A Meta-Grammatical Framework for Dependency Grammar

Ralph Debusmann, Denys Duchier

A Meta-Grammatical Framework for Dependency Grammar

Abstract

We propose a meta-grammatical framework for dependency grammar, accommodating any number of dimensions and restricting their licensed models through the application of parametric principles. We first instantiate this framework to obtain the version of Topological Dependency Grammar (TDG) proposed by (Duchier and Debusmann, 2001). We then describe an extended instantiation which adds support for semantic dependencies (thus also providing an account of control and raising constructions) and meaning assembly.

Topological Dependency Grammar (TDG) (Duchier and Debusmann, 2001) explains linearization phenomena through the interaction of two structures, similar to (Gerdes and Kahane, 2001): a non-ordered tree of syntactic dependencies and an ordered and projective tree of topological dependencies.

In this paper, we propose a generalized meta-grammatical framework, accommodating any number of dimensions (beyond syntax and topology) and restricting their licensed models through the application of parametric principles. First we show how this framework can be instantiated to obtain the earlier version of TDG. We then describe an extended instantiation which adds support for semantic dependencies (thus also providing an account of control and raising constructions) and meaning assembly. Figure 1 sketches briefly how the extended TDG is articulated around multiple-dimensions.

1 ID/LP TDG

In this section, we explain the basics of TDG. We regard TDG as a general grammar framework that needs to be instantiated to yield a particular grammar formalism. One instance is ID/LP TDG, as proposed in (Duchier and Debusmann, 2001).[1]

ID/LP TDG is a lexicalized formulation of dependency grammar (Tesnière, 1959) that combines a non-projective account of syntax with an account of linear precedence inspired by the classical model of topological fields (Höhle, 1986). An analysis consists of two mutually constraining trees: a non-ordered tree of syntactic dependencies (ID tree) and an ordered and projective tree of topological dependencies (LP tree).

We introduce the ID/LP TDG framework using the following German example sentence:

(1) *(dass) eine Frau jeder Mann*
(that) a woman+ACC every man+NOM
zu lieben scheint.
to love seems.
"(that) every man seems to love a woman."

1.1 ID tree

The ID tree is a non-ordered tree of syntactic dependencies where edges are labeled with grammatical functions such as subj for subject or obj for object. Here is an ID tree analysis for example (1).[2]

[1] "ID/LP TDG" used to be called "TDG" in (Duchier and Debusmann, 2001) and various other papers on the subject.

[2] For simplicity, we treat the NPs (*eine Frau* and *jeder Mann*) and the head of the VP (*zu lieben*) as independent words throughout this paper.

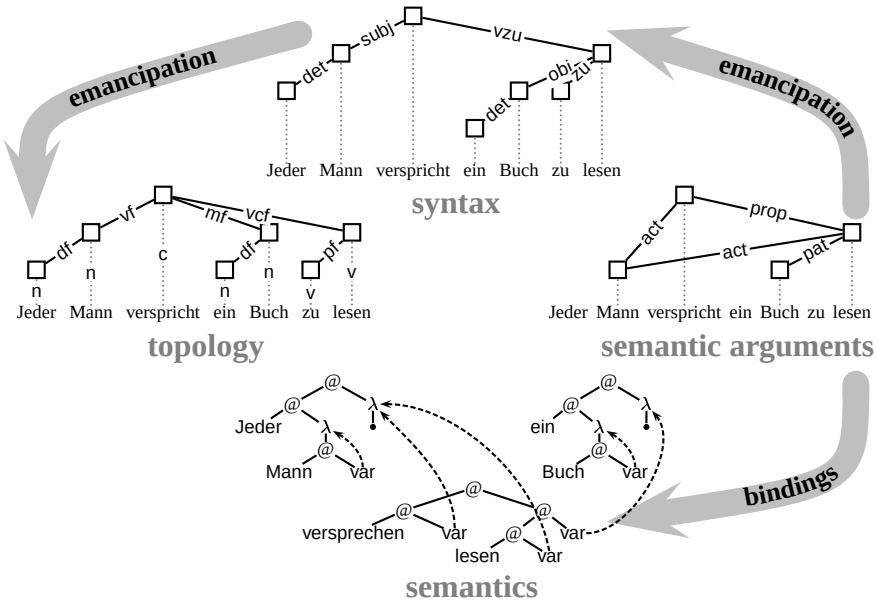
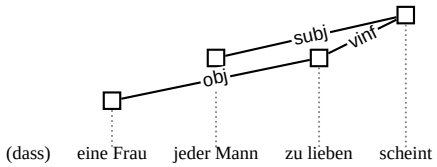


Figure 1: An architectural overview of extended TDG

(2)

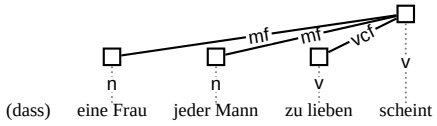


jeder Mann is the subject of *scheint*, as signified by the edge labeled subj. Likewise, *zu lieben* is the infinitival complement (vinf) of *scheint*, and *eine Frau* is the object (obj) of *zu lieben*.

