A Finite-State Functional Grammar Architecture

Alexander Dikovsky

LINA FRE-CNRS 2729, Université de Nantes, France
alexandre.dikovsky@univ-nantes.fr

Abstract. We describe a simple and efficiently implementable grammatical architecture for generating dependency trees from meaning structures. It is represented by finite state tree top-down transducers applied to feature trees to convert them into sequences of typed forms (typed sentences). The correctness of the generated types is checked using a simple and efficient dependency calculus. The corresponding dependency structure is extracted from the correctness proof. Using an English example, it is demonstrated that this transduction can be carried out incrementally in the course of composition of the meaning structure.

1 Introduction

The issue under study in this paper is how to design the grammar so that *the surface representation of sentences were incrementally generated in the course of planning their meaning*. So stated, this issue does not fit the standard generativistic frame, nor even in its modern form [4,15]. The closest suitable frame is that of functional grammar, more specifically, the Prague Functional Generative Description [21], where a similar issue was already studied [22] and the Lexical Functional Grammar [19], especially the Optimal Syntax (OS) proposal [3]. As with OS, the *input meaning representation* we consider is underspecified with respect to the logical form. This representation, called *discourse plan* (DP), inspired by DRT [17] and Frame Semantics [13] was introduced in [8] and developed in [11]. The DP are a kind of feature trees which combine argument thematic roles, lexical types, topic/focus/background partitioning, distributive/non-distributive collectivity, co-reference, scope hierarchy etc., and are interpreted using typed lambda-terms. The DP underspecification is completely different from the universally adopted "scope-precedence-abstraction" underspecification common to many proposals (cf. [20,2,5,14]). Its distinguishing feature is that, theoretically, DP should uniquely encode argument scopes through their *communicative ranks* in *semantic diatheses*, a concept close to the *argument perspective* in Frame Semantics. This fundamental difference has far-reaching semantic consequences. E.g., an adequate DP of the sentence *Every linguist knows two languages* represents only its $\forall\exists$ reading (the dual one needs a different perspective expressed through a different wording). Another consequence is that ideally, in contrast to OS, the correspondence mapping between DP and surface structures should be *functional*. In our approach to generation, we make two important choices: we choose *dependency structures* (DS) as the surface structure representation and

D. Leivant and R. de Queiroz. (Eds.): WoLLIC 2007, LNCS 4576, pp. 131–146, 2007.

we use Categorial Dependency Grammars (CDG) [9,6] as the basic grammatical formalism. CDG are lexicalized type grammars defined using a simple dependency type calculus. They are significantly more expressive than CF-grammars and are parsed in polynomial time. In contrast to OS and similar parallel correspondence syntactic theories originating from LFG, in the generation architecture described in this paper, the functional correspondence mapping converts DP into sequences of typed surface forms. The DP is feasible if the generated typing is proved correct in the dependency calculus. The corresponding surface DS is extracted from the proof. So, at least theoretically, this architecture can be seen as a "generate and check" process. The well-typing proof may or may not be a part of the conversion. In the first case, the conversion may also be seen as incremental "generation-while-planning". One can expect that the generation process is substantially simpler than the corresponding parsing process. Indeed, such hard problems as lexical ambiguity, anaphora resolution, ellipsis etc. do not arise while generation. This is why we choose deterministic finite state tree transducers (FST) as a formal model of the functional correspondence mapping.

In what follows, we sketch out the DP and the CDG dependency calculus, next we define FST over general feature trees and finally we describe their subclass converting DP into typed sequences (*DP-DS-converters*). The architecture is illustrated by a fragment of a modular DP-DS-converter for the English. This example shows that at least the core part of this architecture allows generation-while-planning.

2 Discourse Plans

Our choice of DP in the role of input meaning representation is principally explained by the way it represents the (verb) argument structure. In DP, as in Frame Semantics, one and the same structure (*situation* in DP terms) may have different arguments when seen from different perspectives, and the arguments are identified with *thematic roles* and provided with *lexical types*. In the DP below, are used the most general proto-roles. E.g., the role SBJ corresponds to the semantic subject/agent, OBJ corresponds to the semantic patient/object/theme, INSTR corresponds to the semantic instrument/mediator. We don't choose between different theories of thematic roles (cf. [16]) and proceed from the following assumptions. Firstly, the roles are labels for classes of semanteme arguments which uniquely identify the arguments for every semanteme of a given lexical type. Secondly, the thematic roles form a *semantic prominence* hierarchy with the effect that if a role R_1 is more prominent than a role R_2 ($R_1 > R_2$, e.g., SBJ>OBJ) co-occuring in a situation S, then the R_1-argument *overscopes* the R_2-argument of S. This assumption is perfectly conformable to the main approaches to the prominence order: the logical approach [12], where some variables in the entailed characteristic properties of "less prominent" role arguments are bound with values instantiated in the properties of "more prominent" role arguments, and the approach of lexical semantics (cf. [16]) in which "less prominent" means "deeper in the event structure defining the situation's lexical meaning". The

lexical types used in DP form a hierarchy. Above four most general types: **s** (*sententials*), **n** (*nominators*), **q** (*qualifiers*), **c** (*circumstantials*) there are more specific ones, e.g. $\mathbf{s}{\prec}\mathbf{s_{eff}}$ (*caused effect situations*), $\mathbf{n}{\prec}\mathbf{n_a}$ (*animated nominators*), $\mathbf{q}{\prec}\mathbf{q_{prop}}$ (*property qualifiers*), $\mathbf{c}{\prec}\mathbf{c_{mann}}$ (*manner circumstantials*).

