TALN 2003, Batz-sur-Mer, 11–14 juin 2003

Contextual Grammars and Dependency Trees *

Radu Gramatovici (1), Carlos Martín-Vide (2)

(1) Faculty of Mathematics and Computer Science, University of Bucharest
Academiei 14, 70109, Bucharest, Romania
Email: radu@funinf.cs.unibuc.ro

(2) Research Group on Mathematical Linguistics, Rovira i Virgili University
Pl. Imperial Tàrraco 1, 43005 Tarragona, Spain
Email: cmv@astor.urv.es

Mots-clefs – Keywords

Grammaire contextuelle, arbre de dependance, arbre projectif de dépendance
Contextual grammar, dependency tree, projective dependency tree

Résumé - Abstract

On présente une nouvelle variante de grammaire contextuelle structurée, qui produit des arbres de dépendance. Le nouveau modèle génératif, appelé grammaire contextuelle de dépendance, améliore la puissance générative forte et faible des grammaires contextuelles, tout en étant un candidat potentiel pour la description mathématique des modéles syntactiques de dépendance.

A new variant of structured contextual grammar, which generates dependency trees, is introduced. The new generative model, called dependency contextual grammar, improves both the strong and weak generative power of contextual grammars, while being a potential candidate for the mathematical description of dependency-based syntactic models.

*This work was written during a research visit of the first author at the Research Group on Mathematical Linguistics of Rovira i Virgili University. The research visit was funded by the NATO Scientific Committee.

1 Introduction

Contextual grammars were introduced in 1969 by Solomon Marcus (Marcus, 1969) as an attempt to transform in generative devices some procedures developed within the framework of analytical models (see (Marcus, 1997) for a comprehensive discussion on the linguistic motivations of contextual grammars). In many respects, (the mathematical model of) contextual grammars and (the linguistic model of) dependency grammars ((Mel'čuk, 1987) and others) have the same roots. The similitude between contextual grammars and dependency syntactic models based on dependency trees go further, since both formalisms deal mostly with symbols of the language and less or at all with auxiliary symbols as in the case of formalisms based on constituents trees. A similar argument to this one is presented in (Mráz et al., 2000), where contextual mechanism is described as dual to the *analysis by reduction* linguistic method.

However, no relationship was established yet between contextual and dependency grammars. This happened probably because, originally, contextual grammars developed a string generative mechanism and structures on strings generated by contextual grammars were introduced only recently in (Martín-Vide & Păun, 1998).

In this paper, we propose a new approach to the generation of dependency trees (D-trees), based on contextual grammars. The notion of internal *dependency contextual grammar* (DCG) is introduced, by adding dependency descriptions to contextual rules from ordinary contextual grammars. In a DCG, the derivation relation is constrained in two ways: contexts are selected by (strings of) words, but also by the correct construction of the D-tree. Simplifying, string insertion rules stand for the word order, while dependencies express mainly the dominance relationship between words, as in unordered syntactic D-trees. Local and long distance dependencies are treated in an uniform way, using the smooth contextual mechanism.

Several formal examples are given through out the paper, in order to emphasize the capability of DCGs to describe different aspects of language generation. Concerning the strong generative power, we illustrate in Section 4 the flexibility of DCGs in describing even opposite (top-down and bottom-up) derivation styles, as well as different kinds of dependencies (nested or cross-serial), in a very simple and pragmatic way. In Section 5, we prove that the weak generative power of projective internal DCGs (DCGs that generates only projective D-trees) is beyond the generative power of internal contextual grammars with choice.

2 Marcus Contextual Grammars

Contextual grammars are based on the interplay between (strings of) words in a sentence. They start from the assumption that there is a finite set of (simple) sentences (axioms), which are accepted as well-formed without any doubt. More complex sentences are generated by the insertion of other words, in the form of contexts (pairs of strings, possibly one-sided) selected by the strings already present in the sentence.

