Parsing with polarities

Guy Perrier

LORIA, Université Nancy2

Abstract

Interaction Grammars are a grammatical formalism based on the notion of polarities. Polarities express the resource sensitivity of natural languages by modelling the distinction between saturated and unsaturated syntactic structures. A sentence's syntactic composition is represented as an "electrostatic" process guided by the neutralization of polarities. At the heart of the formalism, there is also the notion of underspecification: syntactic structures are not completely specified trees but tree descriptions. Then, parsing appears as a process of building tree description models. Semantics is represented at a level independent of the syntactic level, but based on the same notions of polarity and description with a difference: descriptions are not tree descriptions but directed acyclic graph (DAG) descriptions. The interface between the two levels is realized in a flexible way by a function that links every syntactic node to at most one semantic node.

Key words: categorial grammars, polarities, tree descriptions

Introduction

The originality and the advantage of Categorial Grammars (CG) with respect to other linguistic formalisms are that they use the resource sensitivity of natural languages as a principle for syntactic composition. A way of highlighting this characteristic is to use polarities: partially specified syntactic trees are decorated with polarities that express a property of non saturation; a positive node represents an available grammatical constituent whereas a negative node represents an expected grammatical constituent; negative nodes seek to merge with positive nodes of the same type and this mechanism of neutralization between opposite polarities drives the composition of syntactic trees to produce saturated trees in which all polarities have been neutralized.
This notion is closely akin to an ancient idea of L. Tesnières [1] and K. Adjukiewicz [2]: a sentence is viewed as a molecule with its words as the atoms; every word is equipped with a valence which expresses its capacity of interaction with other words. This idea was not exploited directely in computational

linguistics until recently : to our knowledge, A. Nasr was the first to propose a formalism using polarized structures [3]; then, we introduced a first version of Interaction Grammars in the framework of linear logic [4]. This version covers only the syntax of natural languages contrary to the version that we will present now, which covers both the syntax and the semantics of natural languages.

By introducing Interaction Grammars (IG), we highlighted the fundamental mechanism of neutralization between polarities underlying CG in a more refined way, because polarities are attached to the features used for describing constituents and not to the constituents themselves — but the essential difference lies in the change of framework: CG are usually formalized in a generative deductive framework, the heart of which is the Lambek Calculus, whereas IG are formalized in a model-theoretic framework. A particular interaction grammar appears as a set of constraints, and parsing a sentence with such a grammar reduces to solving a constraint satisfaction problem. G. K. Pullum and B. C. Scholz highlighted the advantages of this change of framework [5]. Here, we are especially interested in some of these advantages:

- syntactic objects are tree descriptions which combine independent elementary properties in a very flexible way to represent families of syntactic trees;
- underspecification can be represented in a natural way by tree descriptions;
- partially well-formed sentences have a syntactic representation in the sense that, even if they have no complete parse trees, they can be characterized by tree descriptions.

The notion of tree description, which is central in this approach, was introduced by M. Marcus, D. Hindle and M. Fleck to reduce non-determinism in the parsing of natural languages [6]. It was taken again by K. Vijay-Shanker to represent the adjoining operation of TAG in a monotone form [7]. Then, it was studied systematically under a mathematical angle [8] and it gave rise to new grammatical formalisms [9,10].
If model theory provides a declarative framework for IG, polarities provide a step by step operational method for building models of tree descriptions: partially specified trees are superposed under the control of polarities; some nodes are merged in order to neutralize their polarities and the process ends when all polarities are neutralized. At that time, the resulting description represents a completely specified syntactic tree. The ability of the formalism to superpose trees is very important for its expressiveness whereas the control of superposition by polarities is interesting for computational efficiency.

In natural languages, syntax is a means for accessing semantics and a linguistic formalism cannot deal only with syntax by ignoring semantics. The semantics usually associated with CG is Montague's semantics [11]. Every sentence is interpreted by a formula in higher order logic, which is automatically deduced

from the syntactic structure of the sentence and from the interpretations of the words of the sentence. This rigidity is a major defect of the formalism: in a certain sense, the whole semantic representation must be included in the syntactic representation. The latter must be complicated artificially in order to express specific semantic phenomena like quantifier scoping. As a consequence, a sentence gets as many artificial parse trees as interpretations, even if it has a unique actual parse tree.

In IG, semantics is integrated at a level independent from the syntactic representation. The fundamental difference with the model of CG just mentioned is that the semantic level is relatively autonomous with respect to the syntactic level: the mechanism of linking between syntax and semantics is as little constraining as possible. Nevertheless, the semantic representation is based on the same notions of polarity and underspecified structure as the syntactic representation, but with a difference: at the syntactic level, we manipulate syntactic trees and syntactic tree descriptions, whereas, at the semantic level, we manipulate semantic DAGs and semantic DAG descriptions but with the same model-theoretical interpretation: a DAG description represents a set of DAGs, each DAG being a model of the description in the shape of a network of predicate-arguments relations. Polarities are also used to control the building of models of DAG descriptions.

The layout of the paper is as follows:

- Section 1 presents the syntactic level of the formalism centered around the notion of syntactic tree description;
- Section 2 presents the semantic level based on the notion of semantic DAG description;
- in Section 3, we show how the syntactic level interacts with the semantic level in the process of parsing through a linking function;
- in Section 4, we compare IG with the most closely related formalisms.

1 The level of the syntactic tree descriptions

1.1 The form of syntactic tree descriptions

A syntactic tree description is a set of nodes and parenthood, dominance and precedence relations between these nodes. Nodes represent syntactic constituents and relations express dependencies between these constituents. The morpho-syntactic properties of these constituents are described with feature structures. In this way, a tree description can be viewed as a specification representing a set of syntactic trees and each of these trees can be viewed as

a model of the specification.

The bottom part of figure 1 represents a syntactic tree description D_{syn} associated by a lexicon to the sentence *tous en connaissent une application*. This sentence, which means *they all know an application of it*, is said in a context where people are discussing about a given parser. Its grammatical interest lies in the clitic *en (of it)*, which joins on to the transitive verb *connaissent (know)*, not to provide its object but to provide a complement of its object *une application (an application)*: an application is an application of the parser in question.

