate tail DT is a DT that contains a coordinate relation that has no output.

d) The external coordinate head DT is a DT that contains a coordinate relation that has no at least one fixed input.

An external DT can be of any combination of this four types. An *ntpa* or coordinate head DT can also be an external *ntpa* or external coordinate head DT. Between two *ntpa* head DT or between two coordinated head DT we will define the substitution operation. Between an external subordinate head or external coordinate head DT and an external subordinate tail or external coordinate tail DT we will define the link operation.

The substitution operation is defined as follows:

Let us take a DT w that contains a non-terminal head DT p = <p'>(u_1, u_2, ...) where <p'> is a non-terminal and u_1, u_2, ... are DTs that act as entries in the non-terminal <p'>. It is possible that <p'> has no entries at all. It is possible too, that <p'> have zero outputs (if it is a head of w) or one output that go somewhere in the tree w. To substitute p from w by a DT q means to obtain a new tree s that will replace p in w. If q = <q'>() is a non-terminal DT without entries, then s = <q'>(). If q = <q'>(v_1, v_2, ...) is a non-terminal DT where <q'> is a non-terminal and v_1, v_2, ... are dependency trees that act as entries in the non-terminal <q'>, then s = <q'>(u_1, u_2, ..., v_1, v_2, ...). If q = @q'@(v_1, v_2 / t_1, t_2, ...) is a coordinate relation DT where @q'@ is a coordinate relation, v_1, v_2, ... are DTs that act as entries in the fixed entries of the coordinate relation @q'@ and t_1, t_2, ... are DTs that act as entries in the supplementary entries of the coordinate relation @q'@ (it is possible that q' has no supplementary inputs at all.), then s = @q'@(v_1, v_2 / t_1, t_2, ..., u_1, u_2, ...).

The link operation is defined as follows:

If p = @@r@@(w) is a subordinate tail and if q = v(@@r@@()) is a subordinate head DT, than p linked with q will give a DT s = v(@r@(w)). We can see that the external relation becomes a normal relation.

If p = @@r@@(x_1, x_2 / u_1, u_2, ...) is a coordinate tail DT (x_1 can be empty, x_2 can be empty, u_1, u_2, ... can be empty too), q = v(@@r@@(y_1, y_2 /t_1, t_2, ...)) is a coordinate head DT (y_1 can be empty, y_2 can be empty, t_1, t_2, ... can be empty too), x_1 and y_1 can not be both non empty and x_2, y_2

can not be both empty, than p linked with q will give a DT s = v(@r@(z_1, z_2 / t_1, t_2, ..., u_1, u_2, ...)) where $z_1 = x_1$ if x_1 is not empty, $z_1 = y_1$ if y_1 is not empty, $z_2 = x_2$ if x_2 is not empty and $z_2 = y_2$ if y_2 is not empty.

Using these two operations we can define the generation process in GIDG.

6 Generation in a GIDG

We consider 2 rules (i) p_i -> (s_i, q_i) and (j) p_j -> (s_j, q_j) of a GIDG. q_i is formed by a non-terminal DT or by a coordinate DT w$-i$ (that can be also a subordinate head DT or a coordinate head DT) and eventually by one or many external DT $u_{i,1}$, $u_{i,2}$, ... q_j is formed by a non-terminal DT or by a coordinate DT w$-j$ and eventually by one or many external DT $u_{j,1}$, $u_{j,2}$, ... We consider also that s_i and q_i contains p_j. We will obtain a new rule (k) p_k -> (s_k, q_k) where q_k is formed by a non-terminal DT or by a coordinate DT w_k (eventually empty) and eventually by one or many external DT $u_{k,1}$, $u_{k,2}$, ... , as follows:

a) $p_k = p_i$;

b) s_j will replace p_j from s_i obtaining s_k;

c) q_k is obtained from q_i and q_j as follows:

- w_j will substitute p_j in w_i giving w_k or will substitute p_j in $u_{i,1}$, $u_{i,2}$, ... giving $u_{k,1}$, $u_{k,2}$, ... (only one substitution must take place).

- All u_i from q_i that can not be linked with some u_j from q_j will be put in q_k;

- All u_j from q_j that can not be linked with some u_i from q_i will be put in q_k.

d) Finally, any pair of DTs from s_k that can be linked together (but one such DT can participate only in one pair) is replaced by the result of this link. The non-terminal or coordinate DT will remain on the first position in s_k.

We will say that the rule (k) is immediately generated from the rule (i) using the rule (j). If, by applying a sequence of generations (starting with a rule that has a root symbol in the left side), we will obtain finally a rule (h) p_h -> (s_h, q_h) where s_h is a phrase, we will say that s_h is completely generated by GDG (or that s_h is accepted by grammar GDG). In this case, q_h will be the final dependency tree of the phrase s_h.

Remark: When a grammar rule is applied many times, all the elements of the rule are differently labelled for each application.