Formally, an *internal contextual grammar* (CG) is a construct $G = (V, A, C, \phi)$, where V is an alphabet, A is a finite language over V (the set of axioms), C is a finite subset of $V^* \times V^*$ (the set of contexts) and $\phi : V^* \to \mathcal{P}(C)$ is the choice (or selection) recursive map. When $\phi(x) = C$, for all $x \in V^*$, we say that G is a CG *without choice* and write $G = (V, A, C)$. The

internal derivation style (introduced in (Păun & Nguyen, 1980)) of a CG is defined by:

$$x \Rightarrow y \quad \text{iff} \quad x = x_1x_2x_3 \text{ and } y = x_1ux_2vx_3 \text{ for some } (u, v) \in \phi(x_2).$$

If $\overset{*}{\Rightarrow}$ is the reflexive and transitive closure of $\Rightarrow$, then $L(G) = \{x \in V^* \mid \exists w \in A, w \overset{*}{\Rightarrow} x\}$ denotes the language generated by G. We denote by IC and ICC the classes of languages generated by internal CGs without, respectively with choice[1].

Example 2.1 Consider a very simple internal CG with choice $G_1 = (V, A, C, \phi)$, where

$$\begin{aligned} V &= \{\text{John, likes, Lyn, really}\}; \\ A &= \{\text{John likes Lyn}\}; \\ C &= \{(\text{really}, \lambda)\}; \\ \phi(x) &= \begin{cases} \{(\text{really}, \lambda)\}, & \text{if } x = \text{likes}, \\ \emptyset, & \text{otherwise}; \end{cases} \end{aligned}$$

The language generated by G_1 is: John (really)* likes Lyn.

Contextual grammars. even without choice, are able to generate interesting languages, which are spread into (not covering) the traditional Chomsky hierarchy of languages. We denote by FIN, REG, CF, CS the classes of finite, regular, context-free, respectively context-sensitive languages.

Example 2.2 Consider a CG without choice $G_2 = (\{a, b\}, \{\lambda\}, \{(a, b)\})$. We have

$$L(G_2) = \{w \mid w \in \{a, b\}^*, |w|_a = |w|_b \text{ and } |x|_a \geq |x|_b, \forall x \leq w\}$$

where $|w|_a$ denotes the number of occurrences of a in w and $x \leq w$ denotes that $w = xy$ (x is a *prefix* of w). Then, $L(G_2) \in CF \setminus REG$. The language $L(G_2)$ is known as the *Dyck language* over $\{a, b\}$.

Example 2.3 Consider a CG without choice $G_3 = (\{a, b, c\}, \{\lambda\}, \{(xy, z), (x, yz) \mid \{x, y, z\} = \{a, b, c\}\})$. We have

$$L(G_3) = \{w \mid w \in \{a, b, c\}^*, |w|_a = |w|_b = |w|_c\}.$$

Then, $L(G_3) \in CS \setminus CF$. The language $L(G_3)$ is known as the *Bach language* over $\{a, b, c\}$.

More expressive languages can be generated using the choice function together with a condition on the type of the languages selecting contexts. A CG in the *modular presentation* $G = (V, A, (S_1, C_1), \ldots, (S_n, C_n))$ is a CG $G = (V, A, C, \phi)$ such that $C_i \subseteq \phi(x)$, for all $x \in S_i$, $1 \leq i \leq n$. A pair (S_i, C_i) is called a *contextual production*. We say that a CG has an F choice, where F denotes a family of languages, if $S_i \in F$, for all $1 \leq i \leq n$. We denote by $ICC(F)$ the class of languages generated by internal CGs with F-choice. For example, $ICC(REG)$ denotes the languages generated by internal CGs with regular selection languages.

[1]For details on contextual grammars, the reader is referred to the monograph (Păun, 1997).

Example 2.4 Consider $G_4 = (\{a, b, c, d\}, \{abcd\}, (ab^+c, \{(a, c)\}), (bc^+d, \{b, d)\}))$ a CG in the modular presentation. We have

$$L(G_4) \quad = \quad \{a^n b^m c^n d^m \mid n, m \geq 1\}.$$

Then, $L(G_4) \in CS \setminus CF$. $L(G_4)$ is the schematic representation of linguistic cross dependencies.

Though, there are either simple context-free or linguistic relevant languages that cannot be generated by any (internal) CG, even with choice.