1.2 LP tree

The LP tree is an ordered and projective tree of topological dependencies. Edges of the LP tree are labeled with topological fields such as mf for "Mittelfeld" and vcf for verb field. In addition, to order nodes with respect to their dependents, we label the nodes of the LP tree by head fields such as v for verbs and n for nouns. Here is an LP tree analysis for example (1):

(3)



The two NPs *eine Frau* and *jeder Mann* both land in the Mittelfeld (mf) of the finite verb *scheint*, and the infinitival complement *zu lieben* lands in the verb field (vcf) of *scheint*.

LP trees must be well-ordered. To this end, the grammar stipulates a total order on LP edge and node labels. This imposes a linear order among the dependents of a node, and the node itself is positioned with respect to its dependents by its node label. The example grammar presented in this paper stipulates the following total order:

(4) $\mathsf{n} \prec \mathsf{mf} \prec \mathsf{vcf} \prec \mathsf{v}$

The LP tree above is well-ordered with respect to this total order: the topological dependents of *scheint* labeled mf (i.e. *eine Frau* and *jeder Mann*) correctly precede the ones labeled vcf (*zu lieben*):

(5) $[\textit{eine Frau jeder Mann}]_{\mathsf{mf}} < [\textit{zu lieben}]_{\mathsf{vcf}}$

Moreover, the node *scheint* itself correctly follows all of its dependents:

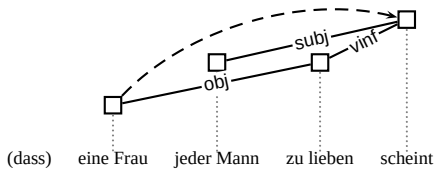
(6) $[\textit{eine Frau jeder Mann}]_{\mathsf{mf}} < [\textit{zu lieben}]_{\mathsf{vcf}} < \textit{scheint}_{\mathsf{v}}$

Note that *eine Frau* and *jeder Mann* have the same incoming edge label mf. That is, their respective order is underspecified, and either permutation (*eine Frau* preceding *jeder Mann* or the other way round) is licensed by the grammar. This reflects that in German, the order of the elements in the Mittelfeld is arbitrary.

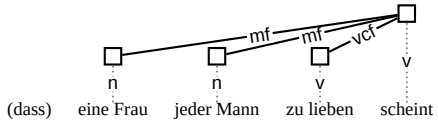
1.3 Climbing

The ID and LP dimensions mutually constrain each other by the *climbing principle*. By the climbing principle, we allow nodes in the ID tree to migrate upwards to nodes higher in the LP tree. In our example for instance, the object *eine Frau* is the daughter of *zu lieben* in the ID tree, and of *scheint* in the LP tree. Hence we say that *eine Frau* has *climbed up* to *scheint*. For illustration, we repeat our ID and LP tree analyses, augmented with a dashed arrow which signifies the climbing up of the node corresponding to *eine Frau*:

(7)



(8)



The effect of climbing is that the shape of the LP tree is a flattening of the ID tree's, similar to the effect achieved by sequence union in (Reape, 1994). This enables ID/LP TDG to handle a variety of difficult word order phenomena, especially in freer word order languages such as Dutch or German.

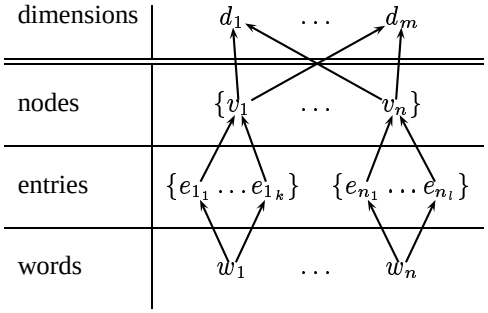
2 Generalized TDG

In this section, we generalize the TDG framework, starting from the ID/LP TDG grammar formalism outlined above.

