Grammars for Local and Long Dependencies.

Alexander Dikovsky
Université de Nantes, IRIN, 2, rue de la Houssinière
BP 92208 F 44322 Nantes cedex 3 France
Alexandre.Dikovsky@irin.univ-nantes.fr

Abstract

Polarized dependency (PD-) grammars are proposed as a means of efficient treatment of discontinuous constructions. PD-grammars describe two kinds of dependencies : *local*, explicitly derived by the rules, and *long*, implicitly specified by negative and positive valencies of words. If in a PD-grammar the number of non-saturated valencies in derived structures is bounded by a constant, then it is weakly equivalent to a cf-grammar and has a $\boldsymbol{O}(n^3)$-time parsing algorithm. It happens that such bounded PD-grammars are strong enough to express such phenomena as unbounded raising, extraction and extraposition.

1 Introduction

Syntactic theories based on the concept of *dependency* have a long tradition. Tesnière (Tesnière, 1959) was the first who systematically described the sentence structure in terms of binary relations between words (*dependencies*), which form a *dependency tree* (*D-tree* for short). D-tree itself does not presume a linear order on words. However, any its surface realization projects some linear order relation (called also *precedence*). Some properties of surface syntactic structure can be expressed only in terms of both dependency (or its transitive closure called *dominance*) and precedence. One of such properties, *projectivity*, requires that any word occurring between a word g and a word d dependent on g be dominated by g. In first dependency grammars (Gaifman, 1961) and in some more recent proposals: link grammars (Sleator and Temperly, 1993), projective dependency grammars (Lombardo and Lesmo, 1996) the projectivity is implied by definition. In some other theories, e.g. in word grammar (Hudson, 1984), it is used as one of the axioms defining acceptable surface structures. In presence of this property, D-trees are in a sense equivalent to phrase structures with head selection [1]. It is for this reason that D-trees determined by grammars of Robinson (Robinson, 1970), categorial grammars (Bar-Hillel et al., 1960), classical Lambek calculus (Lambek, 1958), and some other formalisms are projective. Projectivity affects the complexity of parsing : as a rule, it allows dynamic programming technics which lead to polynomial time algorithms (cf. $\boldsymbol{O}(n^3)$-time algorithm for link grammars in (Sleator and Temperly, 1993)). Meanwhile, the projectivity is not the norm in natural languages. For example, in most European languages there are such regular non-projective constructions as WH- or relative clause extraction, topicalization, comparative constructions, and some constructions specific to a language, e.g. French pronominal clitics or left dislocation. In terms of phrase structure, non-projectivity corresponds to discontinuity. In this form it is in the center of discussions till 70-ies. There are various dependency based approaches to this problem. In the framework of Meaning-Text Theory (Mel'čuk and Pertsov, 1987), dependencies between (some-

[1] See (Dikovsky and Modina, 2000) for more details.

times non adjacent) words are determined in terms of their local neighborhood, which leads to non-tractable parsing (the NP-hardness argument of (Neuhaus and Bröker, 1997) applies to them). More recent versions of dependency grammars (see e.g.(Kahane et al., 1998; Lombardo and Lesmo, 1998; Bröker, 1998)) impose on non-projective D-trees some constraints weaker than projectivity (cf. meta-projectivity (Nasr, 1995) or pseudo-projectivity (Kahane et al., 1998)), sufficient for existence of a polynomial time parsing algorithm. Still another approach is developed in the context of intuitionistic resource-dependent logics, where D-trees are constructed from derivations (cf. e.g. a method in (Lecomte, 1992) for Lambek calculus). In this context, non-projective D-trees are determined with the use of hypothetical reasoning and of structural rules such as commutativity and associativity (see e.g. (Moortgat, 1990)).

In this paper, we put forward a novel approach to handling discontinuity in terms of dependency structures. We propose a notion of a *polarized dependency (PD-) grammar* combining several ideas from cf-tree grammars, dependency grammars and resource-dependent logics. As most dependency grammars, the PD-grammars are analyzing. They reduce continuous groups to their types using *local* (context-free) reduction rules and simultaneously assign partial dependency structures to reduced groups. The valencies (positive for governors and negative for subordinates) are used to specify discontinuous (*long*) dependencies lacking in partial dependency structures. The mechanism of establishing long dependencies is orthogonal to reduction and is implemented by a universal and simple rule of valencies saturation. A simplified version of PD-grammars adapted for the theoretical analysis is introduced and explored in (Dikovsky, 2001). In this paper, we describe a notion of PD-grammar more adapted for practical tasks.