On figure 1, every node appears as a rectangle divided in two parts: its head contains the name of the node and its body contains the associated feature structure. Nodes are linked together by four kinds of relations:

immediate dominance relations: $N > M$ means that M is an immediate sub-constituent of the constituent N, which is graphically represented by a continuous line; moreover, when a rectangle has a down square bracket at its bottom, this means that the number of sub-constituents of the corresponding constituent is fixed; for instance, the bracket associated with the node *cl*2 represents the relation $cl2 > \{subj,\ v\text{-}max2,\ obj2\}$, which means that *cl2* has exactly three daughters: *subj*, *v-max2* and *obj2*; in other words, the clause *cl2* is composed of three constituents: the maximal projection *v-max2* of a verb (the verb possibly completed with clitics, negation, adverbs ...), its subject *subj* and its object *obj2*;

underspecified dominance relations: $N >^* M$ means that the constituent N includes another constituent M at a more or less deep level, which is graphically represented by a dashed line; at the limit, the two constituents can be identified; such relations are used for expressing long distance dependencies and possibilities of applying modifiers; both long distance dependencies and applications of modifiers are bounded by constraints, which are expressed by feature structures labelling dominance relations; then, dominance relations take the form $N >^* [f_1 = v_1, \ldots, f_n = v_n]M$ with the following meaning: every node that dominates N and that is dominated by M (in a large sense) has its feature structure that unifies with $[f_1 = v_1, \ldots, f_n = v_n]$ [1]; for instance, the relation $v\text{-}max2 >^* [cat = v]$ *v-min* means that all intermediate constituents between the bare verb *v-min* and its maximal projection *v-max2* have the category *verb*; in other words, the bare verb can be only transformed by modifiers;

immediate precedence relations: $N \prec M$ means that the constituent M precedes the constituent N immediately in the linear order of the sentence, which is graphically represented by continuous arrows; for instance, the relations $subj \prec v\text{-}max2$ and $v\text{-}max2 \prec obj2$ express the canonical SVO

[1] See later for a more precise definition of the unification of polarized feature structures.

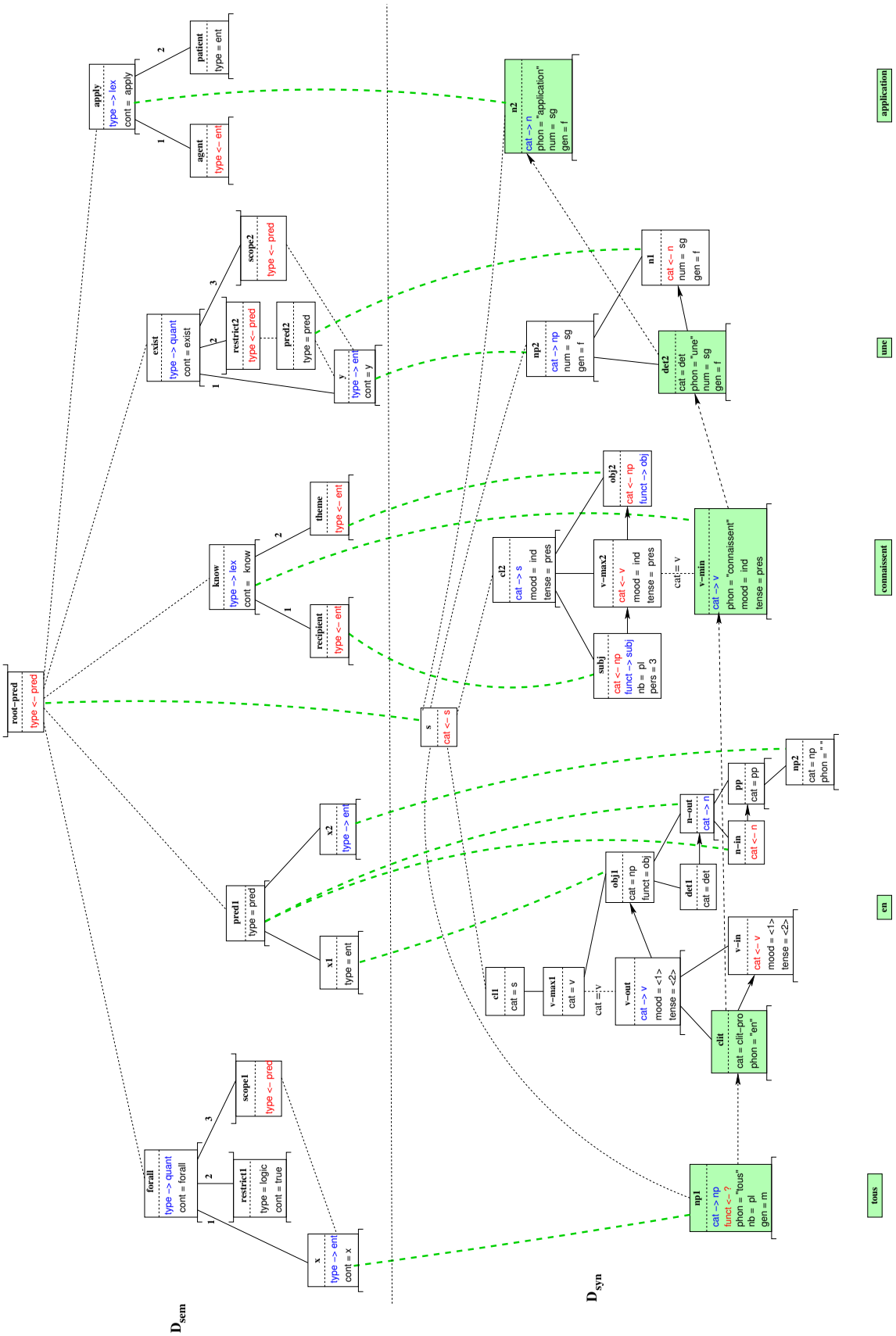

Fig. 1. SynSem description provided by a lexicon for the sentence *tous en connaissent une application*

order;

underspecified precedence relations: $N \prec^+ M$ means that the constituent M precedes the constituent N in the linear order of the sentence but they cannot be identified; such a relation is represented graphically by a dashed arrow; on figure 1, the syntactic nodes represented by grey rectangles correspond to the words of the sentence and word order in the sentence is expressed by underspecified precedence relations between these nodes [2].

Except immediate precedence relations, we take again the types of relations introduced by [7] but this system of relations is open and it can be enriched to express specific constraints of a language.