2.1 Dimensions

Each analysis in TDG consists of any number m of dimensions $D = \{d_1, \ldots, d_m\}$ whose models are labeled directed graphs. These graphs are all made up of the same set of n nodes $V = \{v_1, \ldots, v_n\}$, which correspond $1:1$ to the words in the input sentence $W = w_1 \ldots w_n$. To model lexical ambiguity, we map each word w_i to a set of i_k lexical entries $E_i = \{e_{i_1}, \ldots, e_{i_k}\}$. Lexical entries are standard feature structures containing information about e.g. the subcategorization behavior. A selection function then selects one entry e_{i_j} from E_i that is eventually assigned to the node v_i corresponding to word w_i. Below, we give a sketch of this architecture:

(9)

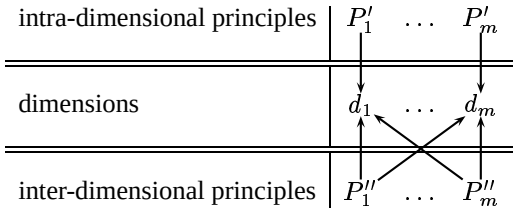


Note that the lexical entries are *shared* across all dimensions. In fact they are one way to make the manifold dimensions mutually constrain each other.

2.2 Principles

Each dimension d_i is subject to a set P_i of *principles*, which are drawn from the same *principles pool* P. The models for dimension d_i are in the class of labeled directed graphs, and the principles further restrict their admissibility. Principles can operate only within one dimension or across several dimensions: We call principles that only operate within one dimension *intra-dimensional principles*, and those that operate across several dimensions *inter-dimensional principles*. Each set P_i of principles on dimension d_i is partitioned into subsets P_i' of intra-dimensional principles and P_i'' of inter-dimensional principles: This is illustrated in the picture below.

(10)



2.3 Principles pool

ID/LP TDG draws all its principles from the following principles pool:

(11) $P = \{$dag, tree, in, out, order, climbing, linking$\}$

Each principle has arguments such as the dimensions $d_1, \ldots, d_m$ on which the principle applies, or functions, taking a node in V as argument, that model lexical features.

Dag principle. *dag(d): Each analysis on dimension d is a directed acyclic graph.*

Tree principle. *tree(d): Each analysis on dimension d is a tree.*

Climbing principle. *climbing(d_1,d_2): A subtree on dimension d_2 may climb up and land higher up on dimension d_1.* Writing $v \rightarrow_d v'$ for an edge from v to v' on dimension d, and $v \rightarrow_d^* v'$ for its reflexive transitive closure, the climbing principle states the following constraint:

$$(12) \quad v \rightarrow_{d_1} v' \quad \Rightarrow \quad v \rightarrow_{d_2}^* v'$$

That is, an edge from v to v' on dimension d_1 is licensed only if v dominates v' on dimension d_2.

Out principle. *out(d,$\mathsf{out}_d : V \rightarrow 2^{\Pi_{\mathcal{L}}}$): The outgoing edges of a node on dimension d must satisfy in label and number the stipulation of the corresponding out_d feature.* This principle is parametrized by the lexical feature out_d, and $\mathcal{L}$ is the set of edge labels on dimension d. Π is a constructor for *label patterns*. Given a set $\mathcal{L}$ of labels, we write $\Pi_{\mathcal{L}}$ for the set of label patterns π that can be formed according to the following abstract syntax:

$$(13) \quad \pi \quad ::= \quad \ell \mid \ell? \mid \ell*$$

where ℓ stands for "precisely one" outgoing edge, $\ell?$ "zero or one", and $\ell*$ "any number" of outgoing edges.

In principle. *in(d,$\mathsf{in}_d : V \rightarrow 2^{\Pi_{\mathcal{L}}}$): The incoming edges of a node on dimension d must satisfy in label and number the stipulation of the corresponding in_d feature.* Symmetrical to the out principle, the in principle constrains the incoming edges of a node.

Order principle. *order(d,$\prec_d$,$\mathsf{on}_d : V \rightarrow 2^{\mathcal{F}_\mathsf{N}}$)* For the order principle, we refer the reader to (Duchier, 2001). Suffice to say that it applies on dimension d with the total order $\prec_d$. Lexical feature on_d states the set of possible node labels.