2 Dependency structures

We fix finite alphabets W of *terminals* (words), C of *nonterminals* (syntactic types or classes), and N of *dependency names*.

Definition 1. *Let $x \in (W \cup C)^+$ be a string. A set $\pi = \{d_1, ..., d_n\}$ of trees (called components of π) which cover exactly x, have no nodes in common, and whose arcs are labeled by names in N is a dependency (D-) structure on x if one component d_t of π is selected as its head* [2]*. We use the notation $x = w(\pi)$. π is a terminal D-structure if x is a string of terminals. When π has only one component, it is a dependency (D-) tree on x.*

For example, the D-structure in Fig. 1 has two components. $V(aux)$ is the root of the non projective head component, the other component $GrPn(quan)$ is a unit tree.

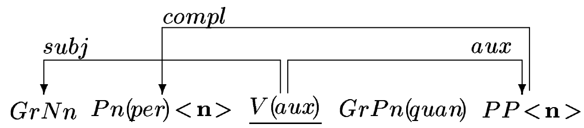


Fig. 1.

In distinction to (Dikovsky, 2001), the nonterminals (and even dependency names) can be structured. We follow (Mel'čuk and Pertsov, 1987) and distinguish syntactical $X(s)$ and morphological $X{<}m{>}$ features of a nonterminal X. The alphabets being finite, the features unification is a means of compacting a grammar.

The D-structures we will use will be *polarized* in the sense that some words will have *valencies* specifying *long dependencies* which must enter or go from them. A *valency* is an expression of one of the forms $+L:r$, $+R:r$ (a *positive* valency), or $-L:r$, $-R:r$ (a *negative* valency), r being a dependency name. For example, the intuitive sense of a positive valency $+R:r$ of a node n is that a long dependency r might go from n somewhere on the right. All nonterminals will be signed: we presume that C is decomposed into two classes : of *positive* ($C^{(+)}$) and *negative* ($C^{(-)}$) nonterminals respectively. D-structures with valencies, *DV-structures*, are defined so that valencies saturation would imply connectivity.

Definition 2. *A terminal n is polarized if a finite list of pairwise different valencies* [3] *$V(n)$ (its valency list) is assigned to it. n is positive, if $V(n)$ does not contain negative valencies, A D-tree with polarized nodes is positive if its root*

[2] We visualize d_t underlining it or its root, when there are some other components.

[3] In the original definition of (Dikovsky, 2001), valencies may repeat in $V(n)$, but this seems to be a natural constraint.

is positive, otherwise it is negative.

A D-structure π on a string x of polarized symbols is a DV-structure on x, if the following conditions are satisfied :

(v1) if a terminal node n of π is negative, then $V(n)$ contains exactly one negative valency,

(v2) if a dependency of π enters a node n, then n is positive,

(v3) the non-head components of π (if any) are all negative.

The polarity of a DV-structure is that of its head.

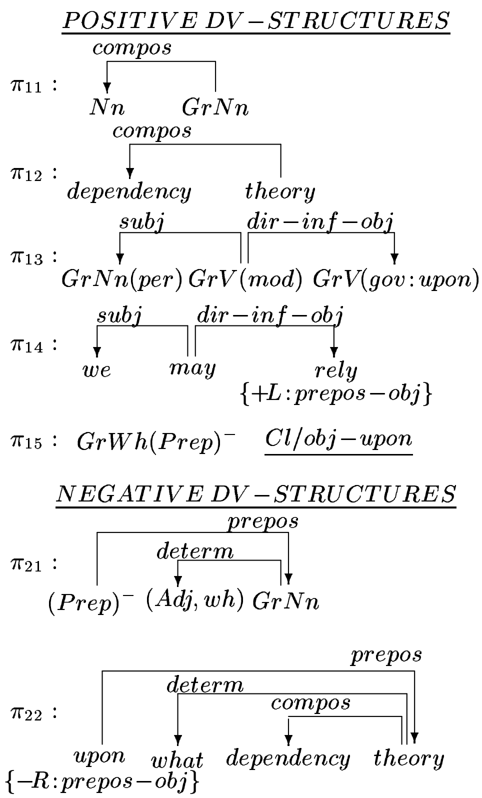


Fig. 2.

In Fig. 2 [4], both words in π_{12} have no valencies, all nonterminals in π_{11} and π_{13} are positive (we label only negative nonterminals), π_{15} is positive because its head component is a positive unit D-tree, π_{21} and π_{22} are negative because their roots are negative.

