Dependency Grammar as Graph Description

Ralph Debusmann
Programming Systems Lab
Universität des Saarlandes, Geb. 45
Postfach 15 11 50
66041 Saarbrücken, Germany
rade@ps.uni-sb.de

Abstract

The paper introduces a generalisation of Topological Dependency Grammar (TDG) (Duchier and Debusmann, 2001). The result, Extensible Dependency Grammar (XDG), is a description language for sets of labeled directed graphs. Lexicalisation turns XDG into a powerful meta grammar formalism. XDG can be instantiated to yield specific grammar formalisms based on dependency grammar. We present one of these instances, Semantic Topological Dependency Grammar (STDG), a new grammar formalism with a syntax-semantics interface to underspecified semantics.

1 Introduction

(Duchier and Debusmann, 2001) introduced the grammar formalism of Topological Dependency Grammar (TDG). The most distinguishing feature of TDG is its ability to specify non-trivial restrictions on word order. These restrictions emerge from the interaction of simple declarative principles, and lead to an elegant account of many notorious word order phenomena in German.

TDG concentrates only on syntax and does not offer a syntax-semantics interface. In this paper, we introduce Semantic Topological Dependency Grammar (STDG), which extends TDG with an interface to underspecified semantics. As the target semantics formalism, we employ the Constraint Language for Lambda Structures (CLLS) (Egg et al., 1998).

2 Extensible Dependency Grammar (XDG)

XDG describes a set of graph dimensions $d_1, \ldots, d_m$, where each graph dimension $d$ corresponds to a graph $G_d(V, E_d)$. All graphs share the same set $V = \{v_1, \ldots, v_n\}$ of nodes, but have different edges $E_d \subseteq V \times L_d \times V$, and different edge labels $L_d$. Feature structures can be attached to each node, where features are functions $V \rightarrow R$ to an arbitrary codomain $R$. Before we extend TDG, we take a step backwards: we generalise TDG to Extensible Dependency Grammar (XDG), a graph description language over sets of labeled directed graphs. Only by lexicalisation, i.e. the addition of lexicalised feature structures, does XDG become a meta grammar formalism for dependency grammar. TDG is one of the possible instances of XDG.

XDG can be likened to Head-driven Phrase Structure Grammar (HPSG) (Pollard and Sag, 1994): both XDG and HPSG are meta grammar formalisms which need to be instantiated with particular principles and parameters to obtain specific grammar formalisms. The fundamental difference is that whereas HPSG is based on typed feature structures, XDG is based on graph descriptions.

The purpose of XDG is twofold. On the one hand, XDG can be utilised for research on existing dependency-based grammar formalisms. On the other, XDG serves as a launchpad for the development of new grammar formalisms like STDG.

The structure of this paper is as follows. We introduce XDG in section 2. In section 3, we develop the new STDG as an instance of XDG. We briefly touch the issue of complexity in section 4, before section 5 concludes.
The well-formedness conditions of an XDG analysis are stipulated by parametrised principles. Each dimension \( d \) uses a set of principles \( P_d \), stipulating restrictions on the licensed graphs on \( d \). Principles can also pose restrictions on any number of dimensions simultaneously. An XDG analysis is well-formed only if all used principles are satisfied on all dimensions.

### 2.1 Principles

Principles are taken from a shared principle library. In the following, we present some of the principles from the current XDG principle library. We describe each principle using three boxes: the top one shows how to invoke the principle, the middle one explains the formal parameters, and the bottom one explains what the principle means.

#### 2.1.1 Tree

\[
\text{dag}(G) \quad G: \text{a graph dimension} \\
G \text{ is a tree.}
\]

#### 2.1.2 Directed acyclic graph

\[
\text{dag}(G) \quad G: \text{a graph dimension} \\
G \text{ is a directed acyclic graph.}
\]

#### 2.1.3 Out

\[
\text{out}(G_d, f) \\
G_d: \text{a graph dimension}, f: V \to (L_d \to 2^N): \text{an out specification} \\
The outgoing edges of each node in \( G_d \) must satisfy in label and number the stipulation of their out specification.
\]

The out principle restricts the number of outgoing edges with a certain label. In (1) below, the out specification of \( v_1 \) requires that the node has 1) no outgoing edges labeled \( l_1 \), 2) one outgoing edge labeled \( l_2 \), and 3) zero, one, two or three outgoing edges labeled \( l_3 \). Thus, the configuration in (1) satisfies the out principle: there is no outgoing edge labeled \( l_1 \), one labeled \( l_2 \), and three labeled \( l_3 \).

\[
(1) \\
\begin{array}{cccc}
& & & v_1 \\
& & v_2 & \\
& v_1 & & \\
\end{array}
\]

#### 2.1.4 In

\[
\text{in}(G_d, f) \\
G_d: \text{a graph dimension}, f: V \to (L_d \to 2^N): \text{an in specification} \\
The incoming edges of each node in \( G_d \) must satisfy in label and number the stipulation of their in specification.
\]

The in principle is the same as the out principle, but for incoming instead of outgoing edges.