Identification of arguments with thematic roles is standard in subcategorization frames, but DP go further and, following the proposal of the Prague school ([21]), introduce into the frames elements of communicative structure to distinguish between different perspectives (*situation profiles* in DP terms). E.g., in the DP in Fig. 1 is used situation ⟪open⟫ in its canonical profile:

$$\left\langle\!\left\langle \mathrm{open}\Big(\textsc{Sbj}^{\mathbf{n_a}}, \textsc{Obj}^{\mathbf{n}}, \textsc{Instr}^{\mathbf{n}}\Big)^{\mathbf{s_{eff}}} \right\rangle\!\right\rangle,$$

where it has value type $\mathbf{s_{eff}}$ and three nominator arguments with the roles Sbj, Obj, Instr. All other possible situation profiles are derived from the canonical one using *semantic diatheses*, i.e. explicit transformations of the canonical profile depending on the speaker's view of the salience and the intended type of each semanteme's *actant* (its proper argument identified by a thematic role, as opposed to other optional circumstantial arguments identified by other attributes). So every situation in every its profile has a particular first order type with primitive argument subtypes. The views are expressed using several *communicative ranks* assigned to the actants: T(topic), O(topic to be elided in the surface structure), $\odot$(focus), $\oplus$(background), $\ominus$(periphery). The topicality order $\mathsf{T}/\mathsf{O} > \odot > \oplus$ corresponds to the prominence order of the actants in the canonical profile. The diatheses assign to the actants new ranks, roles and/or types and eventually change the situation value type. In this way (rather close to [18]) are defined the diatheses of passivization, nominalization, object alternation, etc. E.g., the DP in Fig. 2 makes use of the diathesis $decaus{-}ins$ of ⟪open⟫

$$\langle\!\langle \mathrm{dth}_{\mathrm{decaus-ins}}\,(\mathrm{key}_{\oplus}, \emptyset \hookleftarrow \textsc{Sbj}_{\ominus}, \textsc{Sbj} \hookleftarrow \textsc{Instr}_{\mathsf{T}}, \textsc{Obj} \hookleftarrow \textsc{Obj}_{\odot})^{\mathbf{s_{eff}}} \rangle\!\rangle$$

deriving the profile in which the peripheral Sbj-actant is eliminated, the topicalized Instr-actant is promoted to Sbj and the Obj-actant is left intact.

Example 1. *John easily opened the door with the new key*

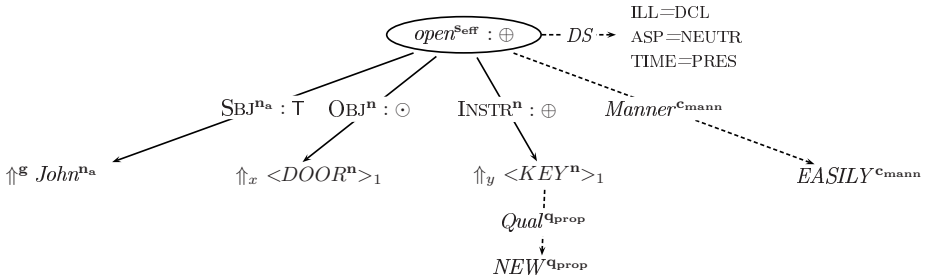


Fig. 1. A DP using the canonical profile of ⟪open⟫

(Representing DP, we use the following graphical notation. The ellipses signify situations. The boxes represent the diatheses, in particular, the assignment of

communicative ranks to the actants. Solid lines link the actants to their semantemes and are labeled with roles. Dashed lines link circumstantials to their governor semantemes and are labeled with names of attributes. Several non-optional attributes represent the so called *discourse status* and state the situation's illocutive status, the aspect and the semantic time). Diamonds represent embedded functions: e.g. the $\iota-$operator. $\Uparrow^{\mathbf{g}}$ stands for global context references and $\Downarrow_x, \Uparrow_x$ stand for anaphoric references introduction and access. The superscripts correspond to lexical types.)

Example 2. *The new key easily opened the door*

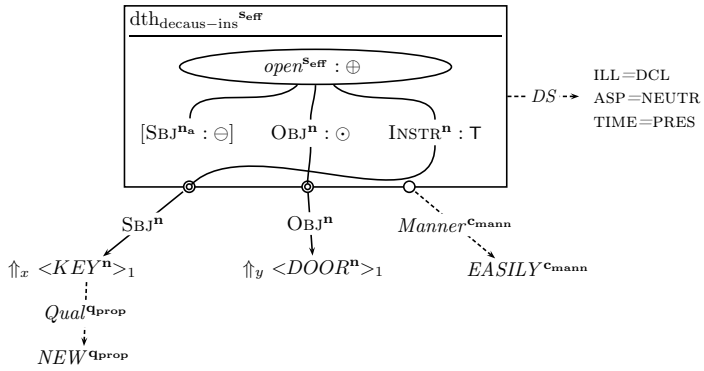


Fig. 2. *DP DP_{decaus} ing a emantic de i a i e f ⟪ en⟫*

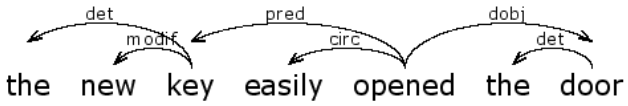
From the semantic perspective, the DP may be seen as first order typed terms serving for compositional definition of semantic expressions. From the syntactic perspective, they are a special kind of feature trees.

Definition 1. *The set of* feature trees *(*f-trees*) over* functors $F = \{f/n \mid n \geq 0\}$ *and sorts Σ is the least set $\mathcal{T}(F, \Sigma)$ which (i) contains all constants $f/0 \in F$, and (ii) contains all expressions $f(b_1 : t_1, \ldots, b_n : t_n)$ for all $f/n \in F$, all* pairwise different *sorts $b_1, \ldots, b_n \in \Sigma$ and all f-trees $t_1, \ldots, t_n \in \mathcal{T}(F, \Sigma)$.*

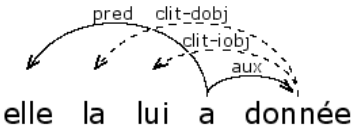
In particular, a DP using as its main situation a situation K in derived profile $\text{dth}_{\text{d}}(\text{K})$, can be seen as the f-tree $\text{dth}_{\text{d}}(\text{K})(\text{s}_1 : \pi_1, \ldots, \text{s}_{\text{m}} : \pi_{\text{m}})$, where the sorts s_i are either the roles of shifted actants with non-null ranks or the attributes identifying present circumstantials, and π_i are the f-trees representing the corresponding argument sub-plans. For instance, in the DP in Fig. 1 using ⟪open⟫ in canonical profile, $s_1 = \text{SBJ}^{\mathbf{n_a}}$, $s_2 = \text{OBJ}^{\mathbf{n}}$, $s_3 = \text{INSTR}^{\mathbf{n}}$, $s_4 = \text{MANNER}^{\mathbf{c_{mann}}}$. On the other hand, in the f-tree representing the DP in Fig. 2, which uses the derivative $\langle\!\langle \text{dth}_{\text{decaus}-\text{ins}}(\text{open}) \rangle\!\rangle$, $s_1 = \text{SBJ}^{\mathbf{n}}$, $s_2 = \text{OBJ}^{\mathbf{n}}$ and $s_3 = \text{MANNER}^{\mathbf{c_{mann}}}$.