Example of generation in GIDG:

We take the example from the section 2. A generation can be:

(9)(1, 2) <phrase> -> (<cd> <sub> <gpc>, <sub>(@s-p@(<gpc>(@@p-cd@@()))), @@p-cd@@(<cd>(@@cd-eps@@())))

(10)(9, 3) <phrase> -> (<cd> <sub> <pred> <cd abv> <eps>, <sub>(@s-p@(<pred>(@p-cd@(<cd abv>()), @p-cd@(<cd>(@cd-eps@(<eps>( )))))))))

(11)(10, 4) <phrase> -> (”pe fata” <sub> <pred> <cd abv> <eps>, <sub>(@s-p@(<pred>(@p-cd@(<cd abv>()), @p-cd@(”pe fata”(@cd-eps@(<eps>( )))))))))

(12)(11, 5) <phrase> -> (”pe fata” ”barbatul” <pred> <cd abv> <eps>, ”barbatul”(@s-p@(<pred>(@p-cd@(<cd abv>()), @p-cd@(”pe fata”(@cd-eps@(<eps>()))))))))

(13)(12, 6) <phrase> -> (”pe fata” ”barbatul” ”a vazut” <cd abv> <eps>, ”barbatul”(@s-p@(”a vazut”(@p-cd@(<cd abv>()), @p-cd@(”pe fata”(@cd-eps@(<eps>()))))))))

(14)(13, 7) <phrase> -> (”pe fata” ”barbatul” ”a vazut” ”-o” <eps>, ”barbatul”(@s-p@(”a vazut”(@p-cd@(”-o”()), @p-cd@(”pe fata”(@cd-eps@(<eps>()))))))))

(15)(14, 8) <phrase> -> (”pe fata” ”barbatul” ”a vazut” ”-o” ”sanatoasa”, ”barbatul”(@s-p@(”a vazut”(@p-cd@(”-o”()), @p-cd@(”pe fata”(@cd-eps@(”sanatoasa”()))))))))

We obtained in the generative part the surface phrase text and in the dependency part the DT corresponding to the phrase.

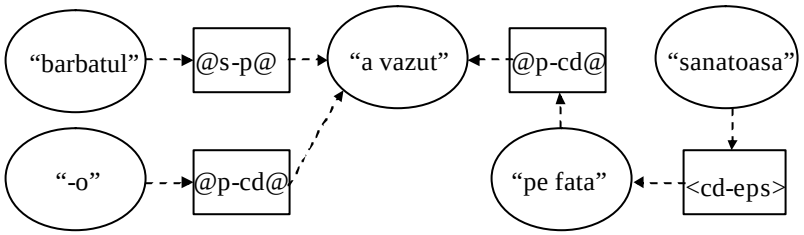

Fig. 2. Dependency Tree Example

Starting from the GDDG and the notion of AVT - Attribute Value Tree in [3][4] is defined the GDDG with feature (GDDGF). GDDGF can represent in a very compact manner a great number of grammar rules. An analog definition allows to obtain GIDGF from GIDG.

Definition: A GIDG with feature structure is a GIDG where each *ntpa* can have associated an AVT. The AVT associated with the non-terminal from the left side of the rules have always only indexed attributes (i.e. attributes that have the same value in all the *ntpa* of the rule).

As we can see GDDGF is a particular case of GIDGF but GIDGF combine the advantages of the GIDG with the AVT.

7 Conclusions

We made the extension of GDDG to obtain GIDG by using indirect relations that link external *ntpa* to the current rule. These relations facilitate the writ-

ing of the grammar rules, especially for long distance relation between phrase parts, without the necessity to have all these parts in the same rule.

By this extension we obtained rules that have in the dependency part many DT-s. Only one of these DT-s is non-terminal DT or coordinate DT, the others DT-s being external DT-s. GIDG can be further extended to have many non-terminal or coordinate non-external DT-s in the dependency part. This will facilitate the writing of recursive rules.

The indirect relations was introduced in GRAALAN (GRAmmar Abstract LANguage) that has many others features also. GRAALAN is currently used to write the grammar of the Romanian language.

References

1. Bröker, Norbert (1998) How to define a context free backbone for DGs: Implementing a DG in the LFG formalism, Papers of the Workshop on the Processing of Dependency-based Grammars (COLING-ACL'98), 29–38
2. Chomsky, Noam (1998) Generative Grammar: Its Basis, Development and Prospects. Studies in English Linguistics and Literature, Special Issue, Kyoto University of Foreign Studies
3. Diaconescu, Stefan (2003) Morphological Categorization Attribute Value Tree and XML, in Mieczyslav A. Klopotek, Slawomir T. Wierzchon, Krzysztof Trojanowski (Eds), Intelligent Infrmation Processing and Web Mining, Proceedings of the International IIS: IIPWM'03 Conference, Zakopane, Poland, June 2-5, Springer, 131–138.
4. Diaconescu, Stefan (2002) Natural Language Understanding Using Generative Dependency Grammar, in Max Bramer, Alun Preece and Frans Coenen (Eds), in Proceedings of ES2002, the 21-nd SGAI International Conference on Knowledge Based Systems and Applied Artificial Intelligence, Cambridge UK, Springer, 439–452
5. Gaifman, H. (1965) Dependency Systems and phrase structure systems, Information and Control, 304–337
6. Hudson, R. (1990) English Word Grammar, Oxford UK, Basil Blackwell
7. McCord, M Slot grammar (1990) A system for simpler construction of practical natural language grammars, in R. Studer (Ed.), Natural Language and Logic, Berlin Heidelberg Springer, 118–145
8. Milward, D. (1994) Dynamic Dependency Grammar, Linguistics and Philosophy 17, December, 561–606
9. Pollard, C., Sag, I. A. (1994) Head-driven Phrase Structure grammar, University of Chicago Press and Standford CSLI Publications
10. Shieber, Stuart M. (1986) An Introduction to Unification Based Approaches to Grammar, CLSII Lecture Notes Series, Number 4, Center for the study of Language and Information, Stanford University
11. Starosta, S (1992) Lexicase revisited. Department of Linguistics, University of Hawaii
12. Tesnière, L. (1959) Éléments de syntaxe structurelle, Paris, Klincksieck