Linking principle. *linking(d_1,d_2,$\mathsf{link}_{d_1,d_2} : V \rightarrow \mathcal{L}_1 \rightarrow 2^{\mathcal{L}_2}$): The dependents of a node on dimension d_1 must have as their incoming edge label on dimension d_2 one of those stipulated by the link_{d_1,d_2} feature.* $\mathcal{L}_1$ is the set of edge labels of dimension d_1, and $\mathcal{L}_2$ of dimension d_2. Essentially, the principle synchronizes two dimensions in a lexical fashion, by the following constraint, where $\ell_1 \in \mathcal{L}_1$ and $\ell_2 \in \mathcal{L}_2$:

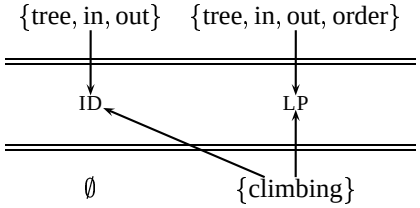
$$(14) \quad v -\ell_1\rightarrow_{d_1} v' \quad \Rightarrow \quad v'' -\ell_2\rightarrow_{d_2} v' \wedge \ell_2 \in \mathsf{link}_{d_1,d_2}(v)(\ell_1)$$

That is, an edge from v to v' on dimension d_1 is licensed only if, on dimension d_2, there is a node v'' and an edge from v'' to v' labeled by one of $\mathsf{link}_{d_1,d_2}(v)(\ell_1)$.

2.4 Obtaining ID/LP TDG

In the following, we instantiate TDG to get the ID/LP TDG grammar formalism:

(15)



On the ID dimension, ID/LP TDG makes use of the principles tree, in and out. On the LP dimension, it reuses the three, and adds the order principle. ID/LP TDG uses only one inter-dimensional principle, viz. the climbing principle on the LP dimension.

The instantiation of TDG in (15) gives rise to the following lexical entry signature:

$$(16) \quad \left[\begin{array}{lcl} \text{ID} & : & \left[\begin{array}{rcl} \mathsf{in} & : & 2^{\Pi_{\mathcal{R}}} \\ \mathsf{out} & : & 2^{\Pi_{\mathcal{R}}} \end{array}\right] \\ \text{LP} & : & \left[\begin{array}{rcl} \mathsf{in} & : & 2^{\Pi_{\mathcal{F}_\mathsf{E}}} \\ \mathsf{on} & : & 2^{\mathcal{F}_\mathsf{N}} \\ \mathsf{out} & : & 2^{\Pi_{\mathcal{F}_\mathsf{E}}} \end{array}\right] \end{array}\right]$$

where $\mathcal{R}$ is the set of edge labels on the ID dimension, $\mathcal{F}_\mathsf{E}$ the set of edge labels, and $\mathcal{F}_\mathsf{N}$ the set of node labels on the LP dimension.

2.5 ID/LP TDG example grammar

We present a simple example grammar for the ID/LP TDG grammar formalism. First, we define the edge labels on the ID and the LP dimension ($\mathcal{R}$ and $\mathcal{F}_\mathsf{E}$), and the node labels on the LP dimension ($\mathcal{F}_\mathsf{N}$):

$$(17) \quad \begin{array}{rcl} \mathcal{R} & = & \{\mathsf{subj}, \mathsf{obj}, \mathsf{vinf}\} \\ \mathcal{F}_\mathsf{E} & = & \{\mathsf{mf}, \mathsf{vcf}\} \\ \mathcal{F}_\mathsf{N} & = & \{\mathsf{n}, \mathsf{v}\} \end{array}$$

For the order principle, we define the total order on $\mathcal{F}_\mathsf{E} \cup \mathcal{F}_\mathsf{N}$ as already posited in (4).

The rest of the grammar is made up of the lexical entries:

jeder Mann $\mapsto$

$$(18) \quad \left[\begin{array}{lcl} \text{ID} & : & \left[\begin{array}{rcl} \mathsf{in} & : & \{\mathsf{subj?}\} \\ \mathsf{out} & : & \emptyset \end{array}\right] \\ \text{LP} & : & \left[\begin{array}{rcl} \mathsf{in} & : & \{\mathsf{mf?}\} \\ \mathsf{on} & : & \{\mathsf{n}\} \\ \mathsf{out} & : & \emptyset \end{array}\right] \end{array}\right]$$

(19) *eine Frau* ↦

$$\begin{bmatrix} \text{ID} : \begin{bmatrix} \text{in} : \{\text{obj?}\} \\ \text{out} : \emptyset \end{bmatrix} \\ \text{LP} : \begin{bmatrix} \text{in} : \{\text{mf?}\} \\ \text{on} : \{\text{n}\} \\ \text{out} : \emptyset \end{bmatrix} \end{bmatrix}$$