#### 2.1.5 Covariance

\[
\text{covariance}(G_{d_1}, G_{d_2}, f) \\
f: V \to 2^{L_{d_1}}: \text{covariant labels specification} \\
Each edge \((v_1, l, v_2)\) in \( G_{d_1} \), where \( l \) is covariant on \( v_1 \), is only licensed if \( v_1 \) is above \( v_2 \) in \( G_{d_2} \).
\]

In (2) below, the edge \((v_1, l, v_2)\) in \( G_{d_1} \) is licensed because \( l \) is a covariant label on \( v_1 \), and \( v_1 \) is above \( v_2 \) in \( G_{d_2} \) (as indicated by the dashed edge).

\[
(2) \\
\begin{array}{ccc}
& v_2 & \\
& v_1 & & \\
\end{array}
\]
2.1.6 Contravariance

\[ \text{contravariance}(G_{d_1}, G_{d_2}, f) \]

\[ f : V \rightarrow 2^{L_{d_1}} : \text{contravariant labels specification} \]

Each edge \((v_1, l, v_2)\) in \(G_{d_1}\), where \(l\) is \textit{contravariant} on \(v_1\), is only licensed if \(v_2\) is above \(v_1\) in \(G_{d_2}\).

The contravariance principle is the converse of the covariance principle.

2.1.7 Linking

\[ \text{linking}(G_{d_1}, G_{d_2}, f_1, f_2) \]

\(G_{d_1}, G_{d_2} : \text{graph dimensions}, f_1 : V \rightarrow (L_{d_1} \rightarrow 2^{L_{d_2}}) : \text{realisation specification}, f_2 : V \rightarrow (L_{d_2} \rightarrow 2^{L_{d_2}}) : \text{substitution specification} \]

An edge \((v_1, l, v_2)\) in \(G_{d_1}\) is only licensed if either:

1. there is a corresponding edge \((v_1, l', v_2)\) in \(G_{d_2}\) and \(v_1 \text{ realises } l\) by \(l'\), or
2. there is an edge \((v_2, l'', v_3)\) in \(G_{d_2}\) and \(v_3 \text{ substitutes } l'\) by \(l''\) and \(v_1 \text{ realises } l\) by \(l'\).

Below, (3) shows a configuration licensed by the first clause of the linking principle: edge \((v_1, l, v_2)\) in \(G_{d_1}\) is licensed because there is a corresponding edge \((v_1, l', v_2)\) in \(G_{d_2}\) and the realisation specification of \(v_1\) specifies that the node can realise \(l\) as \(l'\), i.e. \(l' \in f_1(v_1)(l)\).

(4) displays a configuration licensed by the second clause: edge \((v_1, l, v_2)\) in \(G_{d_1}\) is licensed because there is an edge \((v_2, l'', v_3)\) in \(G_{d_2}\), \(v_3\) substitutes \(l'\) by \(l''\) (i.e. \(l'' \in f_2(v_3)(l')\)) and \(v_1\) realises \(l\) by \(l'\).

2.2 Lexicalisation

By lexicalisation, we can turn the graph description language described above into a powerful dependency-based meta grammar formalism. To this end, we assume a 1:1-correspondence between nodes and words, and assign to each node one of the possible lexical entries for the corresponding word.

The result of lexicalisation is that we can use the lexicon to specify all the features referred to in the principles. For instance lexicalising the out specification of the out principle gives us valency. Lexicalising the realisation and substitution specifications of the linking principle allow us to model lexicalised mappings from thematic roles to grammatical functions which realise them.

3 Syntax-semantics interface

In this section, we introduce the new grammar formalism of Semantic Topological Dependency Grammar (STDG). STDG has a syntax-semantics interface to underspecified semantics in CLLS. Note that CLLS is just one of many possible choices for the target semantics formalism.

3.1 CLLS

CLLS is based on dominance constraints (Marcus et al., 1983). CLLS structures describe \(\lambda\)-terms. In (6) below, we show a CLLS structure describing the strong reading of sentence (5):

\[ A \text{ woman, every man seems to love.} \]
Here, the @ symbol stands for functional application, lam for a lambda binder, var for a variable, and the dashed arrows represent lambda bindings.

3.2 STDG

STDG employs four graph dimensions: $G_{ID}$, $G_{LP}$, $G_{TH}$, $G_{DE}$. As in TDG, the ID (Immediate Domi-
nance) dimension models syntactic dependencies, and its edge labels are grammatical functions like subj (subject), obj (object) and vinf (verb infinitive). The LP (Linear Precedence) dimension models word order, and its edge labels are topologi-
cal fields (linear positions) like topf (topicalisation field), subjf (subject field) and vcf (verbal comple-
ment field). We use the same principles as in (Duchier and Debusmann, 2001).

We introduce the TH (THe matic) dimension to model semantic argument structure. Its edge labels are thematic roles like act (actor), pat (pa-
tient) and prop (proposition). We use the DE (DErivation) dimension to model the derivation of a CLLS structure, with edge labels like s (scope), and r (restriction).

