Some Notes on Generative Capacity of Dependency Grammars

Tomasz Obrębski [*] [†]
[*] Institute of Control
and Information Engineering
Electric Faculty
Poznań University of Technology
Poznań, Poland
Tomasz.Obrebski@put.poznan.pl

Filip Graliński [†]
[†] Department of Computational Linguistics
and Artificial Intelligence,
Faculty of Mathematics
and Computer Science,
Adam Mickiewicz University
Poznań, Poland,
filipg@amu.edu.pl

Abstract

The aim of this paper is to gather and add to the dispersed knowledge concerning the relation between dependency grammars and context-free languages and some of its subclasses. Necessary and sufficient conditions for a dependency system to have context-free power are formulated in a way general enough to be applicable to a broad range of grammatical systems, based on rules or constraints. Certain cases when some of these requirements are not satisfied are also analysed. Formal implications of the presence of dependency types in grammatical systems are discussed.

1 Introduction

The notion of dependency grammar is somewhat vague. Formal systems introduced by Hays (1964) and Gaifman (1965), Maruyama (1990b), Covington (1990), Carroll and Charniak (1992), Duchier and Debusmann (2001), Nivre (2003), Sleator and Temperley (1991)[1] are all called dependency grammars[2], although they are significantly different and have different formal properties.

A number of results concerning descriptive capacity of some types of dependency systems have been reported in the literature, mostly for Gaifman-style grammars (Gaifman, 1965; Nivre, 2002) and Maruyama's grammars (Maruyama, 1990a; White, 1995). Other kinds of systems, in particular the constraint-based systems other than Maruyama's one, have not been analysed in this respect.

The goal of this paper is to gather and add to the dispersed knowledge concerning the relation between dependency grammars and context-free languages and some of its subclasses. Many of the facts and observations contained in this paper may well be known to a number of researchers in the field, though have been only fragmentarily articulated in the literature.

Taking into consideration various types of grammatical systems we will attempt to formulate necessary and sufficient conditions for a dependency grammar to have at least CF generative capacity. The conditions will be stated in a way general enough to be applicable to a broad range of grammatical systems. We will enumerate the concepts which must be expressible in a grammatical system to ensure the CF capacity. Also, some cases when not all of the requirements are satisfied will be analysed.

We will adopt the following general definition of dependency grammar which expresses, at least approximately, the essential feature shared by the above-mentioned systems: a *dependency grammar* is a formal system which, given two finite sets of symbols, defines a correspondence between sequences of symbols from the first set and trees whose nodes are labeled with symbols from the second set; tree nodes must be related one-to-one to sequence elements.

Symbols from the first set are typically called terminal symbols or word forms, while the elements of the second set are referred to as syntactic or lexical categories. Tree arcs may or may not be labeled with dependency types.

Only projective systems with atomic category symbols are considered in this paper. Some of the systems mentioned above go beyond our general definition together with these restrictions (those of Maruyama's, Duchier and Debusmann's, and Sleator and Temperley's). We are not interested, however, in examining the properties of these concrete systems as such, but only in the various ways the rules/constraints describing legal tree structures are formulated in different frameworks.

[1]The list is meant to show the variety of formal systems covered by the term 'dependency grammar', and is not exhaustive in any sense.

[2]Sleator and Temperley's formalism is called *Link* Grammar but it is usually classified as a dependency system.

In this respect the dependency systems may be divided into two broad classes. Into the first class fall systems which simply enumerate all possible branchings which may occur in a tree (Gaifman, 1965; Lombardo and Lesmo, 1998; Carroll and Charniak, 1992). We will call them *rule-based* systems. Into the second class – *constraint-based* systems – fall formalisms where the well-formedness conditions on acceptable tree structures are formulated by means of constraints stating e.g. what arcs are allowed or necessary in given contexts (Maruyama, 1990b; Covington, 1990; Duchier and Debusmann, 2001; Nivre, 2003).

As a special kind of rule-based system we can classify Link Grammar with the word descriptions considered in disjunctive form (Sleator and Temperley, 1991, 8–9), see also remarks concerning Link Grammar in Section 6.