(20) *zu lieben* ↦

$$\begin{bmatrix} \text{ID} : \begin{bmatrix} \text{in} : \{\text{vinf?}\} \\ \text{out} : \{\text{obj}\} \end{bmatrix} \\ \text{LP} : \begin{bmatrix} \text{in} : \{\text{vcf?}\} \\ \text{on} : \{\text{v}\} \\ \text{out} : \emptyset \end{bmatrix} \end{bmatrix}$$

(21) *scheint* ↦

$$\begin{bmatrix} \text{ID} : \begin{bmatrix} \text{in} : \emptyset \\ \text{out} : \{\text{subj}\} \end{bmatrix} \\ \text{LP} : \begin{bmatrix} \text{in} : \emptyset \\ \text{on} : \{\text{v}\} \\ \text{out} : \{\text{mf}*, \text{vcf?}\} \end{bmatrix} \end{bmatrix}$$

The lexical entries for the NPs *jeder Mann* and *eine Frau* are almost identical: on the ID dimension, their incoming edge must be labeled subj (*jeder Mann*) or obj (*eine Frau*), i.e. *jeder Mann* must be a subject, and *eine Frau* an object. On the LP dimension, their only possible incoming edge label is mf, which captures the generalization that nouns can only land in the topological field called Mittelfeld in German. Their node label must be n.

The lexical entry for the infinitive *zu lieben* states that its incoming edge must be vinf on the ID dimension and vcf on the LP dimension. The only possible node label is v. Moreover, *zu lieben* must have precisely one dependent labeled obj on the ID dimension. On the LP dimension, it must not have any outgoing edge.

The lexical entry for *scheint* states the following. On the ID dimension, *scheint* must not have any incoming edges, and it requires precisely one outgoing edge labeled subj. On the LP dimension, *scheint* also must not have any incoming edge, and has node label v. In addition, the lexical entry for *scheint* licenses an arbitrary number of outgoing edges labeled mf, and zero or one outgoing edges labeled vcf.

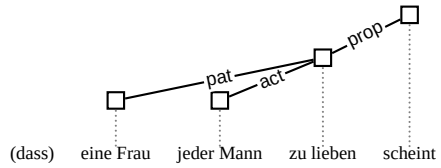
3 ID/LP/TH TDG

In this section, we instantiate TDG to get the ID/LP/TH TDG grammar formalism, extending ID/LP TDG of the previous section by a dimension called TH dimension (for *thematic*).

3.1 ID/LP/TH dags

An analysis on the TH dimension is called TH dag: a directed acyclic graph with edges labeled by semantic roles. Here is a TH dag for sentence (1):

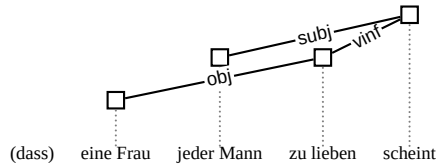
(22)



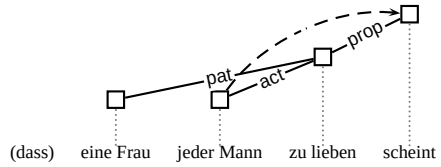
Here, *zu lieben* is a prop-dependent (for *proposition*) of *scheint*. *jeder Mann* is the actor or deep subject (act), and *eine Frau* is the patient or deep object (pat) of *zu lieben*. These labels (except prop) are taken from the set of dependency relations on the tectogrammatical layer of Functional Generative Description (FGD) (Sgall et al., 1986).

TH dags represent the *semantic* argument structure of a sentence, contrary to ID trees, which represent the *syntactic* argument structure. The two do not always match. We contrast below the ID tree (23) with the TH dag analysis (24) of our example sentence:

(23)



(24)



In the ID tree, *jeder Mann* is the subject of *scheint*, and *eine Frau* is the object of *zu lieben*. In the TH dag, *eine Frau* also is a dependent of *zu lieben* (its patient). However, *jeder Mann* is not the dependent of *scheint* but of *zu lieben* (its actor). Again, as in ID/LP TDG, a node has climbed up: *jeder Mann* is the daughter of *zu lieben* on the TH dimension, and of *scheint* on the ID dimension, as indicated by the dashed arrow in (24). In standard linguistics terminology, this phenomenon is called *raising*.

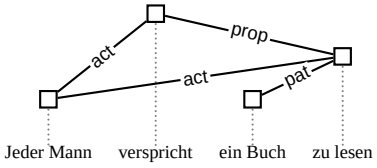
Here, the climbing direction is from the TH to the ID dimension. Before, in ID/LP TDG, we used climbing in the direction from the ID to the LP dimension. ID-LP climbing enabled us to model word

order freedom, whereas the TH-ID climbing allows us to model control and raising phenomena.