Now, if we look at the feature structures associated with the description nodes, we find again the classical notion of feature structure at the root of unification grammars, but with the additional notion of polarity. The originality of IG is to express the distinction between saturated and unsaturated syntactic structures with polarized features. This notion of polarized feature is the continuation of the fundamental idea of CG: capturing the resource sensitivity of natural languages. While a morpho-syntactic feature is usually associated with values in feature-value pairs, a feature in IG is associated with values and polarities in triples (f, p, v) such that:

- f is a feature belonging to a given set *Feat*;
- p is a polarity belonging to the set $\{\rightarrow, \leftarrow, =, \leftrightarrow\}$, reflecting the fact that a feature is positive, negative, neutral or saturated; positive features represent available resources, negative features expected resources and neutral features linguistic properties that do not behave like consumable resources; saturated features result from the combination of positive features with negative features; the corresponding triples are respectively written in the simplified form $f \rightarrow v$,$f \leftarrow v$, $f = v$, $f \leftrightarrow v$; for instance, the constituent *np1* representing the pronoun *tous* is associated with different kinds of polarized features; features $cat \rightarrow np$ and $funct \leftarrow ?$ [3] express that *tous* can provide a noun phrase which is waiting for a syntactic function which is not yet determined;
- v is a finite disjunction $(v_1|\ldots|v_n)$ of atoms v_i belonging to the finite domain D_f of values associated with f; to express the sharing of values by several features, v may include an index i and then it is written $\langle i \rangle (v_1|\ldots|v_n)$.

Formally, a syntactic tree description is defined on a signature constituted of

[2] Word order is not express by immediate precedence relations because of the possible presence of nodes with an empty phonological form between nodes representing full words.

[3] The symbol "?" is an abbreviation for the disjunction of all values of the corresponding domain.

a set *Nodes* of node identifiers and a set *Feat*[4] of features as follows:

Definition 1.1 *A syntactic tree description is a finite set of relations in one of the following form:*

- $(N : (f, p, v_1|\ldots|v_n))$ *or* $(N : (f, p, \langle i\rangle(v_1|\ldots|v_n)))$ *with* $N \in Nodes$, $f \in Feat$, $p \in \{\rightarrow, \leftarrow, =, \leftrightarrow\}$, $v_k \in D_f$ *and* $i \in \mathbb{N}$;
- $M > N$, $M \prec N$ *or* $M \prec^+ N$, *with* $M, N \in Nodes$;
- $M >^* [f_1 = (v_1^1|\ldots|v_1^{p_1}), \ldots, f_n = (v_n^1|\ldots|v_n^{p_n})]$ N *with* $M, N \in Nodes$, $f_i \in Feat$ *and* $v_i^k \in D_{f_i}$.

The diagram in the bottom part of figure 1 is a graphical representation of a syntactic tree description D_{syn} which can be presented as a set of relations according to the definition above.

1.2 Models of syntactic tree descriptions

A tree description is a way of representing a set of trees in a compact way, each tree being a model of this description. For defining the models of a syntactic tree description, we have first to define the form of such models, that of *syntactic trees*.

Definition 1.2 *A syntactic tree is an ordered tree*[5]*, the nodes of which are labelled with feature structures in the form* $[f_1 = v_1, \ldots, f_n = v_n]$ *with* $v_i \in D_{f_i}$.

The form of models being fixed, we must now establish their relationship with descriptions.

Definition 1.3 *A model of a syntactic tree description D is a pair (T, I) in which T is a syntactic tree and I is an interpretation function which maps every node of D to a node of T such that:*

[4] Every element of *Feat* is a pair (f, D_f) constituted of a feature f and a finite domain D_f of constant values.

[5] In an ordered tree, every node has the set of its daughters totally ordered. We can define syntactic trees in a more sophisticated way: the set of the daughters of every node is partitioned into two parts, the subset of the nodes with a full phonological form, which is totally ordered, and the subset of the nodes with an empty phonological form, which is not ordered. This definition is justified by the fact that we use traces to mark the "normal" place of constituents that are not in a canonical setting. Traces are nodes with an empty phonological form and if a constituent contains several traces, we want to avoid to get different models corresponding to different order between these traces. In this paper, we use a total order between all daughters of any node and we avoid the previous problem by choosing a canonical order between traces.

- *if* $(N : (f, p, v_1|\ldots|v_n)) \in D$ *or* $(N : (f, p, \langle i\rangle(v_1|\ldots|v_n))) \in D$, *then there exists some* k *for which* $(f = v_k)$ *belongs to the feature structure of* $I(N)$;
- *if* $(N_1 : (f, p_1, \langle i\rangle w_1) \in D$ *and* $(N_2 : (f, p_2, \langle i\rangle w_2)) \in D$, *then the same feature* $f = v$ *belongs to the feature structures of* $I(N_1)$ *and* $I(N_2)$;
- *if* $(M > N) \in D$, *then* $I(N)$ *is a daughter of* $I(M)$ *in* T;
- *if* $(M > \{N_1, \ldots, N_p\}) \in D$, *then* $I(N)$ *has* p *daughters in* T, *which are* $I(N_1)$, ..., $I(N_p)$;
- *if* $(M >^* [f_1 = (v_1^1|\ldots|v_1^{p_1}), \ldots, f_n = (v_n^1|\ldots|v_n^{p_n})]\ M) \in D$, *then for every node* N' *of* T *that is a descendant of* $I(M)$ *and an ascendant of* $I(N)$ *in a large sense, and for every* f_i, *there exists some* k *for which* $(f_i = v_i^k)$ *belongs to the feature structure of* $I(N')$;
- *if* $(M \prec N) \in D$, *then* $I(N)$ *precedes* $I(M)$ *immediately in* T [6] ;
- *if* $(M \prec^+ N) \in D$, *then* $I(N)$ *precedes* $I(M)$ *in* T.

The syntactic tree description D'_{syn} in the bottom part of figure 2, in which all occurrences of the polarity $\leftrightarrow$ have been replaced by $=$ represents a completely specified ordered tree which is a model of the description D_{syn} of figure 1. The interpretation function is represented implicitly: the head of every node of D'_{syn} include a concatenation of the names of its antecedents in D_{syn}.

1.3 Minimal and neutral models of syntactic tree descriptions

For a syntactic tree description expressing the specification of a whole sentence, the general notion of model of a description is not sufficient for two reasons:

(1) if a description has a model, then it has an infinite number of models and it is necessary to exhibit privileged models representing the syntax of the sentence; such models have a property of *minimality*: informally, a model is minimal when it adds a minimum of information with respect to the description;
(2) polarities are not taken into account in the general definition of model; but, these are introduced precisely to express the distinction between saturated and unsaturated syntactic structures; saturated structures represent the syntax of grammatical sentences and models must capture this notion of saturation; it is done with the property of *neutrality*.

Let us define these two properties of models more precisely.

Definition 1.4 *A model* (T, I) *of a description* D *is* minimal *if every immediate dominance relation in* T *interprets an immediate dominance relation in*

[6] In an ordered tree, the relation of immediate precedence naturally follows from the ordering between the daughters of every node.