2 Minimal constraint-based formulation

Our aim is to examine which conditions are necessary and sufficient for a dependency formalism to be capable of generating all CF languages. First, we will introduce a simple constraint-based formalism which has CF generative capacity and which is minimal in the sense that it cannot be simplified without affecting its descriptive power. This simple system, in our opinion, provides good intuitions as to what has to be expressible in a formal grammatical system to obtain CF power. It should be made clear, however, that the proposed formalism is unlikely to be of any direct practical value.

A *Minimal Constraint-based Dependency Grammar* (MCDG) is defined as a 6-tuple $(\Sigma, C, C_{root}, \mathcal{L}, \mathcal{O}_L, \mathcal{O}_R)$, where:

- Σ – terminal alphabet,
- C – set of categories,
- $C_{root} \subset C$ – set of root categories,
- $\mathcal{L} \subset \Sigma \times C$ – one-to-many relation (lexicon),
- $\mathcal{O}_L : C \to 2^{C \cup \{\varepsilon\}}$ – for each category $A \in C$, $\mathcal{O}_L(A)$ is the set of categories allowed for the left dependent of a node labeled with A; a node of category A has at most one left dependent, which is not obligatory iff the special symbol ε belongs to $\mathcal{O}_L(A)$,
- $\mathcal{O}_R : C \to 2^{C \cup \{\varepsilon\}}$ – defined analogously as $\mathcal{O}_L$, but for the right dependent.

(As it is evident from the above definition, MCDGs license only binary dependency trees.)

For example, the following grammar describes the language $a^n b^n$: $\Sigma = \{a, b\}$, $C = \{A, B\}$, $C_{root} = \{A\}$, $\mathcal{L} = \{(a, A), (b, B)\}$, $\mathcal{O}_L(A) = \mathcal{O}_R(B) = \{\varepsilon\}$, $\mathcal{O}_R(A) = \{B\}$, $\mathcal{O}_L(B) = \{\varepsilon, A\}$.

We are going to prove that MCDGs are weakly equivalent to CFGs. Our approach here is to some extent similar to the method used by Maruyama to prove the context-free power of his constraint dependency grammars (Maruyama, 1990a; White, 1995).

First, let us give some auxiliary definitions: a dependency tree is defined recursively as

$$(D_1, \ldots, D_l, X, D_{l+1}, \ldots, D_{l+r}),$$

where $l, r \geq 0$, $X \in C$ is the root of the tree, D_i are dependency (sub)trees; the root of a tree D will be denoted by $r(D)$, i.e.

$$r((D_1, \ldots, D_l, X, D_{l+1}, \ldots, D_{l+r})) = X,$$

whereas $t(D)$ is the sequence of categories in D:

$$t((D_1, \ldots, D_l, X, D_{l+1}, \ldots, D_{l+r})) = \\ t(D_1) \ldots t(D_l) X t(D_{l+1}) \ldots t(D_{l+r})$$

(In MCDGs $l, r \in \{0, 1\}$, as only binary trees can be generated.)

If D is a dependency tree and G a constraint-based dependency grammar, then $G \vdash D$ is to stand for: "D fulfils requirements expressed by G" (i.e. by $\mathcal{O}_L$ and $\mathcal{O}_R$). The derivation relation holds as follows: $X \Rightarrow^*_G \gamma$ iff $\exists D : G \vdash D \wedge r(D) = X \wedge t(D) = \gamma$, where $X \in C$, $\gamma \in C^*$, D is a dependency tree. Now we define the derivation relation for sequences of words: $X \Rightarrow^*_G w_1 w_2 \ldots w_n$ iff $X \Rightarrow^*_G Y_1 Y_2 \ldots Y_n$ and $(\forall i \leq n)(w_i, Y_i) \in \mathcal{L}$.

Given a dependency grammar G, $L(G)$ is the language generated by G:

$$L(G) = \{\, \alpha \in \Sigma^* \mid X \Rightarrow^*_G \alpha \wedge X \in C_{root} \,\}.$$

Proposition 1 *For every context-free grammar G which does not generate the null string there exists a minimal constraint-based dependency grammar G' such that $L(G') = L(G)$.*

Proof. We may assume that $G = (V, \Sigma, P, S)$ is given in Greibach normal form, i.e. all production rules in $P = \{P_1, \ldots, P_m\}$ are of the form:

$$A \to a B_1 B_2 \ldots B_n,$$

where $A, B_1, B_2, \ldots, B_n \in V, a \in \Sigma, n \geq 0$.