The attentive reader might pose the question why an analysis on the TH dimension is a directed acyclic graph, and not a tree. The answer is that for phenomena such as control, a node in the TH dag can have more than one incoming edge. As an example, consider the following sentence, including the control verb *verspricht*:

(25) *Jeder* *Mann* *verspricht* *ein* *Buch*
Every man+NOM tries a book+ACC
zu *lesen.*
to read.
"Every man tries to read a book."

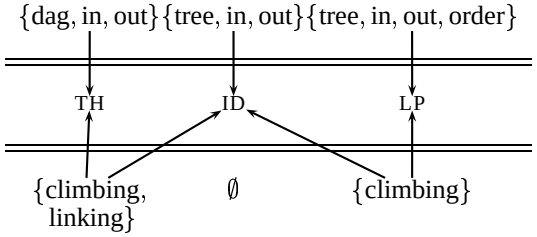
Below, we display the corresponding TH dag:

(26)


Here, *jeder Mann* has two incoming edges: it is not only the actor of *zu lieben*, but also of *verspricht*.

3.2 Obtaining ID/LP/TH TDG

ID/LP/TH TDG is an instantiation of TDG along the following lines:

(27)


That is, ID/LP/TH TDG recycles all principles already used in ID/LP TDG. Additionally, ID/LP/TH TDG utilizes the *dag principle* and the *linking principle*. We use the linking principle to specify by which grammatical function a semantic role can be realized. E.g. subjects typically realize actors. TDG's linking principle has been elaborated in (Korthals and Debusmann, 2002), an approach that even surpasses the linking theory of (Davis, 1998) in some respects.

The instantiation of TDG in (27) gives rise to the following lexical entry signature:

(28)
$$\left[\begin{array}{lcl} \textsc{id} & : & \left[\begin{array}{lcl} \mathsf{in} & : & 2^{\Pi_{\mathcal{R}}} \\ \mathsf{out} & : & 2^{\Pi_{\mathcal{R}}} \end{array}\right] \\ \textsc{lp} & : & \left[\begin{array}{lcl} \mathsf{in} & : & 2^{\Pi_{\mathcal{F}_\mathsf{E}}} \\ \mathsf{on} & : & 2^{\mathcal{F}_\mathsf{N}} \\ \mathsf{out} & : & 2^{\Pi_{\mathcal{F}_\mathsf{E}}} \end{array}\right] \\ \textsc{th} & : & \left[\begin{array}{lcl} \mathsf{in} & : & 2^{\Pi_{\mathcal{S}}} \\ \mathsf{out} & : & 2^{\Pi_{\mathcal{S}}} \\ \mathsf{link} & : & \mathcal{S} \rightarrow 2^{\mathcal{R}} \end{array}\right] \end{array}\right]$$

where the ID and LP subsignatures are the same as in (16). In the TH subsignature, $\mathcal{S}$ is the set of edge labels on the TH dimension.

3.3 ID/LP/TH TDG example grammar

We present a simple example grammar for the ID/LP/TH TDG grammar formalism. We use the same sets of labels for the ID and LP dimensions as before in (17). The total order on the set $\mathcal{F}_\mathsf{E} \cup \mathcal{F}_\mathsf{N}$ of edge and node labels also stays the same. We define the new set $\mathcal{S}$ of edge labels on the TH dimension as follows:

(29) $\mathcal{S} = \{\mathsf{act}, \mathsf{pat}, \mathsf{prop}\}$

The rest of the grammar again consists of the lexical entries:[3]

(30) *jeder Mann* $\mapsto$
$$\left[\begin{array}{lcl} \textsc{th} & : & \left[\begin{array}{lcl} \mathsf{in} & : & \{\mathsf{act?}\} \\ \mathsf{out} & : & \emptyset \\ \mathsf{link} & : & \emptyset \end{array}\right] \end{array}\right]$$

(31) *eine Frau* $\mapsto$
$$\left[\begin{array}{lcl} \textsc{th} & : & \left[\begin{array}{lcl} \mathsf{in} & : & \{\mathsf{pat?}\} \\ \mathsf{out} & : & \emptyset \\ \mathsf{link} & : & \emptyset \end{array}\right] \end{array}\right]$$

(32) *zu lieben* $\mapsto$
$$\left[\begin{array}{lcl} \textsc{th} & : & \left[\begin{array}{lcl} \mathsf{in} & : & \{\mathsf{prop?}\} \\ \mathsf{out} & : & \{\mathsf{act}, \mathsf{pat}\} \\ \mathsf{link} & : & \{\mathsf{act} \mapsto \{\mathsf{subj}\}, \\ & & \mathsf{pat} \mapsto \{\mathsf{obj}\}\} \end{array}\right] \end{array}\right]$$