3 Generalized Categorial Dependency Grammars

The *generalized categorial dependency grammars* (gCDG) recently introduced in [9] and studied in [6,7] are categorial grammars generating sentences together with their *dependency structures* (DS). The DS may be trees (DT) as it is the case in Fig. 3 or not (cf. Fig . 4). Whatever is the case, they are linearly ordered and this order may be *projective*, as in Fig. 3(a), or not, as in Fig. 3(b). The non-projective order is due to *discontinuous dependencies* in which the governor g is separated from the subordinate s by a word not dominated by g (as it is the case of the auxiliary verb a in Fig. 3(b) separating the participle *donnée* from its pronominalized objects).



(a) A projective DT



(b) A non-projective DT (French: **she* it_{FEM} *to-him has given*)

Fig. 3.

Like the classical categorial grammars, gCDG are lexicalized, i.e. are defined by a *lexicon*: a function λ assigning to each word a finite set of its dependency types. A type has the form $[l_m \backslash \ldots \backslash l_1 \; v / r_n \ldots / r_1]^P$, where v is the (continuous) dependency on the governor, $l_m, \ldots, l_1$ and $r_n, \ldots, r_1$ are the (continuous) dependencies of left and right subordinates in the reverse order, and P is a sequence of *polarized valencies* (the *potential*) determining discontinuous dependencies. Types with empty potentials determine projective DT. E.g., the DT in Fig. 3(a) is determined by the type assignments:

$$\text{the} \mapsto det \qquad \text{key} \mapsto [modif * \backslash det \backslash pred] \qquad \text{new} \mapsto modif$$
$$\text{easily} \mapsto circ \; \text{opened} \mapsto [circ * \backslash pred \backslash S / dobj] \; \text{door} \mapsto [det \backslash dobj]$$

where the types $modif$ and $circ$ are *iterated* and S is the sentence type. The polarized valencies are of four kinds: $\swarrow d$ (negative: that of the distant right governor through dependency d), $\searrow d$ (same on the left), $\nwarrow d$ (positive: that of a distant left subordinate through dependency d), $\nearrow d$ (same on the right). A pair of *dual* valencies $\swarrow d$ and $\nwarrow d$ ($\nearrow d$ and $\searrow d$) define a discontinuous dependency d. A negative valency can be anchored on a host word using anchor

types $\#(\swarrow d)$, $\#(\searrow d)$. E.g., the lexicon below uniquely determines the DT in Fig. 3(b) ($\swarrow clit{-}dobj, \swarrow clit{-}iobj$ are anchored on the auxiliary a):

elle $\mapsto pred$ la $\mapsto [\#(\swarrow clit{-}dobj)]^{\swarrow clit\text{–}dobj}$ lui $\mapsto [\#(\swarrow clit{-}iobj)]^{\swarrow clit\text{–}iobj}$
a $\mapsto [\#(\swarrow clit{-}iobj)\backslash\#(\swarrow clit{-}dobj)\backslash pred\backslash S/aux]$ donnée $\mapsto [aux]^{\nwarrow clit\text{–}dobj \nwarrow clit\text{–}iobj}$

Such dependency types are formalized with the following calculus [1]

L$^{\mathbf{l}}$. $C^{P_1}[C\backslash\beta]^{P_2} \vdash [\beta]^{P_1P_2}$
I$^{\mathbf{l}}$. $C^{P_1}[C^*\backslash\beta]^{P_2} \vdash [C^*\backslash\beta]^{P_1P_2}$
Ω$^{\mathbf{l}}$. $[C^*\backslash\beta]^{P} \vdash [\beta]^{P}$
D$^{\mathbf{l}}$. $\alpha^{P_1(\swarrow C)P(\nwarrow C)P_2} \vdash \alpha^{P_1PP_2}$, if $(\swarrow C)P(\nwarrow C)$ satisfies the pairing rule **FA** (first available): P has no occurrences of $\swarrow C, \nwarrow C$.

L$^{\mathbf{l}}$ is the classical elimination rule. Eliminating the argument subtype $C \neq \#(\alpha)$ it constructs the (projective) dependency C and concatenates the potentials. $C = \#(\alpha)$ creates no dependency. **I**$^{\mathbf{l}}$ derives $k > 0$ instances of C. **Ω**$^{\mathbf{l}}$ serves for the case $k = 0$. **D**$^{\mathbf{l}}$ derives discontinuous dependencies. It pairs and eliminates dual valencies satisfying the rule **FA** to create the discontinuous dependency C.

For a DS D and a string x, let $G(D, x)$ denote the relation: D is constructed in a proof $\Gamma \vdash S$ for some $\Gamma \in \lambda(x)$. Then the *language* generated by G is the set $L(G) =_{df} \{w \parallel \exists D\ G(D, w)\}$.

The gCDG are very expressive. Evidently, they generate all CF-languages. They can also generate non-CF languages.

Example 3. *The gCDG*

$$G_{abc} : a \mapsto A^{\swarrow A}, [A\backslash A]^{\swarrow A},\ b \mapsto [B/C]^{\nwarrow A}, [A\backslash S/C]^{\nwarrow A},\ c \mapsto C, [B\backslash C]$$

generates the language $\{a^n b^n c^n \mid n > 0\}$ *[9]. For instance,* $G_{abc}(D^{(3)}, a^3b^3c^3)$ *holds for the dependency structure (DS) in Fig. 4 and the string* $a^3b^3c^3$ *due to the proof in Fig. 5.*

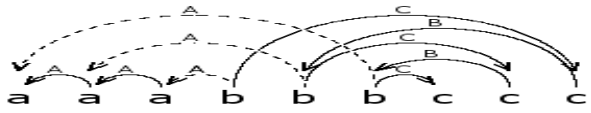


Fig. 4. Dependency structure for $a^3b^3c^3$

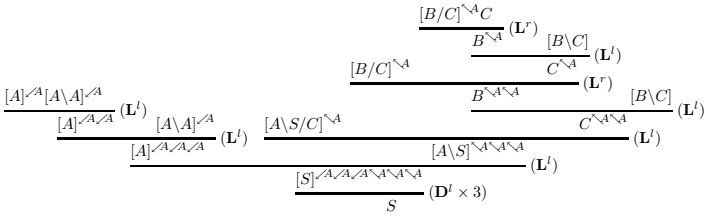


Fig. 5. Dependency structure correctness proof

[1] We show left-oriented rules. The right-oriented are symmetric.