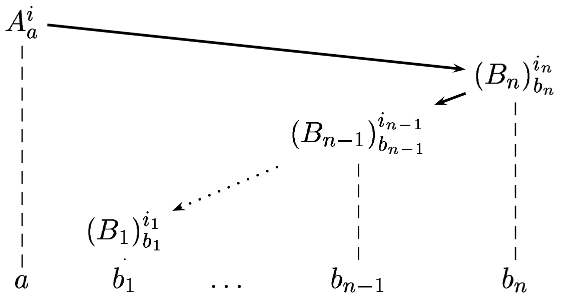

Figure 1: A fragment of dependency tree corresponding to a CFG rule

Let us index all the occurrences of non-terminal symbols in the right-hand sides of the rules: a production rule will be written as

$$A \to aB_1^{i_1}B_2^{i_2}\dots B_n^{i_n},$$

where $B_1, B_2, \dots, B_n$ are non-terminal symbols and $i_1, i_2, \dots, i_n \neq 0$ are indices of the occurrences (index 0 will have a special meaning). By $I(X)$ we denote the set of the indices of occurrences of X in the right-hand sides.

G', weakly equivalent to G, can be constructed in the following way:

- $C = \{\, X^i_w \mid X \to wB_1^{i_1}B_2^{i_2}\dots B_n^{i_n}, i \in I(X) \cup \{0\}\,\}$,
- $C_{root} = \{\, S^0_w \mid S^0_w \in C\,\}$,
- $\mathcal{L} = \{\,(w, X^i_w) \mid w \in \Sigma, X^i_w \in C\,\}$,
- $\mathcal{O}_L(X^0_w) = \{\varepsilon\}$,
- $\mathcal{O}_L(X^i_w) = \{\varepsilon\}$ if $i \neq 0$ and $A \to uX^iB_2^{i_2}\dots B_n^{i_n}$,
- $\mathcal{O}_L(X^i_w) = \{\, Y^j_u \mid Y^j_u \in C\,\}$ if $i \neq 0$ and $A \to vB_1^{i_1}\dots Y^{i_{q-1}}X^{i_q}\dots B_n^{i_n}$, $i = i_q$, $j = i_{q-1}$ for some $q \in \{2, \dots, n\}$ (the left dependent corresponds to the left-adjacent symbol in the right-hand side of a CFG rule),
- $\mathcal{O}_R(X^i_w) = \bigcup_{P_k = X \to w\dots} l(P_k)$, where $l(P_k) = \{\varepsilon\}$ if $P_k = X \to w$, or $l(P_k) = \{\, Y^j_u \mid Y^j_u \in C\,\}$ if $P_k = X \to wB_1^{i_1}B_2^{i_2}\dots Y^{i_n}$, $j = i_n$ (the right dependent corresponds to the last symbol in a CFG rule, in which the governor is the symbol in the left-hand side of the rule).

In other words, the production $A \to aB_1^{i_1}B_2^{i_2}\dots B_n^{i_n}$ will be represented in a dependency tree by a fragment shown in Figure 1.

Example: let us convert the following CFG:

- $V = \{S, A, B\}$,
- $\Sigma = \{x, y, z\}$,
- $P = \{S \to xAA, A \to y, A \to zB, B \to zB, B \to z\}$.

After adding occurrence indices ($P = \{S \to xA^1A^2, A \to y, A \to zB^1, B \to zB^2, B \to z\}$) we can construct an equivalent MCDG:

- $C = \{S^0_x, A^0_y, A^0_z, B^0_z, A^1_y, A^2_y, A^1_z, A^2_z, B^1_z, B^2_z\}$,
- $C_{root} = \{S^0_x\}$,
- $\mathcal{L} = \{(x, S^0_x), (y, A^0_y), (z, A^0_z), (z, B^0_z), (y, A^1_y), (y, A^2_y), (z, A^1_z), (z, A^2_z), (z, B^1_z), (z, B^2_z)\}$,
- $\mathcal{O}_L(S^0_x) = \mathcal{O}_L(A^0_y) = \mathcal{O}_L(A^0_z) = \mathcal{O}_L(B^0_z) = \mathcal{O}_L(A^1_y) = \mathcal{O}_L(A^1_z) = \mathcal{O}_L(B^1_z) = \mathcal{O}_L(B^2_z) = \{\varepsilon\}$, $\mathcal{O}_L(A^2_y) = \mathcal{O}_L(A^2_z) = \{A^1_y, A^1_z\}$
- $\mathcal{O}_R(S^0_x) = \{A^2_y, A^2_z\}$, $\mathcal{O}_R(A^0_y) = \mathcal{O}_R(A^1_y) = \mathcal{O}_R(A^2_y) = \{\varepsilon\}$, $\mathcal{O}_R(A^0_z) = \mathcal{O}_R(A^1_z) = \mathcal{O}_R(A^2_z) = \{B^1_z\}$, $\mathcal{O}_R(B^0_z) = \mathcal{O}_R(B^1_z) = \mathcal{O}_R(B^2_z) = \{\varepsilon, B^2_z\}$

Let $G' \vdash_{-L} D$ mean that either left dependent is not obligatory for $r(D)$ (i.e. $\varepsilon \in \mathcal{O}_L(r(D))$) and $G' \vdash D$, or that this dependent is obligatory and a dependency tree D fulfils the requirements of G' with the exception of $r(D)$ not being connected to a left dependent. The relations $X \Rightarrow^*_{-L,G'} \gamma$ and $X \Rightarrow^*_{-L,G'} \alpha$ (where $\gamma \in C^*, \alpha \in \Sigma^*$) are defined analogously to $X \Rightarrow^*_{G'} \gamma$ and $X \Rightarrow^*_{G'} \alpha$ respectively.

Now it is rather straighforward to prove (by induction on the length of α) that

$$X \Rightarrow^*_G \alpha \leftrightarrow (\exists w \in \Sigma)(\forall i \in I(X)) X^i_w \Rightarrow^*_{-L,G'} \alpha.$$

Taking $X = S$, $i = 0$ we get $S \Rightarrow^*_G \alpha$ iff $(\exists w \in \Sigma) S^0_w \Rightarrow^*_{G'} \alpha$, hence $L(G) = L(G')$.

Proposition 2 *For every minimal constraint-based dependency grammar G there exists a CF grammar G' such that $L(G') = L(G)$.*

Proof. Let $G = (\Sigma, C, C_{root}, \mathcal{L}, \mathcal{O}_L, \mathcal{O}_R)$. The context-free grammar $G' = (C \cup \{S\}, \Sigma, P, S)$ is weakly equivalent to G where $S \notin C$ and $P = \{\, S \to AxB \mid X \in C_{root}, (x, X) \in \mathcal{L}, A \in \mathcal{O}_L(X), B \in \mathcal{O}_R(X)\,\} \cup \{\, X \to AxB \mid X \in C, (x, X) \in \mathcal{L}, A \in \mathcal{O}_L(X), B \in \mathcal{O}_R(X)\,\}$.

Corollary 1 *Every context-free grammar which does not generate the null string can be transformed into an equivalent grammar in which all production rules are of the form: $X \to YaZ$, $X \to Ya$, $X \to aY$, $X \to a$.*

3 Minimal requirements for context-free power

Taking into consideration the way the MCDG system is constructed, let us discuss in less formal manner what is needed for a dependency-based grammatical system to have at least context-free descriptive capacity (the components of MCDG implementing the respective concepts are given in parentheses):

(1) lexicon expressing one-to-many (or many-to-many) correspondence between terminal symbols and category symbols ($\mathcal{L}$),

(2) distinguished subset of category symbols allowed for tree roots (C_{root}),

(3) at least two dependents must be allowed (two $\mathcal{O}$'s),

(4) the ability to indicate the relative position of the dependent with respect to its head ($\mathcal{O}_L$ vs. $\mathcal{O}_R$),

(5) the ability to express that a specific dependent (i.e. of category belonging to a certain class) must be present

(6) the ability to express that a specific dependent (see above) cannot be repeated.