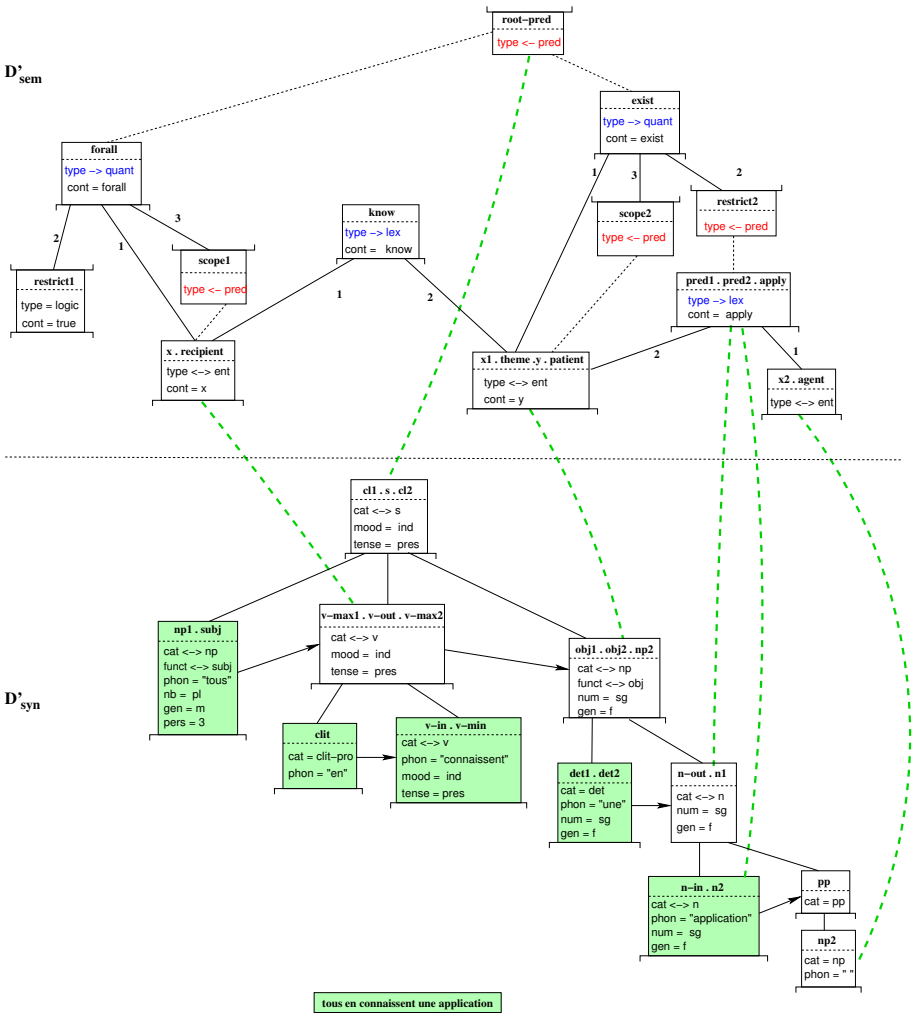

Fig. 2. SynSem description resulting from the parsing of the sentence *tous en connaissent une application*

D and if every node of T with a feature f interprets a node of D with a feature f.

Definition 1.5 *A model (T, I) of a description D is* neutral *if for every feature $f \in Feat$ and for every node N of T, the set $\{(N_1 : (f, p_1, v_1)), \ldots, (N_p : (f, p_p, v_)\}$ of all relations $(N_i : (f, p_i, v_i)) \in D$ such that $I(N_i) = N$, has one of the following properties:*

- *for any i, p_i is equal to $=$;*
- *for any i, p_i is equal to $=$ except one element p_k which is equal to $\leftrightarrow$;*
- *for any i, p_i is always equal to $=$ except two elements p_h and p_k which are equal to $\rightarrow$ and $\leftarrow$.*

This property of neutrality defines the semantics of polarities. We can design another system of polarities with another semantics. For instance, S. Kahane defines a more sophisticated system of polarities which he uses to revisit most

classical grammatical formalisms from a new point of view [12].
Minimal and neutral models of syntactic tree descriptions will be called *valid models* in the rest of the paper. We verify that D'_{syn} in the bottom part of figure 2 is a valid model of the syntactic tree description D_{syn} of figure 1.

From the notion of valid model of a syntactic tree description, we deduce the notions of *refinement* and *equivalence* of syntactic tree descriptions.

Definition 1.6 *A syntactic tree description D_1 is a refinement of a syntactic tree description D_2 if, for every valid model (T, I_1) of D_1, there exists a valid model (T, I_2) of D_2.*
Two syntactic tree descriptions are equivalent if they are refinements of each other.

The empty description $\emptyset$ plays a particular role because it has no valid model. Every description that is equivalent to the empty description is said to be *inconsistent*. Other descriptions play a particular role: descriptions that contain no positive and no negative features are called *saturated descriptions*.

1.4 Computation of valid models by neutralization of polarities

The definition of valid models of syntactic tree descriptions allows to verify if a given syntactic tree is a valid model of a given syntactic tree description but it does not provide any way of computing such a model from a given description. For this, we use an operation, called *neutralization*.

Definition 1.7 *Let D be a description containing two relations $(N_1 : f \rightarrow v_1)$ and $(N_2 : f \leftarrow v_2)$. A neutralization of these two polarized features consists in identifying N_1 and N_2 in D. If we denote the resulting description by D', we write: $D \overset{N_1 \overset{f}{=} N_2}{\longrightarrow} D'$. We denote the reflexive and transitive closure of this relation between D and D' by $\overset{*}{\longrightarrow}$.*

This operation stems from the notion of neutral model of a syntactic tree description. In such a model, every node of the description that bears a positive or a negative feature must be merged with a node of the description that bears a dual feature. From this, follows the computation principle for valid models of syntactic tree descriptions: iterating the operation of neutralization to build saturated descriptions. This principle is justified by the following proposition:

Proposition 1.1 *For any syntactic tree descriptions D_1 and D_2 such that $D_1 \overset{*}{\longrightarrow} D_2$, D_2 is a refinement of D_1.*
If a syntactic tree description D_1 has a valid model (T, I_1), there exists a saturated description D_2 such that $D_1 \overset{*}{\longrightarrow} D_2$ and D_2 has a valid model (T, I_2).

As a consequence, in the iteration of the operation of neutralization, two cases happen:

- we obtain a non saturated description and any neutralization is no longer possible: we can conclude that the description is not consistent;
- the process ends with a saturated description and we can conclude that every valid model of this description corresponds to a valid model of the initial description.

Proposition 1.1 guarantees the correctness and the completeness of the method that consists in iterating neutralizations until we obtain a saturated description for computing valid models of a given description.

An important effect of the operation of neutralization is that it allows the resulting description to be simplified, that is, to be replaced with a simpler but equivalent description. There are two kinds of simplification: the first one concerns features and the second one concerns structural relations between nodes. To describe the simplifications related to features, we need to define an operation of addition between polarities.