(33) *scheint* $\mapsto$
$$\left[\begin{array}{lcl} \textsc{th} & : & \left[\begin{array}{lcl} \mathsf{in} & : & \emptyset \\ \mathsf{out} & : & \{\mathsf{prop}\} \\ \mathsf{link} & : & \{\mathsf{prop} \mapsto \{\mathsf{vinf}\} \end{array}\right] \end{array}\right]$$

In this lexicon, *jeder Mann* can only be an actor, and *eine Frau* a patient by their in features, and *zu lieben* can only be a proposition (prop). In addition, *zu lieben* requires an actor and a patient by its out feature. The link feature states that the actor must

[3] Note that we do not repeat the ID and LP parts of the lexical entries already shown in (18–21).

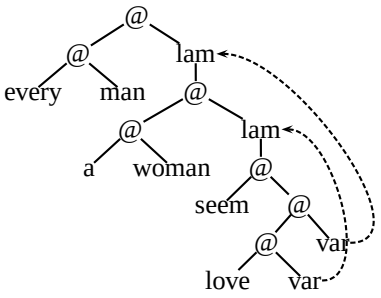
be realized by a subject, and the patient by an object. *scheint* requires a proposition (prop) by its out feature, and this proposition must be realized by an infinitival complement (vinf).

4 Representation of Semantics

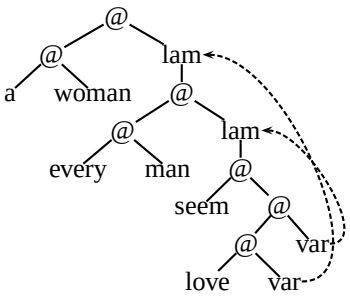
Although there are many alternatives, we will present an approach based on CLLS (Egg et al., 2001) which, as a constraint language, is well suited to our constraint-based methodology.

Our example sentence (1) is scopally ambiguous: it has a weak (34) and a strong (35) reading, represented as follows in CLLS:

(34)



(35)

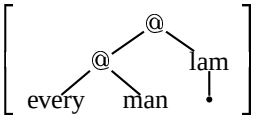


where '@' stands for application, 'lam' for a lambda binder, 'var' for a variable occurrence, and dashed arrows point from each variable occurrence to its corresponding binder.

5 Meaning Assembly

We now turn to the problem of applying our framework to the derivation of CLLS-based representations of semantics. Clearly, we wish that the lexical entries assigned to the nodes of an analysis contribute CLLS fragments, e.g.:

(36) *jeder Mann* $\mapsto$



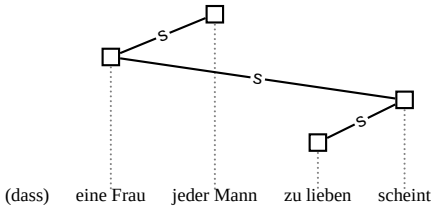
The challenge is then to *assemble* these fragments into a single complete description.

5.1 Derivation tree

Similar to Tree Adjoining Grammar (TAG) (Joshi, 1987), we describe the plugging together of the individual CLLS fragments using the concept of a *derivation tree*. A derivation tree describes which fragments are plugged into which fragments, and at what positions. For this reason, we identify distinguished positions in each fragment. The root node of a fragment is identified by the *root* feature, and the scope position by the *s* feature. This can be seen in the overview of the CLLS part of the lexical entries in Figure 2.[4]

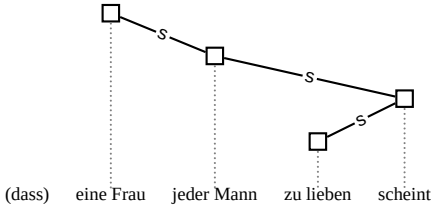
In this manner, we can model derivation trees using dependency trees where edges are labeled by distinguished positions. An edge $w_1 - s \rightarrow w_2$ indicates that the *root* node of w_2's fragment is plugged into the *s* position (scope) of w_1's fragment. We introduce the new DE dimension to represent the derivation tree, leading to the ID/LP/TH/DE TDG instance of the TDG framework. The DE dimension uses the tree, in and out principles. For instance, here is the derivation tree for the weak reading of example (1):

(37)



while the strong reading is given by:

(38)



From the derivation tree, according to the implication below, we read off the additional CLLS equality constraints that express how fragments are plugged together:

(39) $v - \ell \rightarrow_{\text{DE}} v' \quad \Rightarrow \quad v.\ell = v'.root$

[4] For want of space, we omit e.g. the *q* position for quantifiers, and the *r* position for restrictions that are both present in the full account.