Recently, it was shown that the family of gCDG-languages $\mathcal{L}(gCDG)$ is an AFL [7]. Seemingly, it is incomparable with the mildly CS languages [24], generated by multi-component TAG, linear CF rewrite systems, minimalist grammars [23]. $\mathcal{L}(gCDG)$ contains non-TAG languages, e.g. $L^{(m)} = \{a_1^n a_2^n ... a_m^n \mid n \geq 1\}$ for all $m > 0$. In particular, it contains the language $MIX = \{w \in \{a,b,c\}^+ \mid |w|_a = |w|_b = |w|_c\}$, for which E. Bach conjectures that it is not mildly CS. On the other hand, we conjecture [6] that this family does not contain the copy language $L_{copy} = \{xx \mid x \in \{a,b\}^*\}$, which is TAG [2]. gCDG have an efficient polynomial time parsing algorithm [6,7] which rests upon a property of independence of basic categories and polarized valencies in the proofs in the dependency calculus. This property is expressed in terms of *projections* of categories and *"well-bracketing"* criteria for potentials.

For a sequence of categories γ, its *local projection* $\|\gamma\|_l$ and *de valency projection* $\|\gamma\|_v$ are defined as follows:

1. $\|\varepsilon\|_l = \|\varepsilon\|_v = \varepsilon$; $\|\alpha\gamma\|_l = \|\alpha\|_l\|\gamma\|_l$ and $\|\alpha\gamma\|_v = \|\alpha\|_v\|\gamma\|_v$ for a category α.
2. $\|C^P\|_l = C$ et $\|C^P\|_v = P$ for every category C^P.

To speak about "well-bracketing" of potentials, we interpret $\swarrow d$ and $\nearrow d$ as *left brackets* and $\nwarrow d$ and $\searrow d$ as *right brackets*. For a left bracket valency α, the corresponding (*dual*) right bracket valency is denoted $\breve{\alpha}$. The set of left-bracket valencies is denoted V^l. That of right-bracket valencies is denoted V^r. $V =_{df} V^l \cup V^r$. A potential is *balanced* if it is well bracketed in the usual sense.

For a potential P and dual valencies $\alpha \in V^l$, $\breve{\alpha} \in V^r$, the values

$$\Delta_\alpha(P) = max\{|P'|_\alpha - |P'|_{\breve{\alpha}} \mid P' \textit{ is a } \mathsf{suffix} \textit{ of } P\},$$
$$\Delta_{\breve{\alpha}}(P) = max\{|P'|_{\breve{\alpha}} - |P'|_\alpha \mid P' \textit{ is a } \mathsf{prefix} \textit{ of } P\}$$

(where $|\gamma|_\alpha$ is the number of occurrences of α in γ) express the deficits of right and left $\alpha-$brackets in P (i.e. the maximal number of right and left bracket α-valencies which need to be added to P on the right (left) so that it became balanced. It is not difficult to prove that a potential P is balanced iff $\sum_{\alpha \in V} \Delta_\alpha(P) = 0$.

Let $\mathbf{c}$ be the projective core of the dependency calculus, consisting of the rules $\mathbf{L}$, $\mathbf{I}$ and $\mathbf{\Omega}$ and $\vdash_{\mathbf{c}}$ denote the provability relation in this sub-calculus. One of the key results of [6,7] is the following property of *projections independence* providing the abovementioned polynomial time parsing algorithm.

Theorem 1. *For a gCDG G with dictionary λ and a string x, $x \in L(G)$ iff there is $\Gamma \in \lambda(x)$ such that $\|\Gamma\|_l \vdash_{\mathbf{c}} S$ and $\|\Gamma\|_v$ is balanced.*

By the way, from this theorem it follows that the correctness of an assignment of categories to words in a sentence is recognized in real time.

Corollary 1. *For every sequence of types γ with distinguished constituents (i.e. without elimination ambiguity), the problem of provability $\gamma \vdash S$ in the dependency calculus is resolved in real time $|\gamma|$ using one stack and v counters, where v is the number of polarized valency names.*

[2] It doesn't prevent that gCDG explain the cross-serial dependencies [10].

4 Finite State Converters of F-Trees into D-Structures

First of all, we explain how the general f-trees are converted into DS. Next, we will adapt this conversion to the DP.

Definition 2. *Deterministic* FS transducer *(*DFS transducer*) is an automaton on f-trees* $\Gamma = (F, \Sigma, W, Q, q_0, \gamma, \delta)$, *where* $\mathcal{T}(F, \Sigma)$ *is the set of f-trees over* F *and* Σ, W *is a set of words,* Q *is a finite set of states,* q_0 *being the initial state,* γ *is a* lexical interpretation, *i.e. a function* $F \times Q \to W$ *and* δ *is a* transition function *defined as follows.*

Let $B = \{b_1, \dots, b_k\} \subseteq \Sigma$ *be a set of sorts and* $f/k \in F$ *be a functor,* $k \geq 0$. *Then* $\Pi(f, B, Q)$ *denotes the set of all strings* x *over* $(F \times Q) \cup (B \times Q)$ *such that* x *has at most one occurrence* (f, q) *for some* $q \in Q$ *and at most one occurrence of* (b, q) *for some* $q \in Q$ *for every sort* $b \in B$ [3]. *In these terms,* δ *is a function* $\{\delta : (f(b_1, \dots, b_k), q) \mapsto \beta \mid \beta \in \Pi(f, \{b_1, \dots, b_k\}, Q)\}$. *Elementary associations* $((f(b_1, \dots, b_k), q) \mapsto \beta) \in \delta$ *are called* transitions.

For an f-tree π *and a state* $q \in Q$, $\Gamma((\pi, q))$ *is defined by:*

$\Gamma((\pi_1, q_1) \dots (\pi_n, q_n)) = \Gamma((\pi_1, q_1)) \dots \Gamma((\pi_1, q_n))$.
$\Gamma((f, q)) = \gamma(f, q)$, *for all* $f \in F$.
$\Gamma((f(b_1 : \pi_1, \dots, b_k : \pi_k), q)) = \Gamma(\sigma(\delta(f(b_1, \dots, b_k), q)))$,
where σ *is the natural morphism* $\sigma((b_i, q)) = (\pi_i, q)$, $\sigma(\alpha) = \alpha$ *for* $\alpha \notin \{b_1, \dots, b_k\}$.

The transduction $\Gamma : \mathcal{T}(F, \Sigma) \to W^+$ *is defined by* $\Gamma(\pi) = \Gamma((\pi, q_0))$.