Definition 1.8 *The sum $p + q$ of two polarities p and q is given by the following table, where an empty square means that the sum does not exist:*

	$\leftarrow$	$\rightarrow$	$=$	$\leftrightarrow$
$\leftarrow$		$\leftrightarrow$	$\leftarrow$	
$\rightarrow$	$\leftrightarrow$		$\rightarrow$	
$=$	$\leftarrow$	$\rightarrow$	$=$	$\leftrightarrow$
$\leftrightarrow$			$\leftrightarrow$	

Unification between two feature values, denoted $\sqcup$, is defined in a classical way as the intersection of disjunctions of atomic values. Moreover, if both values are indexed, the indices are identified. Then, simplifications related to features are summarized by the following proposition:

Proposition 1.2 *Let D be a syntactic tree description that contains two relations $(N : (f, p_1, v_1))$ and $(N : (f, p_2, v_2))$. If the sum of p_1 and p_2 does not exist or if v_1 and v_2 are not unifiable, then D is inconsistent or else we obtain a description equivalent to D by replacing the two relations with $(N : (f, p_1 + p_2, v_1 \sqcup v_2))$.*

For instance, consider the description D_{syn}. There is only one possibility of neutralizing the feature $(np1 : cat \rightarrow np)$ without producing inconsistency: by means of the feature $(subj : cat \leftarrow np)$, and we obtain a new description

$D_{1_{syn}}$, where the nodes $np1$ and $subj$ are merged in a unique node $np1.subj$ [7]. Then by simplification, we replace the features $(np1.subj : cat \rightarrow np)$ and $(np1.subj : cat \leftarrow np)$ with the feature $(np1.subj : cat \leftrightarrow np)$. In the same way, we replace the features $(np1.subj : funct \leftarrow ?)$ and $(np1.subj : funct \rightarrow subj)$ with the feature $(np1.subj : funct \leftrightarrow subj)$ and we obtain a description $D'_{1_{syn}}$.

We can also establish simplification rules related to the structure of a description. You can find examples of such rules in [13]. Here, we will merely give an illustration of these rules with our example. We try to continue the process of neutralization from the description $D'_{1_{syn}}$. The syntactic function of the clitic *en* is represented by the partial underspecified tree rooted at the node *cl1*. As a clitic, it acts as a modifier of a verb *v-in* put on its right to produce a cliticized verb *v-out*. At the same time, it acts on the object *obj1* of the verb. It provides a complement to the head of *obj1* which is represented by a trace *np2*; this trace is a constituent with an empty phonological form. There is a unique possibility of neutralizing the feature *(v-in:* $cat \leftarrow v$): by means of the feature *(v-min:* $cat \rightarrow v$). We merge the nodes *v-in* and *v-min* and we obtain a new description $D_{2_{syn}}$, which we try to simplify. Since there is a unique path from a node to any descendant in a tree and since there is a constraint $[cat = v]$ on the dominance relations between *v-max1* and *v-out* as well as between *v-max2* and *v-min*, the nodes *v-max1* and *v-max2* merge necessarily. As a consequence, the nodes *cl1* and *cl2* merge also. Finally, since the node *cl2* has exactly three daughters *subj*, *v-max2* and *obj2*, *obj1* and *obj2* merge also. On this example, we remark that a simple neutralization triggers the superposition of two partially specified trees. In the formalism of IG, syntactic composition appears as a process of superposition between underspecified trees under the control of polarities. The ability of superposing trees partially is crucial for the expressiveness of the formalism. Under this angle, IG are closer to unification grammars than tree grammars like TAG. Our example is very difficult to represent in TAG because TAG only allows a tree to be inserted inside another tree.

If we continue the neutralization process on our example, after two neutralizations, we obtain the description D'_{syn} of the figure 2. If we do not take the order between neutralizations into account, the process is completely deterministic and the result, D'_{syn}, is completely specified so that it has a unique valid model, which is also a valid model of D_{syn}.

Generally, the process is not deterministic and the resulting description, even if it is saturated, is not necessarily completely specified and may still include

[7] This operation can be also presented as a neutralization between the features $(np1 : funct \leftarrow ?)$ and $(subj : funct \rightarrow subj)$. Besides, we denote the merged nodes with the concatenation of their names for greater convenience but this is not meaningful.

underspecified dominance relations between nodes. But we can hope that, in realistic applications residual underspecification will be not very important.
The non-determinism related to the choice of neutralizations adds to the non-determinism on the choice of the lexical entries but we can design original methods based on polarities to reduce these two sources of non-determinism [13,14].

2 The level of the semantic DAG descriptions

We do not propose a concrete semantics for IG but rather a framework for representing various concrete semantics. In the following, to illustrate our purpose with an example, we choose the first order predicate calculus as a concrete semantics.
Semantics is represented at a level independent of the syntactic level but with the same principles: the use of the notion of description to express underspecification and the use of the notion of polarity to control the composition of underspecified structures. The main difference is that we now have to represent semantic dependencies between objects in the form of predicate-argument relations. Therefore, completely specified semantic structures will not be ordered trees but DAGs: precedence between nodes is no longer meaningful and a semantic object can be the argument of several predicates.
Since the formalization of the semantic level is similar to that of the syntactic level, we merely describe the original aspects in details.

2.1 The form of semantic DAG descriptions

A semantic DAG description is a set of parenthood and dominance relations between nodes. Parenthood and dominance respectively express predicate-argument and scope relations between semantic objects which are either predicates or individuals.
As at the syntactic level, we can express constraints on the number of daughters of a node N with the relation $N > \{N_1, \ldots, N_p\}$ but, since a node may have several parents, we can also constrain the number of parents with the relation $\{N_1, \ldots, N_p\} > N$. Graphically, this new constraint is represented by an up square bracket at the top of the rectangle corresponding to N.
At the syntactic level, the syntactic function of a daughter of a node is indicated by a feature *funct* attached to this daughter. At the semantic level, since the same object can play different thematic roles in different predicates, the thematic roles of a predicate are differentiated with integers put on the edges representing the predicate-argument links.
The semantic properties of predicates and individuals are described with fea-

ture structures. Features are polarized with the same system of polarities and the same meaning as at the syntactic level [8].
For instance, the top part of figure 1 represents a semantic DAG description D_{sem} associated by a lexicon to the words of the sentence *tous en connaissent une application*. From a semantic point of view, the sentence has two readings because of the scope ambiguity between the two quantifiers *tous* and *une*. In D_{sem}, a quantifier is represented by a predicate with three arguments: the individual involved in the quantification and two predicates which express the restriction and the scope of the quantifier. For sake of simplification, we consider only two semantic features: *type* expresses the type of the object and can take the values *ent* (entity or individual), *lex* (lexical predicate), *quant* (quantifier) or *logic* (logical connective or constant) [9]; *cont* (contents) expresses the value of an object. For *tous*, we consider that there is no restriction, which is expressed by the fact that its restriction *restrict1* is the logical constant *true*. Remark that there is a unicity constraint for the parents of *restrict1* and *scope1* but not for x which can be an argument of several predicates.