Valencies are *saturated* by *long dependencies.*

Definition 3. *Let π be a terminal DV-structure. A triplet $l =< n_1, n_2, r >$, where n_1, n_2 are nodes of π and $r \in N$, is a long dependency with the name r, directed from n_1 to n_2 (notation: $n_1 \overset{r}{- - >} n_2$), if there are valencies $v_1 \in V(n_1), v_2 \in V(n_2)$ such that :*

(v4) either $n_1 < n_2$ (n_1 precedes n_2), $v_1 = +R : r$, and $v_2 = -L : r$, or

(v5) $n_2 < n_1$, $v_1 = +L : r$, and $v_2 = -R : r$.

We will say that v_1 saturates v_2 by long dependency l.

The set of valencies in π is totally ordered by the order of nodes and the orders in their valency lists: $v_1 < v_2$ if

(o1) either $v_1 \in V_\pi(n_1), v_2 \in V_\pi(n_2)$ and $n_1 < n_2$,

(o2) or $v_1, v_2 \in V_\pi(n)$ and $v_1 < v_2$, in $V_\pi(n)$.

Let π_1 be the structure resulting from π by adding the long dependency l and replacing $V_\pi(n_1)$ by $V_{\pi_1}(n_1) = V_\pi(n_1) \setminus \{v_1\}$ and $V_\pi(n_2)$ by $V_{\pi_1}(n_2) = V_\pi(n_2) \setminus \{v_2\}$. We will say that π_1 is a saturation of π by l and denote it by $\pi \prec \pi_1$.

Among all possible saturations of π we will select the following particular one :

Let $v_1 \in V_\pi(n_1)$ be the first non saturated positive valency in π, and $v_2 \in V_\pi(n_2)$ be the closest corresponding [5] non saturated negative valency in π. Then the long dependency $l = \left(n_1 \overset{r}{- - >} n_2\right)$ saturating v_2 by v_1 is first available (FA) in π. The resulting saturation of π by l is first available or FA-saturation (notation: $\pi \prec^{FA} \pi_1$).

We transform the relations $\prec, \prec^{FA}$ into partial orders closing them by transitivity.

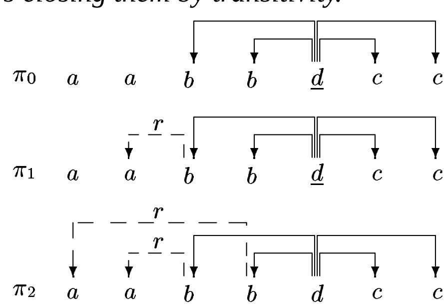


Fig. 3.

Suppose that in Fig. 3, both occurrences of a in π_0 and the first occurrence of a in π_1 have $V(a) = \{-R : r\}$, and both occurrences of b in π_0 and the second occurrence of b in π_1 have $V(b) = \{+L : r\}$. Then $\pi_0 \prec^{FA} \pi_1 \prec^{FA} \pi_2$.

[4] For the reasons of space, in our examples we are not accurate with morphological features. E.g., in the place of *GrV(gov:upon)* we should rather have *GrV(gov:upon)<inf>*.

[5] *Corresponding* means :
(c1) $n_2 < n_1$ and $v_2 = -R : r$ if $v_1 = +L : r$, and
(c2) $n_1 < n_2$ and $v_2 = -L : r$ if $v_1 = +R : r$.

In (Dikovsky, 2001), we prove that

- *If π is a terminal DV-structure and $\pi \prec \pi_1$, then either π_1 has a cycle, or it is a DV-structure* (Lemma 1).

As it follows from Definition 3, each saturation of a terminal DV-structure π has the same set of nodes and a strictly narrower set of valencies. Therefore, any terminal DV-structure has *maximal saturations* with respect to the order relations $\prec$, $\prec^{FA}$. Very importantly, *there is a single maximal FA-saturation* of π denoted $MS^1(\pi)$. E.g., in Fig. 3, $MS^1(\pi_0) = \pi_2$ is a D-tree.

In order to keep track of those valencies which are not yet saturated we use the following notion of *integral valency*.

Definition 4. *Let π be a terminal DV-structure. The integral valency $\sum\limits_{FA} \pi$ of π is the list $\bigcup\limits_{n \; in \; w(\pi)} V_{MS^1(\pi)}(n)$ ordered by the order of valencies in π. If $MS^1(\pi)$ is a d-tree, we say that this D-tree saturates π and call π saturable.*

By this definition, $\sum\limits_{FA} MS^1(\pi) = \sum\limits_{FA} \pi$.