In MCDG (5) and (6) are conjointly implemented by $\mathcal{O}_L$ and $\mathcal{O}_R$.

(1) to (6) given above express the concepts of: (1) *lexical ambiguity*, (2) *"rootness"*, (3) *binary branching*, (4) *dependency direction*, (5) *obligatoriness*, (6) *non-repeatability.*

From the equivalence of MCDGs and CFGs follows that the above conditions are sufficient for a dependency system to have (at least) CF power. We will now show that they are also necessary.

Concepts (2), (4), (5) and (6) are needed to generate the simple language $L = \{ab\}$. Without (2) or (5), $ab \in L$ implies either $a \in L$ or $b \in L$. Without (4), $ab \in L$ implies $ba \in L$. Without (6), $ab \in L$ implies either $aab \in L$ or $abb \in L$.

The constraints given in (5) and (6) must refer to sets of categories and not to single categories. An informal argument goes as follows: let us consider the language a^nbc^n, assuming that a is the root, a must have c as its right dependent, c, in turn, **requires either a or b** as its left dependent.

The concepts (1) and (3) will be discussed in the next two sections.

Finally, it can be noted that the following concepts, frequently met in formal descriptions of language, are not needed for CF capacity:

- ability to express any kind of interrelationship among dependents (e.g. their relative order) other than their mutual exclusion,
- repeatable dependencies (in whatever meaning).

4 No lexical ambiguity

Dependency systems without lexical ambiguity, i.e. ones in which the lexicon is degenerated to a one-to-one relation between terminal and category symbols or in which category symbols do not appear at all, come as grammatical components of several stochastic language models. The generative capacity of two such systems – Carroll and Charniak's (1992) Context-Free Dependency Grammar[3] and Eisner's (2000) Bilexical Dependency Grammar[4] – was analysed by Nivre (2002). He shows that the sets of languages these systems generate are not equal and are both proper subsets of context-free languages (and are neither subsets nor supersets of regular languages)[5], which supports the claim (1) from Section 3.

In the following, the grammatical systems with a one-to-one (or many-to-one) lexicon will be called *word-level grammars*, whereas the systems with a one-to-many (or many-to-many) lexicon – *category-level grammars.*

Two comments to Nivre's observations are in order. Firstly, what is clear but not stated explicitly by Nivre, the reason for the weakness of these grammars is the 'lack' of a lexicon – it is exclusively the one-to-one restriction imposed on the lexicon that makes Carroll and Charniak's system different from Gaifman's one, which has context-free power.

Secondly, the descriptive capacity of word-level grammars (unlike that of category-level ones) is very sensitive to the format in which the rules/constraints are formulated, since transfor-

[3]To each terminal symbol x corresponds one nonterminal symbol noted $\overline{x}$, rules have the CFG-like form $\overline{x} \rightarrow \alpha x \beta$, where α and β are sequences of non-terminals.

[4]No category symbols are introduced; for each terminal symbol, possible sequences of its left and right dependents are specified independently by two regular expressions over terminal symbols.

[5]It is not difficult to show that the class of languages generated by MCDGs with $*$-to-one lexicon is still different from the two (it is actually a proper subset of their common part).

mations from one format (normal form) to another, similar to those which can be done for category-level systems, are not possible, as they usually entail multiplication of category symbols, which is not feasible in word-level systems.

5 No branchings

If we do not allow the tree structure to branch, we obtain the power of regular grammars. This fairly obvious fact can be shown by referring once again to our MCDG.

After removing one $\mathcal{O}$ (e.g. $\mathcal{O}_L$) from the definition of MCDG (with one-to-many lexicon), we get a system equivalent to finite automata over Σ[6] (constructing an automaton from a one-branch MCDG: $Q = C \cup \{q_0\}$, $Q_F = \{c \in C \mid \varepsilon \in \mathcal{O}_R(c)\}$, $c' \in \delta(c, a)$ iff $c' \in \mathcal{O}_R(c) \wedge (a, c') \in \mathcal{L}$, $c' \in \delta(q_0, a)$ iff $c' \in C_{root} \wedge (a, c') \in \mathcal{L}$; in the opposite direction: $C = Q \times Q$, $C_{root} = \{q_0\} \times Q$, $\mathcal{L} = \{\, ((q, q'), a) \mid q, q' \in Q, a \in \Sigma, q' \in \delta(q, a)\,\}$, $\mathcal{O}_R((q_1, q_2)) = \{\, (q_2, q_3) \mid q_3 \in Q\,\}$ and $\varepsilon \in \mathcal{O}_R((q_1, q_2))$ iff $q_2 \in Q_F$).