Definition 3. FT-DS converter $T = (\Gamma, G)$ *is a DFS transducer* Γ *whose lexical interpretation* γ_T *is coordinated with the lexicon* λ *of a gCDG* G *through a typing function* $type : Q \to \bigcup_{w \in W} \lambda(w)$ *as follows:* $\gamma_T(f, q) = w : type(q)$ *and* $type(q) \in \lambda(w)$. T *uniquely determines the following relation between the f-trees, the strings and the D-structures:* $T(\pi, x, D)$ *iff* $\Gamma(\pi) = w_1 : C_1 \dots w_m : C_m$, *where* $x = w_1 \dots w_m$, $C_1 \dots C_m \vdash S$ *and* D *is constructed in this proof.*

Language *of* T: $L(T) = \{x \in W^+ \mid \exists \pi, D(T(\pi, x, D))\}$.

Example 4. *It is not difficult to see that the following FT-DS converter* T_{abc} *generates the language* $L(T_{abc}) = \{a^n b^n c^n \mid n > 0\}$.

$$T_{abc} = \begin{cases} (h(1,2), q_0) \to (1, -)(2, S) & (f(1,2), S) \to (f, S)(1, +)(2, \backslash) \\ (g(1), -) \to (1, -)(g, -) & (f, S) \to b : [A \backslash S / C]^{\backslash A} \\ (g, -) \to a : [A \backslash A]^{\swarrow A} & (f(1,2), +) \to (f, +)(1, +)(2, \backslash) \\ (g_0, -) \to a : A^{\swarrow A} & (f, +) \to b : [B / C]^{\backslash A} \\ (e, \backslash) \to c : [B \backslash C] & (f_1(1), +) \to (f, +)(1, C) \\ (e, C) \to c : C \end{cases}$$

For instance, T_{abc} converts the f-tree $\pi^{(3)}$ in Fig. 6 into the typed string $(a : A^{\swarrow A})(a : [A \backslash A]^{\swarrow A})(a : [A \backslash A]^{\swarrow A})(b : [A \backslash S / C]^{\backslash A})(b : [B / C]^{\backslash A})(b : [B / C]^{\backslash A})(c : C)(c : [B \backslash C])(c : [B \backslash C])$. So $T_{abc}(\pi^{(3)}, a^3 b^3 c^3, D^{(3)})$, where $D^{(3)}$ is the DS in Fig. 4.

[3] This is a resource sensitive, "no copy" constraint.

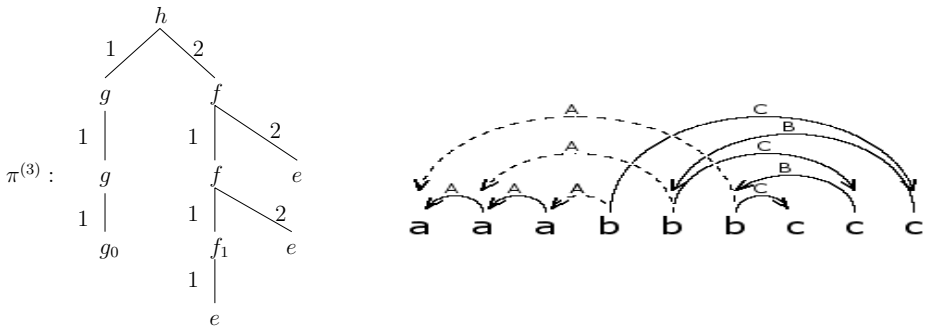


Fig. 6.

The FT-DS converters have the same weak generative power as gCDG in the following sense.

Theorem 2. *The family* $\mathcal{L}(FT-DS(RTL))$ *of languages* $T(L)$*, where* L *is a regular f-tree language and* T *is an FT-DS converter, coincides with* $\mathcal{L}(gCDG)$.

Proof scheme. 1. The inclusion $\mathcal{L}(gCDG) \subseteq \mathcal{L}(FT-DS(RTL))$ is evident. 2. The inverse inclusion follows from the established in [1] weak equivalence of gCDG to dependency structure grammars (DSG), a class of rule based generating dependency grammars. Due to the no copy constraint, for every given signature of functors F and sorts Σ, every FS-DT T applied to $\mathcal{T}(F,\Sigma)$ can be converted into a DSG Γ_T generating the language $T(\mathcal{T}(F,\Sigma))$. It remains to remark that for every FS-DT T and every finite automaton A on f-trees there is an FS-DT T_A such that $T(\Delta(A)) = T_A(\mathcal{T}(F,\Sigma))$, where $\Delta(A)$ is the set of f-trees accepted by A. The latter is proved using the classical construction of a finite automaton for the intersection of languages accepted by finite automata.

5 DP-DS Converters

DP-DS converters are a special kind of the FT-DS converters, where the DP are used in the place of arbitrary f-trees. It is significant that the set of all DP is a regular f-tree language.

In order to arrive at a syntax of DP-DS converters usable in practice, we code the DP and the converter transitions using HPSG-like feature structures. In particular, we use states with two substructures, a semantic one: DS[ILL, ASP, TIME], standing for the discourse status of the converted DP and a morpho-syntactic one: GS[LEFT, RIGHT, VALUE, PTN, GRCL (grammatical class), TNS (tense), NUM (number), PERS (person)] [4], standing for the features of the corresponding surface construction (see Fig. 7-17). Four selected features VALUE, LEFT, RIGHT and PTN in GS serve to define the dependency type $type(q)$. Namely, $type(q) = [\alpha_1\backslash...\backslash\alpha_l\backslash\gamma/\beta_1/.../\beta_r]^P$ if q.GS.LEFT $= \alpha_1\backslash...\backslash\alpha_l$, q.GS.RIGHT $= \beta_1/.../\beta_r$, q.GS.VALUE $= \gamma$ and q.GS.PTN $= P$.

[4] Both lists are open.

Like in HPSG, we make use of partially defined (i.e. generalized) structures and of their specializations (denoted $t_{gener} \preceq t_{spec}$). In particular, $t_1 = t_2 \preceq t$ means that t is a specialization of the most general unifier of t_1 and t_2. The unification is constrained by the hierarchy of primitive types. We also use default values. E.g., the default value PTN $= \varepsilon$ stands for the projective dependency types. Another example: for the *attributes* (sorts of *optional* arguments) listed in the situation argument frames under the name OPTARGS (see Fig. 7), the default value A $= \varepsilon$ stands for the absent arguments A. OPTARGS $= \emptyset$ means that all optional arguments are absent.

The transitions are also represented by partial feature structures and so can be consecutively specialized. The right hand side of a transition is represented as the value of the sort TO. The values of the sort PO stand for right hand side orderings.