Saturability is easily expressed in terms of integral valency (Lemma 2 in (Dikovsky, 2001)) : *Let π be a terminal DV-structure. Then :*

- *$MS^1(\pi)$ is a D-tree iff it is cycle-free and $\sum\limits_{FA} \pi = \emptyset$,*
- *π has at most one saturating D-tree.*

The semantics of PD-grammars will be defined in terms of composition of DV-structures which generalizes strings substitution.

Definition 5. *Let $\pi_1 = \{d_1, ..., d_k\}$ be a DV-structure, A be a nonterminal node of one of its components, and $\pi_2 = \{d'_1, ..., \underline{d'_t}, ..., d'_l\}$ be a DV-structure of the same polarity as A and with the head component d'_t. Then the result of the composition of π_2 into π_1 in A is the DV-structure $\pi_1[A \backslash \pi_2]$, in which π_2 is substituted for A, the root of d'_t inherits all dependencies of A in π_1, and the head component is that of π_1 (changed respectively if touched on by composition)*[6].

It is easy to see that DV-structure π in Fig. 4 can be derived by the following series of compositions of the DV-structures in Fig. 2:

$\pi_{12} = \pi_{11}[Nn \backslash dependency, GrNn \backslash theory]$,
$\pi_{22} = \pi_{21}[(Prep)^- \backslash upon, (Adj, wh) \backslash what,$
$\quad GrNn \backslash \pi_{12}]$,
$\pi_{14} = \pi_{13}[GrNn(per) \backslash we, GrV(mod) \backslash may,$
$\quad GrV(gov : upon) \backslash rely]$,
$\pi = \pi_{15}[GrWh(Prep)^- \backslash \pi_{22},$
$\quad Cl/obj{-}upon \backslash \pi_{14}]$,
and $d = MS^1(\pi)$.

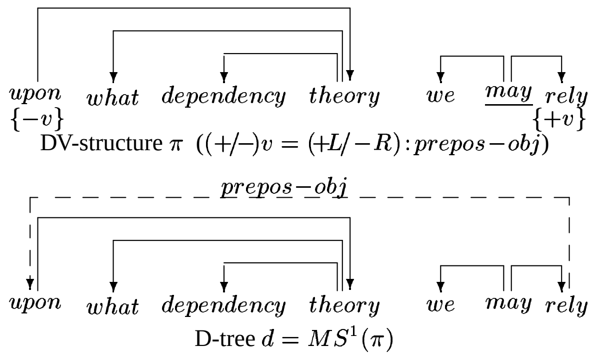


Fig. 4.

The DV-structures composition has natural properties:

- *The result of a composition into a DV-structure π is a DV-structure of the same polarity as π (Lemma 3 in (Dikovsky, 2001)).*
- *If $\sum\limits_{FA} \pi_1 = \sum\limits_{FA} \pi_2$, then $\sum\limits_{FA} \pi_0[A \backslash MS^1(\pi_1)] = \sum\limits_{FA} \pi_0[A \backslash MS^1(\pi_2)]$ for any terminal π_1, π_2 (Lemma 4 in (Dikovsky, 2001)).*

3 Polarized dependency grammars

Polarized dependency grammars determine DV-structures in the bottom-up manner in the course of reduction of phrases to their types, just as the categorial grammars do. Each reduction step is accompanied by DV-structures composition and by subsequent FA-saturation. The yield of a successful reduction is a D-tree. In this paper, we describe a superclass of grammars in (Dikovsky, 2001) which are more realistic from the point of view of real applications and have the same parsing complexity.

Definition 6. *A PD-grammar is a system $G = (W, C, N, I, \lambda, R)$, where W, C, N are as described above, $I \subseteq C^{(+)}$ is a set of axioms (which are positive nonterminals), $\lambda \subseteq W \times C \times L$ is a ternary relation of lexical interpretation, L being*

[6] This composition generalizes the substitution used in TAGs (Joshi et al., 1975) (A needs not be a leaf) and is not like the adjunction.