Indeed, the language $L_1 = \{a^n \mid n \geq 1\} \cup \{a^n b^n \mid n \geq 1\}$ cannot be generated with (internal) CG. The problem comes from the fact that, in order to generate all the sentences of the form a^n, $n \geq 1$ one needs to define a context (u, v) containing only the letter a (at least one occurrence) and selected by a set of strings also made only by as. Then, there is no mechanism to avoid the insertion of the context (u, v) into sentences of the form $a^n b^n$ (for an appropriate n) and producing in this way sentences of the form $a^{n+i} b^n$, with $i > 0$, which do not belong to the language L_1 (details of the proof are in (Păun, 1997), page 58). Thus, $L_1 \in CF \setminus ICC$.

A similar problem occurs with the language $L_2 = \{a^n b^n c^n \mid n \geq 0\}$, a non-context-free language, which schematically represents the multiple (triple) agreement that can be found also in some linguistic constructions. In (Păun, 1997), page 46, it is proved that $L_2 \notin ICC$ using a pumping lemma, which holds for internal CG. Constructively speaking, the problem comes again from the fact that once one considers a context of the form (a, bc) to generate only strings with an equal number of as, bs and cs, this context cannot be applied anywhere, but only between as and bs (the left side of the context), respectively between bs and cs (its right side). The only way to do this would be to associate with the context a selector of the form b^n, $n \geq 1$, but this is not enough, because it would be possible to apply the context also to a shorter string of bs than intended and generate strings of the form $a^n b^m a b^p b c b^r c^n$, with $m + p + r = n$, which do not belong to L_2.

In (Martín-Vide et al., 1997) (also (Marcus et al., 1998)), it is shown that languages as L_1 or L_2 can be generated by internal CGs with a condition of maximality on the length of the substrings of the given string, which are used to select contexts at some derivation step. In Section 5, we provide with a different method for the improvement of the (weak) generative power of CGs.

3 Dependency Contextual Grammars

A dependency tree (D-tree) is a tree whose nodes are labelled over an alphabet of terminal symbols. Sometimes, its edges are also labelled over another alphabet (of syntactic categories), but we will not use this feature here as it is not relevant for the purpose of this paper. We will introduce D-trees using the concept of a structured string from (Marcus, 1967) (see also (Martín-Vide & Păun, 1998), (Marcus et al., 1998)).

Let V be an alphabet. If n is a natural number, we denote by $[n]$ the set of the first n natural numbers. A *structured string* over V is a pair (x, ρ_x), where $x \in V^+$ is a non-empty string over V and $\rho_x \subseteq [|x|] \times [|x|] \setminus \{(i, i) \mid i \in [|x|]\}$ is an anti-reflexive binary relation, called *dependency relation* on x. If x is a string and $i \in [|x|]$, we denote by $x(i)$ the i-th symbol of x. If $i \rho_x j$, then we say that $x(j)$ depends on $x(i)$. Let us denote by ρ_x^+ (and call *dominance relation* on x) the transitive closure of ρ_x. If $i \rho_x^+ j$, then we say that $x(i)$ dominates $x(j)$.

A structured string $t = (x, \rho_x)$ is called a *D-tree* iff the dependency relation ρ_x induces a structure of tree over x, i.e. i) there is $1 \leq r \leq [|x|]$ such that $x(r)$ does not depend on any symbol of x (r is called the root of t); ii) for any $i \in [|x|] \setminus \{r\}$, there is a unique index $j \in [|x|]$ such that $x(i)$ depends on $x(j)$; iii) ρ_x^+ is an anti-reflexive relation, i.e. $(i, i) \notin \rho_x^+$, for any $i \in [|x|]$. We denote by $\Delta(V)$ the set of D-trees over a set V of terminals.

Linguistically motivated, the structures over CGs were introduced in (Martín-Vide & Păun, 1998) (see al(Marcus et al., 1998), or Section 7.6 in (Păun, 1997)). Bracketed CGs, i.e. contextual grammars working on strings of terminal symbols enriched with a well-formed structure of brackets, were extensively studied in (Kappes, 2000). We are interested here in the second type of structured CGs introduced in (Martín-Vide & Păun, 1998), which do not use brackets for creating tree-like structures on strings, but binary relations between terminal symbols. However, the definition of a dependency CG given below is formally different from the definition of a structured CG given in (Martín-Vide & Păun, 1998).