5.2 Lambda bindings

To recover the lambda bindings for CLLS, we again identify distinguished positions in the CLLS fragments: we identify the n lambda binders in a fragment with features $lam_1, \ldots, lam_n$, and the m variable occurrences by features $var_1, \ldots, var_m$. N is the union of all these features.

We make use of the information contained in the TH dag to recover the lambda bindings. We introduce two new lexical features lam and var of type $\mathcal{S} \rightarrow 2^N$, mapping lam semantic roles to sets of lambda binder positions and to sets of variable positions respectively. lam and var map a semantic role θ either to the singleton set of the position corresponding to θ, or otherwise to the empty set. For an edge $v{-}\theta{\rightarrow_{\text{TH}}}v'$ in the TH dag, we obtain the corresponding lambda binding (dashed arrow) as follows:

(40) $v{-}\theta{\rightarrow_{\text{TH}}}v' \quad \Rightarrow \quad v.\text{var}.\theta \qquad v'.\text{lam}.\theta$

However note that we do not obtain lambda bindings for all edges: we obtain a lambda binding for edge $v{-}\theta{\rightarrow_{\text{TH}}}v'$ only if both $v.\text{var}.\theta$ and $v'.\text{lam}.\theta$ denote singletons. Thus we can exclude lambda bindings to positions in fragments which do not correspond to individuals (e.g. *scheint*).

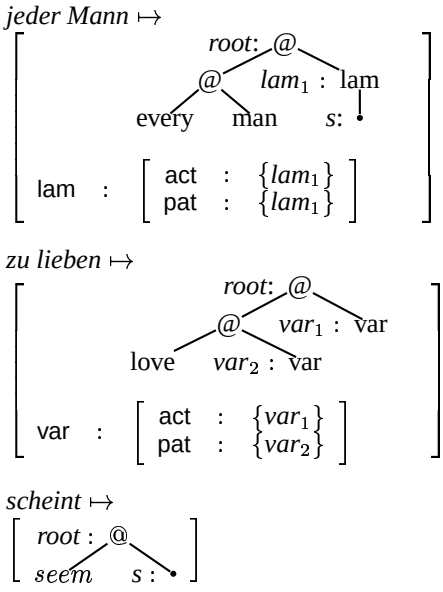


Figure 2: CLLS parts of the lexical entries

6 Conclusion

We presented a meta-grammatical framework, articulated around the notion of dependencies and lexicalized constraints, that generalizes TDG: instead of being limited to two interacting dimensions, one dedicated to syntax and the other to topology, this new framework can accommodate any number of dimensions. Furthermore, parametric principles are used to stipulate the models licensed for each dimension, as well the restrictions arising from inter-dimensional interactions. We have shown how it can be instantiated to obtain the earlier ID/LP TDG. We then described the ID/LP/TH extension which provides support for semantic dependencies and accounts for control and raising constructions. Finally the DE extension was presented to support meaning assembly in CLLS.

References

Tony Davis. 1998. *Linking as constraints in the hierarchical lexicon.*

Denys Duchier and Ralph Debusmann. 2001. Topological dependency trees: A constraint-based account of linear precedence. In *ACL 2001 Proceedings*.

Denys Duchier. 2001. Lexicalized syntax and topology for non-projective dependency grammar. In *MOL 8 Proceedings*.

Markus Egg, Alexander Koller, and Joachim Niehren. 2001. The constraint language for lambda structures. *Journal of Logic, Language, and Information.*

Kim Gerdes and Sylvain Kahane. 2001. Word order in german: A formal dependency grammar using a topological hierarchy. In *ACL 2001 Proceedings*.

Tilman Höhle. 1986. Der Begriff "Mittelfeld", Anmerkungen über die Theorie der topologischen Felder.

Aravind K. Joshi. 1987. An introduction to tree-adjoining grammars. In *Mathematics of Language*.

Christian Korthals and Ralph Debusmann. 2002. Linking syntactic and semantic arguments in a dependency-based formalism. In *COLING 2002 Proceedings*.

Mike Reape. 1994. Domain union and word order variation in german. In *German in Head-Driven Phrase Structure Grammar*.

Petr Sgall, Eva Hajicova, and Jarmila Panevova. 1986. *The Meaning of the Sentence in its Semantic and Pragmatic Aspects*.

Lucien Tesnière. 1959. *Eléments de Syntaxe Structurale*.