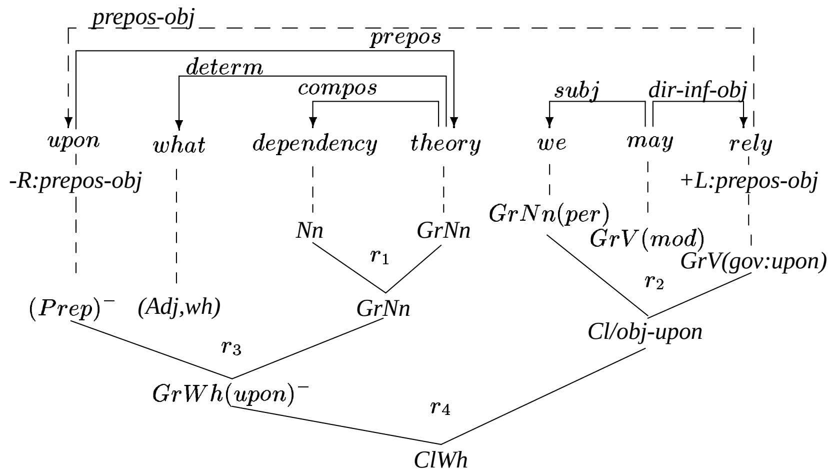


Fig. 5.

the set of lists of pairwise different valencies, and R is a set of reduction rules. For simplicity, we start with the strict reduction rules (the only rules in (Dikovsky, 2001)) of the form $\pi \to A$, where $A \in C$ and π is a DV-structure over C of the same polarity as A (below we will extend the strict rules by side effects). In the special case, where the DV-structures in the rules are D-trees, the PD-grammar is local[7].

Intuitively, we can think of λ as of the combined information available after the phase of morphological analysis (i.e. dictionary information and tags). So $(w, A, vl) \in \lambda$ means that a type A and a valency list vl can be a priori assigned to the word w.

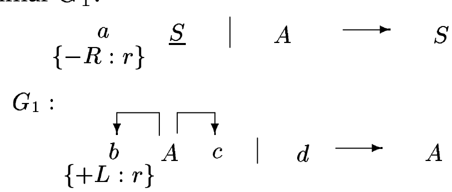
Semantics. *1. Let $r = (w, A, vl) \in \lambda$ and π_0 be the unit DV-structure w with $V(w) = vl$. Then r is a reduction of the structure π_0 to its type A (notation $\pi_0 \vdash^r A$) and vl is the integral valency of this reduction denoted by $vl = \sum\limits_r \pi_0$.*

2. Let $r = (\pi \to A)$ be a reduction rule with k nonterminals' occurrences $A_1, ..., A_k$ in π, $k > 0$, and $\pi_1 \vdash^{\rho_1} A_1, ..., \pi_k \vdash^{\rho_k} A_k$ be some reductions. Then $\rho = (\rho_1 ... \rho_k; r)$ is a reduction of the structure $\pi_0 = MS^1(\pi[A_1 \backslash \pi_1, ..., A_k \backslash \pi_k])$ to its type A (notation $\pi_0 \vdash^\rho A$). $\rho_1, ..., \rho_k$ as well as ρ itself are subreductions of ρ. The integral valency of π_0 via ρ is $\sum\limits_\rho \pi_0 = \sum\limits_{FA} \pi[A_1 \backslash \pi_1, ..., A_k \backslash \pi_k] = \sum\limits_{FA} \pi_0$. A D-tree d is determined by G if there is a reduction $d \vdash^\rho S$, $S \in I$. The DT-language determined by G is the set $D(G)$ of all D-trees it determines. $L(G) = \{w(d) \mid d \in D(G)\}$ is the language determined by G. $\mathcal{L}(PDG)$ denotes the class of languages determined by PD-grammars.

[7]Local PD-grammars are strongly equivalent to dependency tree grammars of (Dikovsky and Modina, 2000) which are generating and not analyzing as here.

By way of illustration, let us consider the PD-grammar G_0 with the lexical interpretation λ containing triplets:
$(we, GrNn(per), [\ ])$, $(may, GrV(mod), [\ ])$
$(rely, GrV(gov{:}upon), [+L{:}prepos-obj])$,
$(upon, (Prep)^-, [-R{:}prepos-obj])$,
$(what, (Adj, wh), [\ ])$, $(dependency, Nn, [\ ])$,
$(theory, GrNn, [\,])$, and the following reduction rules whose left parts are shown in Fig. 2:
$r_1 = (\pi_{11} \to GrNn)$,
$r_2 = (\pi_{13} \to Cl/obj-upon)$,
$r_3 = (\pi_{21} \to GrWh(upon)^-)$,
$r_4 = (\pi_{15} \to ClWh)$.
Then the D-tree d in Fig. 4 is reducible in G_0 to $ClWh$ and its reduction is depicted in Fig. 5.

As we show in (Dikovsky, 2001), the weak generative capacity of PD-grammars is stronger than that of cf-grammars. For example, the PD-grammar G_1:



generates a non-cf language $\{w(n) \mid w(n) = a^n b^n d c^n,\ n \geq 0\}$. D-tree π_2 in Fig. 3 is determined by G_1 on $w(2)$. Its reduction combined with the diagram of local and long dependencies is presented in Fig. 6.