This corresponds to a Gaifman-style grammar with only $*(X)$, $X(*)$ and $X(*, Y)$ rules (cf. Section 6).

To complete the puzzle, we may finally consider the class of languages defined by word-level regular grammars. This small class is defined by a restricted subclass of finite automata in which all transitions through a given symbol a always go to the same destination state, regardless of the source state (cf. the construction of an automaton from an MCDG) – thus, states may be identified with symbols.

6 Rule-based grammars

We will restrict ourselves here to two remarks concerning possible minimal normal forms for rule-based grammars.

In Gaifman's original formulation (Gaifman, 1965), the rules take the forms: $X(Y_1, \ldots, Y_i, *, Y_{i+1}, \ldots, Y_n)$ (a node of category X may have dependents of categories $Y_1, \ldots, Y_n$ in this order and the position of X is indicated by $*$), $X(*)$ (a node of category X may be a leaf), and $*(X)$ (the root node may be of category X).

For context-free power it is enough to have the rules of the following five types: $*(X)$, $X(*)$, $X(*, Y)$, $X(Y, *)$, $X(Y, *, Z)$, since each MCDG can easily be transformed into a Gaifman grammar with this kind of rules.

[6]We take into consideration only languages which do not contain the empty string.

Another normal form for Gaifman grammars, where at most two dependents are allowed, can be derived straightforwardly from 2-Greibach Normal Form[7], namely: $*(X)$, $X(*)$, $X(*, Y)$, $X(*, Y, Z)$.

Let us note that Gaifman's system in the latter form can be obtained (minor details ignored) from Link Grammar by allowing at most one left $(-)$ connector for each word. The left connector would serve for linking the word to its governor, while right $(+)$ connectors would link the word to its dependents; connector names would play the role equivalent to that of category symbols.

7 Maruyama's grammars

In this section some properties of the grammatical system called Constraint Dependency Grammar (CFG) will be discussed. This system, introduced by Maruyama (1990a), differs in many respects from other dependency systems and, in general, does not fall into the definition of dependency grammar as adopted in Introduction. CDG does not relate a sequence of terminal symbols (words) to a tree, but rather to a set of directed graphs with outdegree limited to 1. These graphs are defined by *role* values – roles can be seen as variables, each terminal has k roles, where k is the parameter of the grammar called *degree*. Each role defines one graph. Role value is a pair composed of a *label* and an index of a terminal symbol (modifiee) and can be viewed as a labeled arc pointing from one terminal to another. The constraints in CDGs are expressed by means of logical formulae over assignments of role values to roles.

We will take into consideration only the subclass of CDGs restricted to systems in which (1) roles determine the dependency relation between terminal symbols and (2) projectivity of this relation is ensured (No-Crossover constraints are imposed (White, 1995, p. 12)).

It was shown (Maruyama, 1990a) that any context-free language can be generated by a CDG with two roles[8]. In the proof constraints containing only one or two variables were used (unary/binary constraints).

[7]A CFG is in 2-Greibach Normal Form if its productions are of the following form: $X \to x$, $X \to xY$, $X \to xYZ$, where x is a terminal symbol and X, Y, Z are non-terminal symbols. Every CFG which does not generate the null string can be transformed into an equivalent grammar in 2-Greibach Normal Form (Hopcroft and Ullman, 1979).

[8]CDGs can generate some non-context-free languages as well.

Now, let us discuss why two variables and two roles are necessary. Two variables are necessary and sufficient to express non-repeatability constraint of the type: $mod(x) = mod(y) \wedge label(x) \in K \wedge label(y) \in K \wedge \Rightarrow pos(x) = pos(y)$, which means that no two distinct terminals can be connected to their modifee through labels both belonging to K, where K is a set of labels.