We call $G = (V, A, P_1, \ldots, P_n)$ an *internal dependency contextual grammar* (IDCG) iff V is an alphabet, $A \subseteq \Delta(V)$ is a finite set of D-trees over V (the axioms) and for any $i \in [n]$, $P_i = (S_i, c_i, d_i)$ is a dependency contextual production, with $S_i \subseteq V^*$ (the set of selectors), $c_i = (u_i, v_i) \in V^* \times V^*$ (the context) and $d_i \subseteq [([|u_i v_i|] \cup V) \times ([|u_i v_i|] \cup V)] \setminus V \times V$ (the set of new dependencies). A derivation in an IDCG is defined as a binary relation over the set of D-trees over V by:

$(x, \rho_x) \Rightarrow_d (y, \rho_y)$ iff $x = x_1 x_2 x_3, y = x_1 u x_2 v x_3$ and there is $i \in [n]$, with $x_2 \in S_i$, $c_i = (u, v)$ and ρ_y built under the following rules:
- ρ_y contains all the dependencies in ρ_x and no other dependencies occur between the symbols originating in x,
- ρ_y contains all the dependencies between the symbols of c_i, described in d_i, and no other dependencies occur between the symbols originating in c_i,
- ρ_y may contain dependencies between the symbols of x and the new symbols introduced by c_i as described by d_i i.e. if a dependency occurs between $u_i v_i(j)$ and a symbol a from x, then $(j, a) \in d_i$ (respectively $(a, j) \in d_i$).

The difference between the above definition of a IDCG and the definition of a structured CG from (Martín-Vide & Păun, 1998) consists in the fact that the new symbols inserted by some dependency contextual rule do not attach to some specified (localized) selector symbols, but to some selector symbols having the *value* specified in the set of new dependencies.

If $\stackrel{*}{\Rightarrow}_d$ is the reflexive and transitive closure of $\Rightarrow_d$, then $DL(G) = \{t \in \Delta(V) \mid \exists s \in A, s \stackrel{*}{\Rightarrow}_d t\}$ denotes the *dendrolanguage* (the set of D-trees) generated by G. Then, $L(G) = \{x \in V^+ \mid \exists (x, \rho_x) \in DL(G)\}$ denotes the language generated by G. We denote by $IDCC$ the class of languages generated by IDCGs.

Example 3.1 We reconsider Example 2.1, from the previous section, now introducing dependencies on words. We define an internal DCG $G_1' = (V, A, P)$, where

$$\begin{aligned} V &= \{\text{John, likes, Lyn, really}\}; \\ A &= \{(\text{John likes Lyn}, \{(2,1),(2,3)\})\}; \end{aligned}$$

$$P \;=\; (\{\text{likes}\}, (\text{really}, \lambda), \{(\text{likes}, 1)\}\}).$$

One of the D-trees generated by G_1' is:

$$(\text{John really likes Lyn}, \{(3,1),(3,4),(3,2)\})$$

(see Figure 1), which underlies the string: John really likes Lyn.

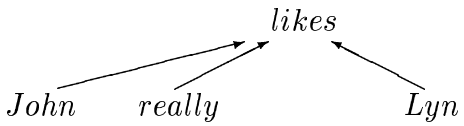

Figure 1: A D-tree for *John really likes Lyn*

Projectivity is a very common property of D-trees, which expresses the fact that the vertical projections of the nodes do not intersect the edges of the tree. We will consider in the following a particular case of DCG, which generates only projective D-trees.

Let x be a string. A sequence $x(i)\dots x(j)$, $1 \leq i \leq j \leq |x|$, of consecutive symbols of x is called an *interval* of x. Let (x, ρ_x) be a D-tree. The *maximal projection* of a symbol $x(i)$, $i \in [|x|]$, of x, is the sequence $x(i_1)\dots x(i_n)$, $n \in [|x|]$, of (not necessarily consecutive) symbols of x, such that $i_j < i_k$, for any $1 \leq j < k \leq n$, and $\{i_1, \dots, i_n\} = \{i\} \cup \{j \mid i\rho^+ j\}$.