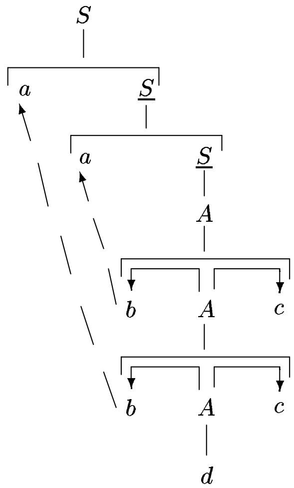


Fig. 6.

The local PD-grammars are weakly equivalent to cf-grammars, so they are weaker than general PD-grammars. Meanwhile, what is really important concerning the dependency grammars, is their strong generative capacity, i.e. the D-trees they derive. From this point of view, the grammars like G_1 above are too strong. Let us remark that in the reduction in Fig. 6, the first saturation becomes possible only after *all* positive valencies emerge. This means that the integral valency of subreductions increases with n. This seems to be never the case in natural languages, where next valencies arise only after the preceding ones are saturated. This is why we restrict ourself to the class of PD-grammars which have such a property.

Definition 7. *Let G be a PD-grammar. For a reduction ρ of a terminal structure, its defect is defined as $\sigma(\rho) = max\{|\sum_{\rho'} \pi'| \mid \rho'$ is a subreduction of $\rho\}$. G has bounded (unbounded) defect if there is some (there is no) constant q which bounds the defect of all its reductions. The minimal constant q having this property (if any) is the defect of G (denoted $\sigma(G)$).*

There is a certain technical problem concerning PD-grammars. Even if in a reduction to an axiom all valencies are saturated, this does not guarantee that a D-tree is derived: the graph may have cycles. In (Dikovsky, 2001) we give a sufficient condition for a PD-grammar of never producing cycles while FA-saturation. We call the grammars satisfying this condition *lc- (locally cycle-) free.* For the space reasons, we don't cite its definition, the more so that the linguistic PD-grammars should certainly be lc-free. In (Dikovsky, 2001) we prove the following theorem.

Theorem 1. *For any lc-free PD-grammar G of bounded defect there is an equivalent cf-grammar.*

Together with this we show an example of a DT-language which cannot be determined by local PD-grammars. This means that not all structures determined in terms of long dependencies can be determined without them.

4 Side effect rules and parsing

An important consequence of Theorem 1 is that lc-free bounded defect PD-grammars have a $\boldsymbol{O}(n^3)$ parsing algorithm. In fact, it is the classical Earley algorithm in charter form (the charters being DV-structures). To apply this algorithm in practice, we should analyze the asymptotic factor which depends on the size of the grammar. The idea of theorem 1 is that the integral valency being bounded, it can be compiled into types. This means that a reduction rule $\pi \rightarrow A$ should be substituted by rules $\pi[A_1 \backslash A_1[V_1], ..., A_k \backslash A_k[V_k]] \rightarrow A[V_0]$ with types keeping all possible integral valencies not causing cycles. Theoretically, this might blow up $v^{k(q+1)}$ times the size of a grammar with defect q, v valencies and the maximal length k of left parts of rules. So theoretically, the constant factor in the $\boldsymbol{O}(n^3)$ time bound is great. In practice, it shouldn't be as awful, because in linguistic grammars q will certainly equal 1, one rule will mostly treat one valency (i.e. $k = 1$) and the majority of rules will be local. Practically, the effect may be that some local rules will have variants propagating upwards a certain valency: $\pi[B \backslash B[v]] \rightarrow A[v]$. The actual problem lies elsewhere. Let us analyze the illustration grammar G_0 and the reduction in Fig. 5. This reduction is successful due to the fact that the negative valency $-R\!:\!prepos\!-\!obj$ is assigned to the preposition $upon$ and the corresponding positive valency $+L\!:\!prepos\!-\!obj$ is assigned to the verb $rely$. What might serve the formal basis for these assignments? Let us start with $rely$. This verb has the strong government over prepositions $on, upon$. In the clause in Fig. 4, the group of the preposition is moved, which is of course a sufficient condition for assigning the positive valency to the verb. But this condition is not available in the dictionary, nor even through morphological analysis ($rely$ may occur at a certain distance from the end of the clause). So it can only be derived in the course of reduction, but strict PD-grammars have no rules assigning valencies. Theoretically, there is no problem: we should just introduce into the dictionary both variants of the verb description – with the local dependency $prepos-obj$ to the right and with the positive valency $+L{:}prepos-obj$ to the left. Practically, this "solution" is inacceptable because such a lexical ambiguity will lead to a brute force search. The same argument shows that we shouldn't assign the negative valency $-R{:}prepos-obj$ to $upon$ in the dictionary, but rather "calculate" it in the reduction. If we compare the clause in Fig. 4 with the clauses *what theories we may rely upon; what kind of theories we may rely upon; the dependency theories of what kind we may rely upon* etc., we see that we can assign a $-R$ valency to *wh*-words in the dictionary and then *raise* negative valencies till the FA-saturation. The problem is that in the strict PD-grammars there are no rules of valency raising. For these reasons we extend the reduction rules by side effects sufficient for the calculations of both kinds.