2.2 Minimal and neutral models of semantic DAG descriptions

At the syntactic level, a syntactic tree description represents a set of syntactic trees. In a similar way, a semantic DAG description represents a set of semantic DAGs, which constitute its models. A semantic DAG is a DAG whose nodes are labelled with semantic feature structures, all features having the form $f = v$ with $v \in D_f$.
Definition 1.3 can be easily adapted to characterize the notion of model of a semantic DAG description. Then, definitions 1.4 and 1.5 are usable without any change to characterize minimal and neutral models of semantic DAG descriptions. Such models are also called valid models of semantic DAG descriptions.
Description D_{sem} has two minimal and neutral models, the semantic DAGs represented on figures 3 and 4 [10], which correspond to the two possible relationships between the scopes of the quantifiers . Since we have chosen the first order predicate calculus as the concrete semantics, every DAG is a geometrical representation of a first order formula. The formula can be deduced automatically from the corresponding DAG if we have an interpretation of every quantifier. The interpretation of *forall* is $\forall x(restrict1 \Rightarrow scope1)$ and the interpretation of *exist* is $\exists y(restrict2 \wedge scope2)$. Therefore, the DAG of figure 3 represents the formula $\forall x(true \Rightarrow \exists y(apply(x_2, y) \wedge know(x, y))$,

[8] The system of polarities could be relaxed to allow n positive features f to be neutralized by n negative features f at the same node. In this paper, $n = 1$.
[9] The value *pred* is an abbreviation of *lex|quant|logic*.
[10] To be in accordance with the definition of model, we have to replace all occurrences of the polarity $\leftrightarrow$ in figures 3 and 4 with the polarity $=$.

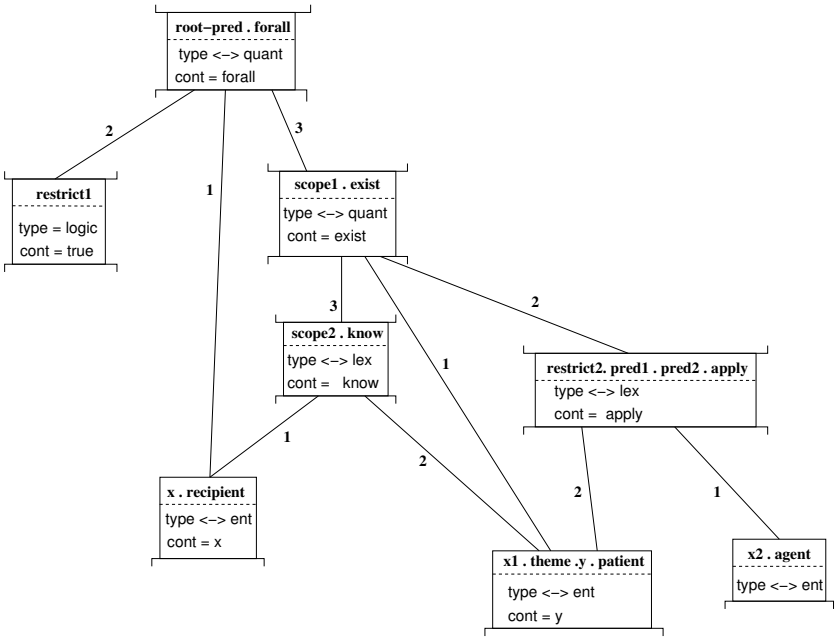

Fig. 3. Semantic DAG representing the first interpretation of the sentence *tous en connaissent une application*

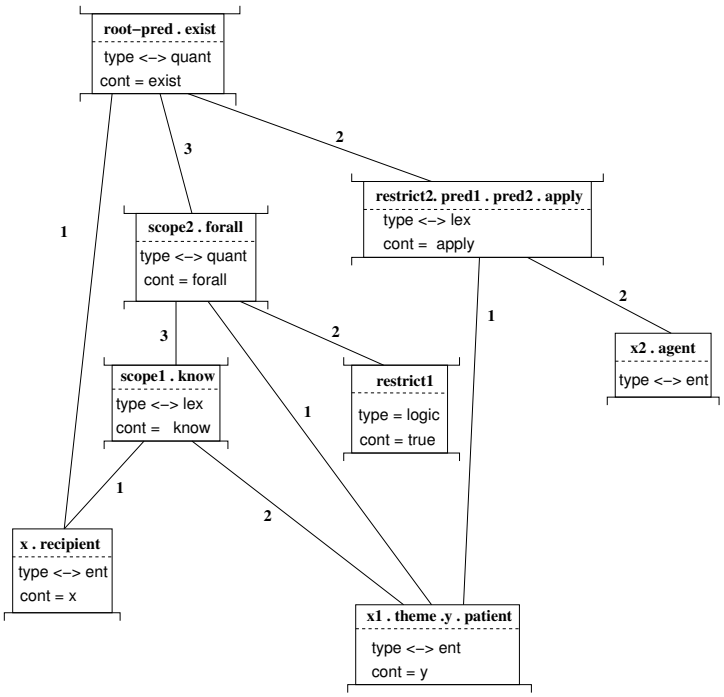

Fig. 4. Semantic DAG representing the second interpretation of the sentence *tous en connaissent une application*

which reduces to $\forall x(\exists\ y(apply(x_2, y) \wedge know(x, y))$. The free variable x_2 represents an anaphora, which refers to the entity "the parser about which we are talking" in the context of the sentence. The DAG of figure 4 represents the formula $\exists y(apply(x_2, y) \wedge (\forall x(true \Rightarrow know(x, y))))$, which reduces to

$\exists y(apply(x_2, y) \wedge (\forall x\ know(x, y)))$.