Then, a D-tree is *projective* iff all the maximal projections of its symbols are intervals (see (Dikovsky & Modina, 2000)). We denote by $P\Delta(V)$ the set of projective D-trees over an alphabet V.

An IDCG $G = (V, A, P_1, \dots, P_n)$ is called a *projective internal dependency contextual grammar* (PIDCG) iff the axioms are projective D-trees, $A \subseteq P\Delta(V)$, and the derivation relation, $\Rightarrow_{pd}$ is defined as the restriction of $\Rightarrow_d$ on the set $P\Delta(V)$ of projective D-trees. Correspondingly, the dendrolanguage generated by a PIDCG is defined exactly as in the case of an IDCG, but using the derivation relation $\Rightarrow_{pd}$. We denote by $PIDCC$ the class of languages generated by PIDCGs.

The IDCG in Example 3.1 is a PIDCG.

4 Strong Generative Properties

After introducing structures on strings generated by CGs, one may speak about the strong generative power of the structured CGs. The strong generative power means the capacity of (dependency, in our case) CGs to generate different dendrolanguages, underlying the same language. In this section, we investigate some structural properties of IDCGs.

Top-down and bottom-up derivation

The derivation of phrases in a context-free grammar is a top-down process starting with the root of the (constituent) tree and leading to its leaves. Among the novelties brought by tree-adjoining

grammars (Joshi, 1987) and other formalisms in the area of the mild context sensitiveness, was the fact that derivation is not necessarily a top-down process. It is important to remark the capacity of IDCGs to construct the (same) structure of the (same) string in totally different ways.

Example 4.1 Consider $G_5 = (\{a, b\}, \{(ab, \{(1, 2)\})\}, (\{\lambda\}, (a, b), \{(a, 1), (1, 2)\}))$ and $G_6 = (\{a, b\}, \{(ab, \{(1, 2)\})\}, (a^+b^+, (a, b), \{(1, a), (1, 2)\}))$ two IDCGs.

The D-tree represented in Figure 2 corresponding to the string a^3b^3 is generated in a top-down manner (at any step, the axiom ab rests in the top, while the new context (a, b) is added at the bottom of the tree) by G_5 and in a bottom-up manner (the new context is added always in the top) by G_6.

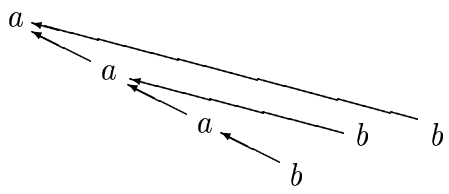

Figure 2: D-tree of the string a^3b^3 generated by G_5 and G_6

Nested and cross-serial dependencies

In Example 4.1, we defined two IDCGs, which are rather ambiguous with respect to the structures associated with well-formed strings. For example, in both grammars we can associate several D-trees with the string a^3b^3, and not only the D-tree in Figure 2. The D-tree from Figure 2 is a projective D-tree, which represents a set of nested dependencies between the symbols a and b that were inserted in the string at the same derivation step. Indeed, one may notice that the last b depends on the first a, the second b depends on the second a, while the first b depends on the last a. Another possible structure of a^3b^3, generated by the grammar G_6, is the non-projective D-tree in Figure 3, which encounters mixed dependencies.

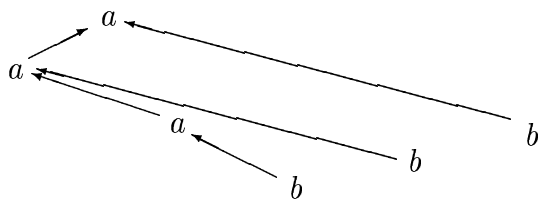

Figure 3: D-tree of the phrase a^3b^3 generated by G_6

In practice, one may want to control the distribution of dependencies over the string in a very rigorous manner. The following example illustrates the capacity of PIDCGs to generate D-trees with specific dependencies.

Example 4.2 First, we modify the grammar G_6 from Example 4.1 in order to obtain only D-trees representing nested dependencies. Actually, we may keep the same grammar and consider the projective derivation $\Rightarrow_{pd}$. G_6 becomes a PIDCG (since the axiom is a projective D-tree). Then the D-tree represented in Figure 2 is the only D-tree associated with a^3b^3 and generated by G_6 in the projective derivation style. Also the language generated by G_6, in this case, is exactly $\{a^nb^n \mid n \geq 1\}$.