A fragment of DP-DS transitions for English

Below, we present several transition schemes sufficient to generate the sentences in examples 1,2 from the corresponding DP. Fig. 7 presents the necessary argument frames.

$$dict\left[\begin{array}{ll}\text{KEY} & \text{open}\\ \text{PROFILE} & \textit{canon}\\ \text{SBJ} & [\text{TYPE } \mathbf{n_a}]\\ \text{OBJ} & [\text{TYPE } \mathbf{n}]\\ \text{INSTR} & [\text{TYPE } \mathbf{n}]\\ \text{OPTARGS} & \textit{list}\\ \text{VTYPE} & \mathbf{s_{eff}}\end{array}\right]$$

(a) Canonical profile of ⟪open⟫

$$dict\left[\begin{array}{lll}\text{KEY} & \text{open} & \\ \text{PROFILE} & \textit{dth} & \left[\begin{array}{ll}\text{NAME} & \text{decaus-ins}\\ \text{SHIFTS} & \left[\begin{array}{ll}\text{SBJ} & \left[\begin{array}{ll}\text{TYPE} & \mathbf{n_a}\\ \text{RANK} & \ominus\\ \text{SHIFT} & \emptyset\end{array}\right]\\ \text{OBJ} & \left[\begin{array}{ll}\text{TYPE} & \mathbf{n}\\ \text{RANK} & \odot\\ \text{SHIFT} & \text{OBJ}\\ \text{TSHIFT} & \mathbf{n}\end{array}\right]\\ \text{INSTR} & \left[\begin{array}{ll}\text{TYPE} & \mathbf{n}\\ \text{RANK} & \top\\ \text{SHIFT} & \text{SBJ}\\ \text{TSHIFT} & \mathbf{n}\end{array}\right]\end{array}\right]\\ \text{VALUE} & \left[\begin{array}{ll}\text{TYPE} & \mathbf{s_{eff}}\\ \text{TSHIFT} & \mathbf{s_{eff}}\end{array}\right]\end{array}\right]\\ \text{SBJ} & [\text{TYPE } \mathbf{n}] & \\ \text{OBJ} & [\text{TYPE } \mathbf{n}] & \\ \text{OPTARGS} & \textit{list} & \\ \text{VTYPE} & \mathbf{s_{eff}} & \end{array}\right]$$

(b) Diathesis ⟪$\text{dth}_{\text{decaus-ins}}$⟫ of ⟪open⟫

diΔδδΠoys

Two transition schemes in Fig. 8 serve to select the argument frame corresponding to the situation profile. Fig. 9 presents a transition scheme applied to the main situations with OBJ-argument in declarative DP. So it applies to both profiles of ⟪open⟫ in Fig. 7(a),(b). Its specific cases assign the axiom value type S to the main transitive verb (e.g. ⟪open⟫) in the surface form.

$$t_{(dcl,obj)}\begin{bmatrix}\textsc{sem} & dict\begin{bmatrix}\textsc{Obj} & x_{obj}\\ \textsc{vtype} & \mathbf{s}\end{bmatrix}\\ \textsc{state} & \begin{bmatrix}\textsc{ds} & \begin{bmatrix}\textsc{ill dcl}\end{bmatrix}\end{bmatrix}\\ \textsc{to} & \begin{bmatrix}\textsc{head} & \begin{bmatrix}\textsc{state} & \begin{bmatrix}\textsc{gs} & \begin{bmatrix}\textsc{value}\ S\end{bmatrix}\end{bmatrix}\end{bmatrix}\end{bmatrix}\end{bmatrix}\end{bmatrix}$$

Fig. 9. Transition scheme for declarative OBJ-argument situations

The scheme in Fig. 10 applies to the situations in semantical past time and neutral aspect corresponding in English to the surface simple past tense of main verbs. The transitions fitting this scheme assign to the situation's surface verb form $form(K, f)$ its left and right subordinates' subtypes, propagates the corresponding value types to SBJ-sort and OBJ-sort subplans and, at the same time, positions these subplans with respect to the verb. A similar scheme for semantical past time and perfect aspect is found in Fig. 11.

$$t_{PN}\begin{bmatrix}\textsc{sem} & dict\begin{bmatrix}\textsc{key}\ K\end{bmatrix}\\ \textsc{state} & \begin{bmatrix}\textsc{ds} & \begin{bmatrix}\textsc{asp} & \textsc{neutr}\\ \textsc{time} & \textsc{past}\end{bmatrix}\end{bmatrix}\\ \textsc{to} & \begin{bmatrix}\textsc{Sbj} & \begin{bmatrix}\textsc{state} & \begin{bmatrix}\textsc{gs} & \begin{bmatrix}\textsc{value}\ pred\end{bmatrix}\end{bmatrix}\end{bmatrix}\\ \textsc{head} & \begin{bmatrix}\textsc{state} & \begin{bmatrix}\textsc{gs}\ f & \begin{bmatrix}\textsc{left} & circ*\backslash pred\backslash\\ \textsc{right} & /circ*/dobj\\ \textsc{tns} & \textsc{PastSimple}\\ \textsc{GrCl} & \textsc{Vt}\end{bmatrix}\end{bmatrix}\\ \textsc{lex} & form(K,f)\end{bmatrix}\\ \textsc{Obj} & \begin{bmatrix}\textsc{state} & \begin{bmatrix}\textsc{gs} & \begin{bmatrix}\textsc{value}\ dobj\end{bmatrix}\end{bmatrix}\end{bmatrix}\\ \textsc{po} & \{\textsc{Sbj} < \textsc{head} < \textsc{Obj}\}\end{bmatrix}\end{bmatrix}\end{bmatrix}$$

Fig. 10. Transition scheme for situations in neutral aspect past time

The transition schemes in Fig. 12 and 13 apply to the situations in neutral (respectively, perfect) aspect and semantical past time if there is a MANNER-circumstantial. E.g., the transitions fitting the neutral aspect scheme place this argument between the SBJ-sort argument and the verb representing the situation if the circumstantial is an adverbial. Otherwise, it places this argument on the right of the OBJ-sort argument. Fig. 14 presents an example of a terminal transition scheme for adverbials without (optional) arguments. The cases of this scheme select the corresponding form and finalize the dependency type assigning $type(q) = q.\textsc{gs}.\textsc{value}$ (i.e. $q.\textsc{gs}.\textsc{ptn} = q.\textsc{gs}.\textsc{left} = q.\textsc{gs}.\textsc{right} = \varepsilon$). A similar scheme for adjective modifiers is found in Fig. 17.