2.3 Computation of valid models by neutralization of polarities

As the syntactic level, polarities provide us with a method for computing the valid models of a given description. The operation of neutralization is defined in a same way and the principle of computation is the same: to iterate the operation of neutralization until we obtain a saturated DAG description, that is, a DAG description without positive and negative polarities. Then, the problem reduces to computing valid models of the resulting DAG description. If we start with the semantic DAG description D'_{sem}, we obtain the completely specified DAGs of figures 3 and 4 after four neutralizations. In this example, the valid models result immediatly from the neutralization process and we hope that, in realistic cases, it will occur similarly.
Now, it is not always useful to compute the valid models of a given DAG description, especially if their number is high due to the presence of many quantifiers in the sentence. For instance, in the context of translation, it can be preferable to keep the underspecified representation, which is more compact. By the way, it is this need of translation that motivated the design of new formalisms to represent semantics, formalisms that are centered around the notion of underspecification [15–17]. Our proposal is situated at a more meta level and we can express several formalisms that we have just mentioned inside the framework that we propose. Moreover, this provides an original mechanism of polarities which is used for controlling the process of reducing underspecification. From this point of view, it fits in with *Hole Semantics* [15]. In this formalism, the nodes of descriptions are either constants or holes and a mechanism of plugging holes with constants provides the means of reducing underspecification and producing models of a description. A second original feature of our mechanism is that description nodes can be either individuals or predicates, whereas in the other mentioned formalisms, only predicates are considered as description nodes. This uniformization of the representation has two advantages: it allows to go further with underspecification by leaving the type of a predicate argument non determined; secondly, this simplifies the syntax-semantics interface because some syntactic constituents can correspond to individuals as well as predicates.

3 Interaction between the syntactic level and the semantic level in the parsing process

In the two previous sections, we have described the syntactic and semantic levels separately. The linking between the two levels is performed by a simple

function that maps syntactic nodes to semantic ones. In figure 1, this function maps every node of D_{syn} to one node of D_{sem} at most and it is represented by dashed edges.
This linking function is not total because some syntactic nodes, most grammatical words for example, have no semantic representation. Figure 1 shows it as it shows that the function is not necessarily injective or surjective. Now, the objects that we manipulate are triples (D_{syn}, D_{sem}, L), which we call *SynSem descriptions* and which are defined as follows:

Definition 3.1 *A SynSem description is a triple (D_{syn}, D_{sem}, L) such that:*

- *D_{syn} is a syntactic tree description;*
- *D_{sem} is a semantic DAG description;*
- *L is a linking function that maps every node of D_{syn} to one node of D_{sem} at most.*

The definition of a valid model of a SynSem description stems naturally from the corresponding definitions for syntactic tree and semantic DAG descriptions. Since IG are lexicalized, the linking function between syntax and semantics is defined in the lexicon. Let us define it more precisely by giving the definition of a particular Interaction Grammar.

Definition 3.2 *An interaction grammar G is a pair (Ax_G, Lex_G) such that:*

- *Ax_G is particular SynSem description called the* axiom *of G. It consists of a single syntactic node linked with a single semantic node.*
- *Lex_G is a map that associates every word w of the language with a set of pairs (D_w, Anc_w) such that D_w is a SynSem description and Anc_w is a distinguished node of the syntactic part of D_w, called the* anchor *of D_w. Lex_G is the* lexicon *of G.*

The axiom Ax_G of the grammar is the SynSem description expressing the expected result of parsing a sentence. For instance, in figure 1, the axiom of the corresponding grammar consists of the syntactic node s linked with the semantic node *root-pred*.
Besides, in an entry (D_w, Anc_w) of the lexicon Lex_G, the anchor Anc_w represents the position of the word w in the underspecified syntactic tree constituting the syntactic side of the SynSem description D_w. In figure 1, the anchors of the descriptions associated to the words of the sentence are represented by grey rectangles.

Parsing a sentence $w_1.w_2 \dots w_n$ with an Interaction Grammar G consists first in selecting an entry (D_i, Anc_i) from the lexicon Lex_G for each word w_i. Then,

we make the disjoint union $D_0 = \biguplus_{i=1}^{n} D_i$ [11] . The SynSem description D_0 must be completed with precedence relations between the anchors Anc_i, which express word order in the sentence. It must be also completed with the axiom Ax_G and relations expressing its property of root both at the syntactic level and at the semantic level. We obtain a SynSem description D which represents a specification of the sentence $w_1.w_2 \ldots w_n$ with the grammar G. The SynSem description of figure 1 is built in this way.
Now, the problem consists in finding all valid models of D. Every model has two sides: a syntactic side which constitutes a parse tree of the sentence and a valid model of the syntactic side D_{syn} of D and a semantic side which constitutes a semantic representation of the sentence and a valid model of the semantic side D_{sem} of D. If we assume that the parsing process is driven by the syntax, by iterating feature neutralization inside D_{syn}, we reduce underspecification step by step until we obtain a syntactic tree which is completely specified and in which all polarized features have been neutralized.

In this process, the linking function between syntax and semantics plays two roles:

- it contributes to specify D_{sem}: when two syntactic nodes merge in a neutralization, their corresponding semantic nodes, if they exist, merge at the same time;
- the same mechanism can entail feedback from the semantic level to the syntactic one: if some constraints in D_{sem} forbid the merging of two semantic nodes which is entailed by the merging of two syntactic nodes, both mergings fail.

At the end of a successful parsing process, the description D_{sem} is generally more specific than initially but it can remain underspecified and we have to continue feature neutralizations at the only semantic level in order to obtain all semantic representations of the parsed sentence. We can drop this last phase if we want to keep a factorized semantic representation of the sentence.

4 Comparison with related approaches of the syntax-semantics interface

In the presentation of IG, we have stressed the interaction between syntax and semantics. That leads us to concentrate our attention on the syntax-semantics interface in the following comparison with related formalisms.

[11] If two elements D_i and D_j of the disjoint union have nodes with the same names or feature values with the same indices, we must rename the problematic nodes or indices in D_i or D_j.

IG have some similarities with Synchronous TAG [18,19]. Both aim at linking two levels of representation, which use the same composition principle: feature neutralization for IG and adjunction for TAG.
Nevertheless, the two formalisms have deep differences. Firstly, the adjoining operation of TAG is less flexible than feature neutralization: it allows only tree insertion, while feature neutralization allows tree superposition. Secondly, the interface between syntax and semantics is more constraining in TAG than in IG: every adjunction at the semantic level must be linked with an adjunction at the syntactic level so that the syntactic and the semantic derivation trees are isomorphic [19]. This isomorphism limits the expressiveness of the formalism, even if it can be relaxed a bit [20]. This is to be contrasted with IG, where, because of the flexibility of the syntax-semantics interface, feature cancellation at the semantic level can be disconnected from feature cancellation at the syntactic level.
A way of relaxing the rigidity of the interface syntax-semantics while keeping TAG as syntactic formalism is to base this interface on derived trees and not on derivation trees and to use a specific semantic formalism with its proper composition principle. The syntax-semantics interface uses specific features of syntactic nodes in derived trees which refer to semantic objects. The unification of these features in the construction of derived trees contribute to the construction of the semantic representation in parallel. There are different proposals based on this approach; they mainly differ in the semantic formalism that they use: Glue Semantics for A. Frank and John van Genabith [21] and Hole Semantics for C. Gardent and L. Kallmeyer [22]. These proposals are very close to IG in their design of the syntax-semantics interface and in the way of representing and solving underspecification at the semantic level: in both formalisms, Glue Semantics and Hole Semantics, the mechanism of composition can be interpreted through neutralization of polarities.