The second grammar generates the same language, $\{a^nb^n \mid n \geq 1\}$, using a totally different method, which hides cross-serial dependencies between the symbols a and b introduced at the same derivation step. Consider $G_7 = \{a, b\}, \{(ab, \{(1, 2)\})\}, (\{a^+\}, (a, b), \{(1, a), (b, 2)\}))$ a PIDCG. The (only) D-tree associated with the string a^3b^3 is represented in Figure 4.

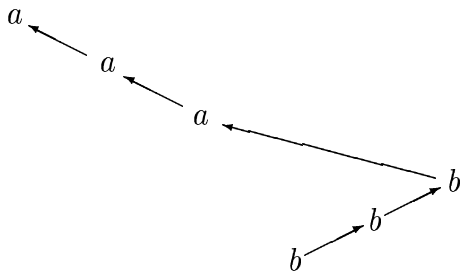

Figure 4: D-tree of the string a^3b^3 generated by G_7

5 Weak Generative Properties

In this section, we will prove that using PIDCGs also the weak generative capacity of CGs is improved.

Theorem 5.1 *For any internal contextual grammar G, there exists a weak equivalent internal projective dependency contextual grammar G'.*

Proof Let $G = (V, A, (S_1, C_1), \ldots, (S_n, C_n))$ be an ICG. We may consider that each contextual production corresponds to one context. We construct the following PIDCG $G' = (V, A', (S_1, C_1, d_1), \ldots, (S_n, C_n, d_n))$, where

$$\begin{aligned} A' &= \{(x, \rho_x) \mid x \in A, \rho_x = \{(i, i+1) \mid i \in [|x| - 1]\}\} \\ d_i &= \{(a, j) \mid j \in [|u_iv_i|], a \in V\}, \end{aligned}$$

for $c_i = (u_i, v_i)$ and for all $i \in [n]$. Then, $L(G') = L(G)$. □

From the above proof, it is worth to note that on the contextual side nothing changes when adding dependencies: if G is without choice then G' is also without choice; if G has an F-choice then G' has also an F-choice.

Now consider the languages discussed in the end of Section 2, $L_1 = \{a^n \mid n \geq 1\} \cup \{a^nb^n \mid n \geq 1\}$ and $L_2 = \{a^nb^nc^n \mid n \geq 0\}$. These languages cannot be generated by any ICG, but they can be generated by PIDCGs.

Example 5.1 In order to obtain the first language, let us consider the following PIDCG $G_8 = (\{a, b\}, \{(a, \emptyset), (ab, \{(1, 2)\})\}, (\{a\}, (a, \lambda), \{(1, a)\}), (\{ab\}, (a, b), \{(b, 1), (b, 2)\}))$. We have $L(G_8) = L_1$. Figure 5 illustrates the unique D-tree and one of the possible D-trees generated by G_8 and associated with the strings a^3, respectively a^3b^3.

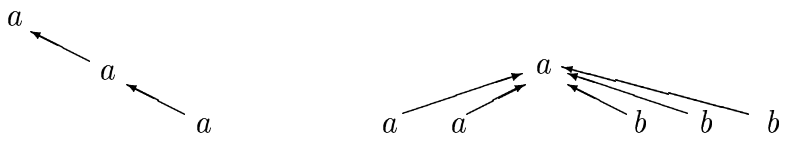

Figure 5: D-trees of a^3, respectively a^3b^3 generated by G_8

Example 5.2 Let $G_9 = (\{a, b, c\}, \{(abc, \{(2, 1), (2, 3)\}, (\{b^+, (a, bc), \{(a, 1), (b, 2), (c, 3)\}))$ be another PIDCG. We obtain the second language, $L(G_9) = L_2$. Figure 6 illustrates one of the possible D-trees generated by G_9 and associated with the string $a^3b^3c^3$.