As was demonstrated in Section 3 the necessary requirements for CF power of a dependency system include the capability of expressing both obligatoriness and non-repeatability. One role makes it possible to express either one or the other type of constraint but not both of them: for non-repeatability the role "arc" must point from the dependent to the head (the head plays the role of the modifee), whereas for obligatoriness the arc must go in the opposite direction (the dependent is the modifee). Hence, in order to express obligatoriness and non-repeatability at least two CDG roles are needed (cf. the use of the roles *governor* and *needs* in (White, 1995) or *head-role* and *body-role* in (Maruyama, 1990a)). Note that an MCDG can be straightforwardly expressed by a CDG with three roles: *governor*, *needs-left*, *needs-right*.

8 Dependency types

We start our discussion of dependency types with some observations regarding the use of the *label* function in Maruyama's grammars. In the examples given in (Maruyama, 1990b), the *label* plays the role similar to that of a dependency type (some of the labels appearing in the examples: DET, $SUBJ$, OBJ, $ROOT$[9]). On the one hand, thus, the system may be perceived as a word-level system with typed dependencies. On the other hand, in the proof showing the context-free power of Maruyama's grammar (Maruyama, 1990a), the *label* function values "store" nonterminal symbols – the category of a *label*'ed node. In this case the *label* function plays in fact the role of the lexicon and the system appears more as a category-level system with untyped dependencies.

These observations concerning the dual potential functionality of *label* may suggest that the function of a lexicon in a grammatical system may be implemented with the use of dependency types and vice versa.

[9]$ROOT$ is a special label used for the root node, as a label is mandatory for all role values.

From this perspective, let us consider Gaifman's dependency system. The transformation between this system and its typed word-level equivalent can be performed according to the scheme shown in Figure 2. In order to retain equivalence, the tree context covered by the word-level rule has to be extended with the arc leading to the head node. The rules in the typed word-level version of Gaifman's system could take the following form: $\langle\sigma, x\rangle(\tau_1, \ldots, \tau_i, *, \tau_{i+1}, \ldots, \tau_n)$, where $\sigma, \tau_1, \ldots, \tau_n$ are dependency types and x – a terminal symbol (x, which is linked to its head by σ, can govern n nodes linked via dependencies of types $\tau_1, \ldots, \tau_n$).

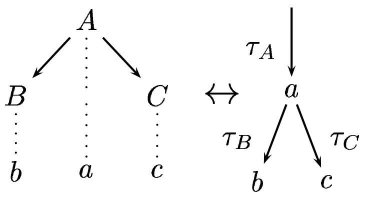

Figure 2: From category symbols to types and *vice versa*

To sum up, both dependency types and categories may be seen as providing terminal symbols with additional information giving the same syntactic potential.

9 Minimal constraint-based formulation with types

Let us consider here a simple dependency formalism with constraints related to dependency types. Such an approach may be regarded as more elegant and is more in the spirit of linguistic descriptions (Mel'čuk, 1988; Hudson, 1990).

Thus, let us define herein a typed version of dependency grammar as a 9-tuple $(\Sigma, C, \mathcal{L}, T, T_{sgl}, T_{obl}, T_{left}, T_{right}, D)$, where Σ, C and $\mathcal{L}$ have the same meaning as in an MCDG, T is a set of dependency types, $T_{sgl}, T_{obl}, T_{left}, T_{right} \subseteq T$, T_{sgl} contains non-repeatable types, T_{obl} – obligatory types, T_{left} – leftward types, and T_{right} – rightward types, and finally $D \subseteq C \times T \times C$ ($(c, \tau, c') \in D$ means that c may govern c' via dependency τ). If $\tau \in T_{obl}$ then dependency τ must occur for each node with category $c \in C$ such that there exists at least one $c' \in C, (c, \tau, c') \in D$. Note that no restriction is imposed on the relative order of dependents.

(This kind of formalism forms the basis of the dependency system for Polish described in (Obrębski, 2003).)