On the TH dimension, we invoke the following principles and parameters: $\text{dag}(G_{TH})$, $\text{in}(G_{TH}, \text{in}_{TH})$, $\text{out}(G_{TH}, \text{out}_{TH})$, and linking($G_{TH}, G_{ID}$, real, subs). The dag principle ensures that every analysis on the TH dimension is a directed acyclic graph. It cannot be a tree since we also want to properly handle control constructions like Mary persuades Peter to sleep., where the controlled noun Peter has more than one mother in the argument structure (here, it is the patient of persuades and the actor of sleep). Moreover, the in principle assigns possible thematic roles to words, the out principle models valency, and the linking principle ensures that thematic roles are realised by the appropriate grammatical functions on the ID dimension (e.g. actors are realised by subjects).

On the DE dimension, we use the following principles and parameters: $\text{tree}(G_{DE})$, $\text{in}(G_{DE}, \text{in}_{DE})$, $\text{out}(G_{DE}, \text{out}_{DE})$, $\text{covariance}(G_{DE}, G_{ID}, \text{co})$, and $\text{contravariance}(G_{DE}, G_{ID}, \text{contra})$.

Below, we give an example STDG analysis of the strong reading of sentence (5).\(^1\) (7) displays the ID and LP dimensions, and (8) the TH and DE dimensions.

\(^1\)Note that for simplicity, we treat a woman, every man and to love each as a word.
a woman the patient of to love. The linking principle establishes the correct mappings of of actor to subject, and patient to object.

The DE analysis states that every man is in the scope of a woman, seems is in the scope of every man, and to love in the scope of seems. The covariance principle establishes that the edge (seems, s, love) is covariant: we state in the lexicon that seems is covariant on s (scope) labels, and hence the edge corresponds to the edge (seems, vinf, love) on the ID dimension which has the same direction. Conversely, we use the contravariance principle to ensure that the edge (every man, s, seems) is contravariant: in the lexicon, we state that every man is contravariant on s, and hence we license the analysis where the edge corresponds to the edge (seems, subj, every man) on the ID dimension which has the opposite direction.

3.3 From STDG to CLLS

To build the CLLS representation of the STDG analysis, we assume a lexicalised mapping from words in the sentence to CLLS fragments, as illustrated in (9) below:

(9)

We use 1) the DE analysis to plug these fragments together, and 2) the TH analysis to recover the lambda bindings. We show an example of this in (10) and (11):

(10)

(11)

(10) repeats the TH and DE analyses of sentence (5). The DE analysis determines how to plug the four fragments together: every man is in the scope of a woman, seems is in the scope of every man, and to love in the scope of seems. The TH analysis determines the lambda bindings: every man is the actor and a woman the patient of to love. The edge (seems, prop, to love) does not correspond to a lambda binding. We show the resulting CLLS analysis in (11).

Note that in the example shown, we derive a fully specified CLLS representation of the semantics from a full STDG parse. We can however obtain underspecified CLLS representations from partial STDG parses. E.g. we can decide not to enumerate the solutions on the DE dimension to get a CLLS analysis underspecified with respect to scope.
4 Complexity

Due to its generality, the complexity of XDG itself cannot be determined. This is not a problem: we have intentionally developed XDG as a meta grammar formalism, and the goal is to find tractable instances of XDG for which complexity results can be established and efficient parsers can be found.

TDG seems to be one of these tractable instances. (Koller and Striegnitz, 2002) have shown that the parsing complexity of TDG is NP-complete. But parsing times of the constraint-based TDG parser, and also of the new, generalised XDG parser (using a TDG instance) suggest that parsing is polynomial in the average-case. So far, we have only confirmed this for small test grammars, but we are currently working on getting results for bigger grammars.

Denys Duchier, Alexander Koller, Marco Kuhlmann, Stefan Thater and me (p.c.) have established an equivalence between a restricted variant of TDG and Tree Insertion Grammars (TIG) (Schabes and Waters, 1994). TIG is itself a restricted variant of Tree Adjoining Grammars (TAG) (Joshi, 1987).

We have also found out that TDG and TAG are not equivalent. It will be interesting to see whether we can find an instance of XDG (probably similar to TDG) which falls into the same class of mildly context-sensitive languages such as TAG and others (e.g. CCG (Steedman, 2000)).

As of yet, we have no complexity results for STDG.

5 Conclusions

We generalised TDG to get Extensible Dependency Grammar (XDG), a graph description language that can be turned into a powerful meta grammar formalism by lexicalisation. We used XDG as a launchpad for the new grammar formalism of Semantic Topological Dependency Grammar (STDG). STDG is an instance of XDG that allows compositional semantics construction.

XDG is increasingly used by other linguists: (Duchier and Kruijff, 2003) create an interface to information structure, and (Korthals, 2003) induces word order constraints from the German TIGER corpus. (Dienes et al., 2003) use statistical information to guide the XDG parser using A* search.

In the future, our plans include the integration of preferences (e.g. PP attachment). We are also continuing our search for equivalences between instances of XDG and existing grammar formalisms such as TAG, and we are working on creating bigger grammars.

References