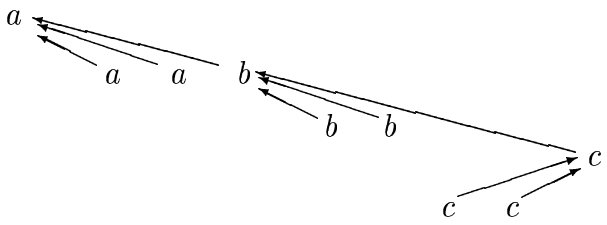

Figure 6: D-tree of the string $a^3b^3c^3$ generated by G_9

To conclude this section, we establish the following result.

Theorem 5.2 *The strict inclusion $ICC \subset PDICC$ holds.*

6 Conclusions

We introduced a formal model for the generation of dependency trees, dealing with free word order phenomena. The problem is currently under study (see (Holan et al., 1998), (Dikovsky, 2001), (Gramatovici & Plátek, 2003)) and is far to be solved.

The model of dependency contextual grammars we propose here is based on two main characteristics: the intrinsic similitude between dependency trees and contextual grammars and the flexibility of the latter in expressing non-local dependencies. Our approach is closer to the intrinsic motivations of original dependency grammars than other current similar attempts are.

The research of dependency contextual grammars just started since the model has to be further tested on both mathematical and linguistical sides.

References

Dikovsky A. (2001). Grammars for Local and Long Dependencies. *Proceedings of the 39th Annual Meeting of the ACL*.

Dikovsky A. & Modina L. (2000). Dependencies on the Other Side of the Curtain. In *Traitement Automatique des Langues (TAL)*, **41 (1)**, 79-111.

Gramatovici R. & Plátek M. (2003). On D-trivial Dependency Grammars. *Submitted*.

Holan T., Kuboň V., Oliva K. & Plátek M. (1998). Two Useful Measures of Word Order Complexity. In *Proceedings of the Coling'98 Workshop "Processing of Dependency-Based Grammars"*, Polguere A. & Kahane S. eds., University of Montreal, Montreal, 21-28.

Joshi A. K. (1987). An Introduction to Tree Adjoining Grammars. In *Mathematics of Languages*, Manaster-Ramer A. ed., Amsterdam, Philadelphia: Johm Bejamins, 87-114.

Kappes M. (2000). *Bracketed Contextual Grammars*. Doctoral Dissertation, Johann Wolfgang Goethe Universität, Frankfurt am Main.

Marcus S. (1967). *Algebraic Linguistics. Analytical Models*. New-York, London: Academic Press.

Marcus S. (1969). Contextual Grammars, *Rev. Roum. Math. Pures Appl.*. **14 (10)**, 69-74.

Marcus S. (1997). Contextual Grammars and Natural Languages. In *The Handbook of Formal Languages*, Rozenberg G. & Salomaa A. eds., Berlin, Heidelberg, New-York: Springer-Verlag, vol. 2, 215-235.

Marcus S., Martín-Vide C. & Păun Gh. (1998). Contextual Grammars as Generative Models of Natural Languages. *Computational Linguistics*, **24 (2)**, 245-274.

Martín-Vide C. & Păun Gh. (1998). Structured Contextual Grammars. *Grammars*, **1 (1)**, 33-55.

Martín-Vide C., Mateescu A., Miquel-Verges J. & Păun Gh. (1997). Internal contextual grammars: minimal, maximal and scattered use of selectors. In *Proceedings of The Fourth Bar-Ilan Symposium on Foundations of Artificial Intelligence. Focusing on Natural Languages and Artificial Intelligence - Philosophical and Computational Aspects*, Koppel M. & Shamir E. eds., Menlo Park: AAAI Press, 159-168.

Mel'čuk I. (1987). *Dependency Syntax. Theory and Practice*, Albany: State University of New-York Press.

Mráz F., Plátek M. & Procházcha M. (2000). Restarting automata, deleting and Marcus grammars. In *Recent Topics in Mathematical and Computational Linguistics*, Martin-Vide C. & Păun Gh. eds., Bucharest: Romanian Academy Publishing House, 218-233.

Păun Gh. (1997). *Marcus Contextual Grammars*, Dordrecht, Boston, London: Kluwer.

Păun Gh. & Nguyen X. M. (1980). On the inner contextual grammars, *Rev. Roum. Math. Pures Appl.*, **25**, 641-651.