$$t_{PP}\left[\begin{array}{ll}\textsc{sem} & dict\left[\textsc{k}\right.\end{array}\right.$$

$$t_{(\mathbf{c},noargs)}\left[\begin{array}{l}\text{SEM}\quad dict\left[\begin{array}{ll}\text{KEY} & K\\ \text{OPTARGS} & \emptyset\\ \text{VTYPE} & \mathbf{c}\end{array}\right]\\ \text{STATE}\left[\text{GS}\left[\text{VALUE}\ t\right]\right]\\ \text{TO}\quad\left[\text{HEAD}\left[\begin{array}{ll}\text{STATE} & \left[\text{GS}\ q_{\mathbf{c}}\left[\begin{array}{ll}\text{VALUE} & t\\ \text{LEFT} & \varepsilon\\ \text{RIGHT} & \varepsilon\\ \text{GrCl} & \text{Adv}\end{array}\right]\right]\\ \text{LEX} & form(K,q_{\mathbf{c}})\end{array}\right]\right]\end{array}\right]$$

Fig. 14. Transition scheme for adverbials

$$t_{(\mathbf{n},def)}\left[\begin{array}{l}\text{SEM}\ dict\left[\begin{array}{ll}\text{PROFILE} & \Uparrow_{ref}< K >_1\\ \text{VTYPE} & \mathbf{n}\end{array}\right]\\ \text{TO}\quad\left[\begin{array}{ll}\text{DET} & \left[\begin{array}{ll}\text{STATE} & \left[\text{GS}\ q_d\left[\begin{array}{ll}\text{LEFT} & \varepsilon\\ \text{RIGHT} & \varepsilon\\ \text{VALUE} & det\\ \text{GrCl} & \text{Det}_{def}\\ \text{PERS} & 3\\ \text{NUM} & sg\end{array}\right]\right]\\ \text{LEX} & form(\text{ART},q_d)\end{array}\right]\\ \text{HEAD} & \left[\begin{array}{ll}\text{STATE} & \left[\text{GS}\ q_N\left[\begin{array}{ll}\text{LEFT} & modif*\backslash det\backslash\\ \text{RIGHT} & /attr*\\ \text{GrCl} & \text{N}_{comm}\\ \text{NUM} & q_d.\text{NUM}\\ \text{PERS} & q_d.\text{PERS}\end{array}\right]\right]\\ \text{LEX} & form(K,q_N)\end{array}\right]\\ \text{PO} & \{\text{DET} < \text{HEAD}\}\end{array}\right]\end{array}\right]$$

Fig. 15. Transition scheme for determined nominators

$$t_{(\mathbf{n},Qual)}\left[\begin{array}{l}\text{SEM}\ dict\left[\text{OPTARGS}\left[\text{Qual}\left[\text{TYPE}\ \mathbf{q_{prop}}\right]\right]\right]\\ \text{TO}\quad\left[\begin{array}{ll}\text{HEAD} & h\\ \text{DET} & d\\ \text{Qual} & \left[\text{STATE}\left[\text{GS}\left[\text{VALUE}\ modif\right]\right]\right]\\ \text{PO} & \{\text{DET} < \text{Qual} < \text{HEAD}\}\end{array}\right]\end{array}\right]$$

Fig. 16. Transition scheme for qualifiers of determined nominators

$$t_{(\mathbf{q},noargs)}\left[\begin{array}{l}\text{SEM}\quad dict\left[\begin{array}{ll}\text{KEY} & K\\ \text{OPTARGS} & \emptyset\\ \text{VTYPE} & \mathbf{q}\end{array}\right]\\ \text{STATE}\left[\text{GS}\left[\text{VALUE}\ t\right]\right]\\ \text{TO}\quad\left[\text{HEAD}\left[\begin{array}{ll}\text{STATE} & \left[\text{GS}\ q_{Adj}\left[\begin{array}{ll}\text{VALUE} & t\\ \text{LEFT} & \varepsilon\\ \text{RIGHT} & \varepsilon\\ \text{GrCl} & \text{Adj}\end{array}\right]\right]\\ \text{LEX} & form(K,q_{Adj})\end{array}\right]\right]\end{array}\right]$$

Fig. 17. Transition scheme for adjective modifiers

In Fig. 18 is shown a computation of a DP-DS-converter whose transitions are determined by the transition schemes in Fig. 8-17. In the course of this computation, the sentence *The new key easily opened the door* is generated from the DP DP_{decaus} in Fig. 2 using five transitions:

$t_{decaus-ins} = t_{(dcl,obj)} = t_{PN} = t_{(PN,\mathbf{c_{mann}})} \preceq t_1$,
$t_{(\mathbf{n},Qual)} = t_{(\mathbf{n},def,K=KEY)} \preceq t_2$, $t_{(\mathbf{q},noargs,K=NEW)} = \preceq t_3$,
$t_{(\mathbf{c},noargs,K=EASILY)} = \preceq t_4$, $t_{(\mathbf{n},def,K=DOOR,\mathrm{OPTARGS}=\emptyset)} \preceq t_5$.

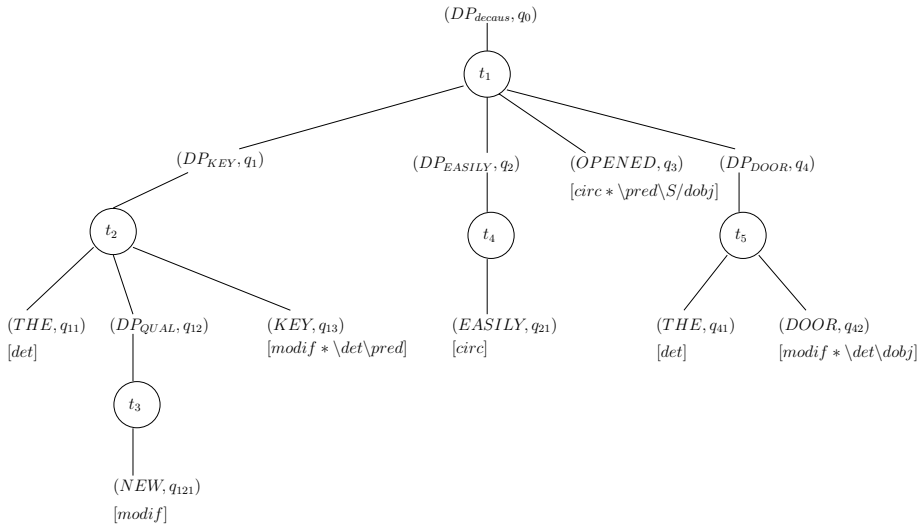


Fig. 18. Generation of *The ne ke ea il ened he d*

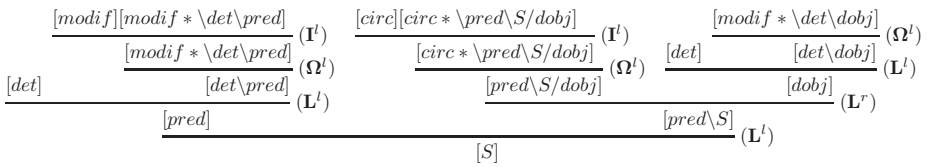


Fig. 19. A correctness proof

A proof of correctness of this computation is shown in Fig. 19. It is easy to see that in this proof is constructed the D-tree in Fig. 3(a).