Definition 8. *We introduce two kinds of side effects: valency raising* $([v_1](i) \overset{r}{\uparrow} v_2)$ *and valency assignment* $((i) \Leftarrow v)$, v, v_1, v_2 *being valency names and* i *an integer. A rule of the form*

$$\pi([v_1](i) \overset{r}{\uparrow} v_2) \to A$$

with k *nonterminals* $A_1, ..., A_k$ *in* π *and* $1 \leq i \leq k$ *is valency raising if :*

(r1) v_1, v_2 *are of the same polarity,*

(r2) a local dependency r *enters* A_i *in* π,

(r3) for positive v_1, v_2, $\pi \to A$ *is a strict reduction rule,*

(r4) if v_1, v_2 *are negative, then* $A_i, A \in C^{(-)}$, *and replacing* A_i *by any positive nonterminal we obtain a DV-structure* [8]*. A rule of the form*

$$\pi((i) \Leftarrow v) \to A$$

with k *nonterminals* $A_1, ..., A_k$ *in* π *and* $1 \leq i \leq k$ *is valency assigning if :*

(a1) for a positive v, $\pi \to A$ *is a strict reduction rule,*

(a2) if v *is negative and* A_i *is the root of* π, *then* $A_i \in C^{(+)}$ *and* $A \in C^{(-)}$,

(a3) if v *is negative and* A_i *is not the root of* π, *then* $A \in C^{(-)}$, $A_i \in C^{(+)}$ *is a non head component of* π [9] *and replacing* A_i *by any negative nonterminal we obtain a DV-structure.*

Semantics. *We change the reduction semantics as follows.*

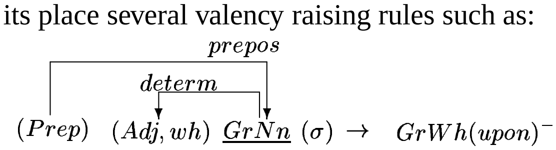
- *For a raising rule* $\pi([v_1](i) \overset{r}{\uparrow} v_2) \to A$, *the result of the reduction is the DV-structure* $\pi_0 = MS^1(\alpha(v_2, \pi[A_1 \backslash \pi_1, ..., A_i \backslash \delta(v_1, \pi_i), ..., A_k \backslash \pi_k]))$, *where* $\delta(v, \pi')$ *is the DV-structure resulting from* π' *by deleting* v *from* $V(root(\pi'))$, *and* $\alpha(v, \pi')$ *is the DV-structure resulting from* π' *by adding* v *to* $V(root(\pi'))$.
- *For a valency assignment rule* $\pi((i) \Leftarrow v) \to A$, *the result of the reduction is the DV-structure* $\pi_0 = MS^1(\pi[A_1 \backslash \pi_1, ..., A_i \backslash \alpha(v, \pi_i), ..., A_k \backslash \pi_k])$.

A PD-grammar with side effect rules is a PDSE-grammar.

This definition is correct in the sense that the result of a reduction with side effects is always a DV-structure. We can prove

Theorem 2. *For any lc-free PDSE-grammar* G *of bounded defect there is an equivalent cf-grammar.*

Moreover, the bounded defect PDSE-grammars are also parsed in time $\boldsymbol{O}(n^3)$. In fact, we can drop negative v_1 in raising rules (it is unique) and indicate the type of $root(\pi)$ in both side effect rules, because the composition we use makes this information local. Now, we can revise the grammar G_0 above, e.g. excluding the dictionary assignment $(upon, (Prep)^-, [-R{:}prepos-obj])$, and using in its place several valency raising rules such as:



where $\sigma = ([\,](3(GrNn)) \overset{prepos}{\uparrow}{}_{Prep}$ *-R:prepos-obj*$)$.