In the CG approach, P. de Groote and R. Muskens propose very close formalisms, which aim at relaxing and abstracting CG: respectively Abstract Categorial Grammars (ACG) [23] and Lambda Grammars (LG) [24]. These formalisms have several dimensions; one dimension can be instanciated by syntax and another dimension by semantics but the number of dimensions is not limited. In every dimension, the mechanism of composition of the structure is realized by the implicative fragment of intuitionistic linear logic in the form of a linear lambda-calculus and there is an isomorphism between the different dimensions : every application or abstraction in a dimension is linked with an operation of same type in the other dimensions. This isormorphism does not lead to the same rigidity as Synchronous TAG because, in every dimension, it is possible to use a language specific to this dimension and because the common language of composition can use higher order terms in a powerful manner. This use of higher order terms may be an advantage with respect to IG but at the same time, the fact that the composition of structures is strictly linear is a limitation: partial superposition of structures is not possible and an

example like *tous en connaissent une application* is not tractable simply with ACG or LG.

The integration of Minimal Recursion Semantics in HPSG [17] is another example of flexibility of the syntax-semantics interface, which is realized by co-indexation in feature structures, which represent both syntax and semantics. Unification is the principle of composition of structures and it can be viewed as partial superposition of DAGs. Contrary to IG, there is no control mechanism for guiding this superpostion with the goal of saturating structures. Saturation is guaranted afterwards by well-formedness principles.

References

[1] L. Tesnière, Comment construire une syntaxe, Bulletin de la Faculté des Lettres de Strasbourg 7 - $12^{ième}$ année (1934) 219–229.

[2] K. Adjukiewicz, Die syntaktische konnexität, Studia Philosophica 1 (1935) 1–27.

[3] A. Nasr, A formalism and a parser for lexicalised dependency grammars, in: 4th Int. Workshop on Parsing Technologies, Prague, Czechoslowakia, State University of NY Press, 1995, pp. 186–195.

[4] G. Perrier, Interaction grammars, in: 18th International Conference on Computational Linguistics, CoLing 2000, Sarrebrücken, 2000, pp. 600–606.

[5] G. K. Pullum, B. C. Scholz, On the Distinction between Model-Theoretic and Generative-Enumerative Syntactic Frameworks, in: P. de Groote, G. Morrill, C. Retoré (Eds.), Logical Aspects of Computational Linguistics, LACL 2001, Le Croisic, France, Vol. 2099 of Lecture Notes in Computer Science, Springer Verlag, 2001, pp. 17–43.

[6] M. Marcus, D. Hindle, M. Fleck, D-Theory: Talking about Talking about Trees, in: 21st Annual Meeting of the Association for Computational Linguistics, 1983, pp. 129–136.

[7] K. Vijay-Shankar, Using description of trees in a tree adjoining grammar, Computational Linguistics 18 (4) (1992) 481–517.

[8] J. Rogers, K. Vijay-Shanker, Reasoning with descriptions of trees, in: 30th Annual Meeting of the Association for Computational Linguistics, 1992, pp. 72–80.

[9] L. Kallmeyer, Tree description grammars and underspecified representations, Ph.D. thesis, Universität Tübingen (1999).

[10] O. Rambow, K. Vijay-Shanker, D. Weir, D-tree substitution grammars, Computational Linguistics 27 (1) (2001) 87–121.

[11] B. Carpenter, Type-logical Semantics, MIT Press, Cambridge, Massachusetts, 1998.

[12] S. Kahane, Grammaires d'unification polarisées, in: 11ième Conférence sur le Traitement Automatique des Langues, TALN 2004, Fès, Maroc, 2004, pp. 233–242.

[13] G. Bonfante, B. Guillaume, G. Perrier, Analyse syntaxique électrostatique, Traitement Automatique des Langues 44 (3) (2003) 93–120.

[14] G. Bonfante, B. Guillaume, G. Perrier, Polarization and abstraction of grammatical formalisms as methods for lexical disambiguation, in: 20th International Conference on Computational Linguistics, CoLing 2004, Genève, Switzerland, 2004, pp. 303–309.

[15] J. Bos, Predicate logic unplugged, in: P. Dekker, M. Stokhof (Eds.), 10th Amsterdam Colloquium, 1995, pp. 133–142.

[16] M. Egg, J. Niehren, P. Ruhrberg, F. Xu, Constraints over lambda structures in semantic underspecification., in: 17th International Conference on Computational Linguistics and 36th Annual Meeting of the Association for Compuational Linguistics, COLING/ACL'98, Montreal,Quebec, Canada, 1998, pp. 353–359.

[17] A. Copestake, D. Flickinger, I. Sag, Minimal Recursion Semantics - an Introduction, draft (1999).

[18] S. Shieber, Y. Schabes, Synchronous Tree-Adjoining Grammars, in: 13th International Conference on Computational Linguistics, CoLing'90, Helsinki, Vol. 3, 1990, pp. 1–6.

[19] S. Shieber, Restricting the Weak-Generative Capacity of Synchronous Tree-Adjoining Grammars, Computational Intelligence 10 (4) (1994) 371–385.

[20] O. Rambow, G. Satta, Synchronous Models of Language, in: 34th Annual Meeting of the Association for Computational Linguistics, ACL'96, Santa Cruz, 1996, pp. 116–123.

[21] A. Frank, J. van Genabith, GlueTag: Linear Logic based Semantics for LTAG, in: M. Butt, T. H. King (Eds.), 6th International Lexical-Functional Grammar Conference, LFG01, Hong Kong, CSLI Publications, 2001, pp. 104–126.

[22] C. Gardent, L. Kallmeyer, Semantic construction in FTAG, in: 11th Conference of the European Chapter of the Association for Computational Linguistics, EACL'2003, Budapest, Hungary, 2003.

[23] P. de Groote, Towards Abstract Categorial Grammars, in: 39th Annual Meeting of the Association for Computational Linguistics and 10th Conference of the European Chapter of the Association for Computational Linguistics,Toulouse, France, 2001, pp. 148–155.

[24] R. Muskens, Lambda Grammars, in: Prospects and Advances in the Syntax/Semantics Interface, Lorraine-Saarland Workshop Series, Nancy, 2003, pp. 29–32.