The typed dependency grammars of the form given above have CF power since any MCDG can be easily transformed into such a typed grammar: for each $c \in C$ one leftward type $\tau_{c,l}$ and one rightward type $\tau_{c,r}$ are introduced, now $T_{sgl} = T$, $T_{left} = \{\, \tau_{c,l} \mid c \in C \,\}$, $T_{right} = \{\, \tau_{c,r} \mid c \in C \,\}$, $T_{obl} = \{\, \tau_{c,l} \mid \varepsilon \notin \mathcal{O}_L(c) \,\} \cup \{\, \tau_{c,r} \mid \varepsilon \notin \mathcal{O}_R(c) \,\}$, and $D = \{\, (c, \tau_{c,l}, c') \mid c' \in \mathcal{O}_L(c) \setminus \{\varepsilon\} \,\} \cup \{\, (c, \tau_{c,r}, c') \mid c' \in \mathcal{O}_R(c) \setminus \{\varepsilon\} \,\}$. This form of typed grammar can be seen as its minimal formulation.

For alternative ways of expressing the obligatoriness and non-repeatability (exclusion) constraints based on types or categories, see (Duchier and Debusmann, 2001) and (Bés and Blache, 1999) respectively.

10 Conclusion

The minimal requirements for a dependency grammar formalism to have context-free power have been identified. These are: the capability of expressing lexical ambiguity, "rootness", dependency direction, non-repeatability (exclusivity), obligatoriness (related to types or sets of categories), branching structure.

A formalism called Minimal Constraint-based Dependency Grammar has been introduced. Although this grammatical system is not likely to have any practical value, it could be useful for determining the descriptive power of simple dependency systems of various kinds, since transformations between MCDGs and such different systems as Maruyama's and Gaifman's are straightforward.

Certain functional interchangeability of categories and dependency types has been shown. Both types and categories may be seen as providing terminal symbols with additional information giving the same syntactic potential. Types does not affect the power in case of category-level systems.

References

G. Bés and P. Blache. 1999. Proprieété et analyse d'un language. In *Proc. of TALN'99*, Cargèse.

Glenn Carroll and Eugene Charniak. 1992. Two experiments on learning probabilistic dependency grammars from corpora. Technical Report CS-92-16, Brown University, Department of Computer Science, March.

Michael A. Covington. 1990. Parsing discontinuous constituents with dependency grammar. *Computational Linguistics*, 16(4):234–236.

Denys Duchier and Ralph Debusmann. 2001. Topological dependency trees: A constraint-base account of linear precedence. In *Proc. of the 39th Annual Meeting of the ACL*, Toulouse.

Jason M. Eisner. 2000. Bilexical grammars and their cubic-time parsing algorithms. In H. Bunt and A. Nijholt, editors, *Advances in Probabilistic and Other Parsing Technologies.* Kluwer.

Haim Gaifman. 1965. Dependency systems and phrase-structure systems. *Information and Control*, 8:304–307.

David Hays. 1964. Dependency theory: A formalism and some observations. *Language*, 40(4):511–525.

Jeffrey D. Hopcroft and John E. Ullman. 1979. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley.

Richard Hudson. 1990. *English Word Grammar.* Basil Blackwell.

Vincenzo Lombardo and Leonardo Lesmo. 1998. Formal aspects and parsing issues of dependency theory. In *Proceedings of COLING/ACL'98*, pages 787–93, Montréal.

Hiroshi Maruyama. 1990a. Constraint dependency grammar. Research report RT0044, IBM, Tokyo.

Hiroshi Maruyama. 1990b. Structural disambiguation with constraint propagation. In *Proceedings of the 28th ACL*, pages 31–38, Pittsburgh, PA.

Igor A. Mel'čuk. 1988. *Dependency Syntax: Theory and Practice.* State University of New York Press.

Joakim Nivre. 2002. Two models of stochastic dependency grammar. MSI report 02118, Växjö University: School of Mathematics and System Engineering, Växjö.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of IWPT'03*, pages 149–160, Nancy.

Tomasz Obrębski. 2003. MTT-compatible computationally efficient surface-syntactic parser. In *Proceedings of MTT'03*, pages 259–268, Paris.

Daniel Sleator and Davy Temperley. 1991. Parsing english with a link grammar. Technical Report CMU-CS-91-196, Carnegie Mellon University, Computer Science Department.

Christopher M. White. 1995. Converting context-free grammars to constraint dependency grammars. Master's thesis, Purdue University, August.