5 Conclusion

The main ideas underlying our approach to discontinuity are the following:

[8] So this occurrence of A_i in π contradicts to the point (v2) of definition 2.

[9] So this occurrence of A_i in π contradicts to the point (v3) of definition 2.

• Continuous (local, even if non projective) dependencies are treated in terms of trees composition (which reminds TAGs). E.g., the French pronominal clitics can be treated in this way.

• Discontinuous (long) dependencies are captured in terms of FA-saturation of valencies in the course of bottom-up reduction of dependency groups to their types. As compared with the SLASH of GPSG or the regular expression lifting control in non projective dependency grammars, these means turn out to be more efficient under the conjecture of bounded defect. This conjecture seems to be true for natural languages (the contrary would mean the possibility of unlimited extraction from extracted groups).

• The valency raising and assignment rules offer a way of deriving a proper valency saturation without unwarranted increase of lexical ambiguity.

A theoretical analysis and experiments in English syntax description show that the proposed grammars may serve for practical tasks and can be implemented by an efficient parser.

6 Acknowledgments

I would like to express my heartfelt gratitude to N. Pertsov for fruitful discussions of this paper. The idea of valency raising has emerged from our joint work over a project of a PD-grammar for a fragment of English.

References

Y. Bar-Hillel, H. Gaifman, and E. Shamir. 1960. On categorial and phrase structure grammars. *Bull. Res. Council Israel*, 9F:1–16.

N. Bröker. 1998. Separating surface order and syntactic relations in a dependency grammar. In *Proc. COLING-ACL*, pages 174–180, Montreal.

A.Ja. Dikovsky and L.S. Modina. 2000. Dependencies on the other side of the Curtain. *Traitement Automatique des Langues (TAL)*, 41(1):79–111.

A. Dikovsky. 2001. Polarized non-projective dependency grammars. In Ph. De Groote and G. Morrill, editors, *Logical Aspects of Computational Linguistics*, number 2099 in LNAI. Springer Verlag. To be published.

H. Gaifman. 1961. Dependency systems and phrase structure systems. Report p-2315, RAND Corp. Santa Monica (CA). Published in: Information and Control, 1965, v. 8, n ° 3, pp. 304-337.

R.A. Hudson. 1984. *Word Grammar*. Basil Blackwell, Oxford-New York.

A.K. Joshi, L.S. Levy, and M. Takahashi. 1975. Tree adjunct grammars. *Journ. of Comput. and Syst. Sci.*, 10(° 1):136–163.

S. Kahane, A. Nasr, and O. Rambow. 1998. Pseudo-projectivity : A polynomially parsable non-projective dependency grammar. In *Proc. COLING-ACL*, pages 646–652, Montreal.

J. Lambek. 1958. The mathematics of sentence structure. *American Mathematical Monthly*, pages 154–170.

A. Lecomte. 1992. Proof nets and dependencies. In *Proc. of COLING-92*, pages 394–401, Nantes.

V. Lombardo and L. Lesmo. 1996. An earley-type recognizer for dependency grammar. In *Proc. 16th COLING*, pages 723–728.

V. Lombardo and L. Lesmo. 1998. Formal aspects and parsing issues of dependency theory. In *Proc. COLING-ACL*, pages 787–793, Montreal.

I. Mel'čuk and N.V. Pertsov. 1987. *Surface Syntax of English. A Formal Model Within the Meaning-Text Framework*. John Benjamins Publishing Company, Amsterdam/Philadelphia.

M. Moortgat. 1990. La grammaire catégorielle généralisée : le calcul de lambek-gentzen. In Ph. Miller and Th. Torris, editors, *Structure of languages and its mathematical aspects*, pages 127–182. Hermes, Paris.

A. Nasr. 1995. A formalism and a parser for lexicalized dependency grammars. In *Proc. Int. Workshop on Parsing Technology*, pages 186–195, Prague.

P. Neuhaus and N. Bröker. 1997. The Complexity of Recognition of Linguistically Adequate Dependency Grammars. In *Proc. of 35th ACL Annual Meeting and 8th Conf. of the ECACL*, pages 337–343.

Jane J. Robinson. 1970. Dependency structures and transformational rules. *Language*, 46(° 2):259–285.

D. D. Sleator and D. Temperly. 1993. Parsing English with a Link Grammar. In *Proc. IWPT'93*, pages 277–291.

L. Tesnière. 1959. *Éléments de syntaxe structurale*. Librairie C. Klincksieck, Paris.