Properties of DP-DS-conversion architecture. Our English example illustrates the following important properties of the DP-DS-conversion.

1. The transition schemes are either terminal (lexical or referential), or one argument, or they decompose a situation profile into constituents which fix the elimination order of local dependencies' subtypes.

2. The specific transition schemes are not guessed, but uniquely determined (through unification) by the substructures to which they apply.
3. The transitions do not copy or move a material. Unbounded discontinuous dependencies are established through dual valencies in correctness proofs.
4. In the place of a general proof search is used a real time type checking proof (see Corollary 1) witnessing for or against the feasibility of a DP when applied to the type sequence generated from it.

One can see that due to these properties the finite-state conversion and the type checking can be carried out in parallel in the course of DP composition.

6 Conclusion

This work is theoretical. It presents a new compositional generativistic architecture in which the surface syntax is defined in terms of dependency types, the meaning structures uniquely encode the arguments' scopes, surface forms and precedence, and where the meaning structure is converted into the corresponding surface structure using a deterministic fs-tree transducer. An experimental work is needed to make sure that this deterministic architecture can be realized through a system of thematic roles, semantic diatheses, lexical definitions and fs-transitions in a way that the resulting DP-DS-conversion of the set of input DP would be complete with respect to the underlying gCDG.

References

1. Béchet, D., Dikovsky, A., Foret, A.: Dependency structure grammars. In: Blache, P., Stabler, E.P., Busquets, J.V., Moot, R. (eds.) LACL 2005. LNCS (LNAI), vol. 3492, pp. 18–34. Springer, Heidelberg (2005)
2. Bos, J.: Predicate logic unplugged. In: Dekker, P., Stokhof, M. (eds.) Proc. of the 10th Amsterdam Colloquium, pp. 133–142 (1995)
3. Bresnan, J.: Optimal syntax. In: Dekkers, J., van der Leeuw, F., van de Weijer, J. (eds.) Optimal theory: Phonology, Syntax and Acquisition, pp. 334–385. Oxford University Press, Oxford (2000)
4. Chomsky, N.: The Minimalist Program. MIT Press, Cambridge, Massacusetts (1995)
5. Copestake, A., Flickinger, D., Malouf, R., Riehemann, S., Sag, I.: Translation using minimal recursion semantics. In: TMI95. Proc. of the 6th Int. Conf. on Theoretical and Methodological Issues in Machine Translation, Leuven, Belgium (1995)
6. Dekhtyar, M., Dikovsky, A.: Categorial dependency grammars. In: Moortgat, M., Prince, V. (eds.) Proc. of Intern. Conf. on Categorial Grammars, pp. 76–91, Montpellier (2004)
7. Dekhtyar, M., Dikovsky, A.: Generalized categorial dependency grammars (2007) Submitted paper available from `http://www.sciences.univ-nantes.fr/info/perso/permanents/dikovsky/`
8. Dikovsky, A.: Linguistic meaning from the language acquisition perspective. In: FG 2003. Proc. of the 8th Intern. Conf. Formal Grammar 2003, Vienna, Austria, pp. 63–76 (August 2003)

9. Dikovsky, A.: Dependencies as categories. In: Recent Advances in Dependency Grammars. COLING'04 Workshop, pp. 90–97 (2004)
10. Dikovsky, A.: Multimodal categorial dependency grammars (Submitted paper)
11. Dikovsky, A., Smilga, B.: Semantic roles and diatheses for functional discourse plans. In: MTT 2005. Proc. of the 2nd International Conference Meaning-Text Theory, pp. 98–109 (2005)
12. Dowty, D.R.: Thematic proto-roles and argument selection. Language 67, 547–619 (1991)
13. Fillmore, C.: Frame semantics. In: Linguistic Society of Korea (ed.) Linguistics in the Morning Calm, pp. 111–138, Seoul, Hanshin (1982)
14. Fox, C., Lappin, S.: Underspecified interpretations in a curry-typed representation language. Journal of Logic and Computation 15(2), 131–143 (2005)
15. Grimshaw, J.: Projection, heads and optimality. Linguistic Inquiry 28, 373–422 (1997)
16. Hovav, M.R., Levin, B.: Deconstructing the thematic hierarchy. In: 38th Seoul Linguistics Forum, December 10-11, Seoul National University, Seoul, Korea (2004)
17. Kamp, H., Reyle, U.: From Discourse to LOgic: An introduction to modeltheoretic semantics, formal logic and Discourse Representation Theory. Kluwer Academic Publishers, Dordrecht (1993)
18. Padučeva, E.V.: Dynamic models in lexical semantics. Jazyki slavjanskoj kul'tury, Moscow (Russ.) (2004)
19. Kaplan, R., Bresnan, J.: Lexical functional grammar: A formal system for grammatical representation. In: Bresnan, J.W. (ed.) The Mental Representation of Grammatical Relations, pp. 173–281. MIT Press, Cambridge (1982)
20. Reyle, U.: Dealing with ambiguities by underspecification. Journal of Semantics 10(2), 123–179 (1993)
21. Sgall, P., Hajičová, E., Panevová, J.: The Meaning of the Sentence in its Semantic and Pragmatic Aspects. Academia, Dordrecht/Reidel/Prague (1986)
22. Sgall, P.: A dependency based specification of topic and focus ii. SMIL. Journal of Linguistic Calculus (1-2) 110–140 (1980)
23. Stabler, E.: Derivational minimalism. In: Retoré, C. (ed.) LACL 1996. LNCS (LNAI), vol. 1328, pp. 68–95. Springer, Heidelberg (1997)
24. V.-Shanker, K., Weir, D.J.: The equivalence of four extensions of context-free grammars. Mathematical Systems Theory 27, 511–545 (1994)