Extensible Dependency Grammar: Modular Linguistic Modeling Through Intersection

Ralph Debusmann

Programming Systems Lab, Saarbrücken, Germany

Charles University, Prague, December 5, 2006
What this talk is about

- Extensible Dependency Grammar (XDG)
- new grammar formalism for natural language
- explores the combination of:
  1. dependency grammar
  2. model theory
  3. parallel architecture
- results:
  1. modularity: grammars can be extended by any linguistic aspect, each modeled independently
  2. emergence: complex linguistic phenomena emerge as the intersection of the linguistic aspects
Overview

1. Introduction
2. Formalization
3. Implementation
4. Application
5. Conclusions
Dependecy Grammar

- traditional (Chomsky 1957): syntax of natural language analyzed in terms of phrase structure grammar:
  - hierarchically arranges substrings called phrases
  - nodes labeled by syntactic categories

```
S
/|\
|NP|VP|
/|
|Det|N  |V|
/|
|Every|baby|wants|
/|
|Part|VP|
  |
|to|V|
  |eat|
```
dependency grammar (Tesnière 1959):
- hierarchically arranges words
- edges labeled by grammatical functions
- mothers: heads, daughters: dependents

```
det  subj  vinf
  det
    det
      Every
```

```
sbj
  subj
    subj
      baby
```

```
vinf
  vinf
    vinf
      wants
```

```
part
  part
    part
      to
```

```
eat
  eat
    eat
```
Advantages

- flexibility: dependency analyses need not be trees but can be arbitrary graphs
- need not be ordered
- perfectly suited for modeling linguistic aspects other than syntax, e.g. predicate-argument structure, where the models are unordered DAGs
traditional: generative perspective on grammar (Chomsky 1957):

1. start with the empty set
2. use production rules to generate the well-formed models
Model Theory

- traditional: generative perspective on grammar (Chomsky 1957):
  1. start with the empty set
  2. use production rules to generate the well-formed models
Model Theory

- traditional: generative perspective on grammar (Chomsky 1957):
  1. start with the empty set
  2. use production rules to generate the well-formed models
Model Theory

- traditional: generative perspective on grammar (Chomsky 1957):
  1. start with the empty set
  2. use production rules to generate the well-formed models
Model Theory

- traditional: generative perspective on grammar (Chomsky 1957):
  1. start with the empty set
  2. use production rules to generate the well-formed models

- Every baby wants to eat
model theory: eliminative perspective (Rogers 1996):

1. start with the set of all possible models
2. use well-formedness conditions to eliminate all non-well-formed models
model theory: eliminative perspective (Rogers 1996):

1. start with the set of all possible models
2. use well-formedness conditions to eliminate all non-well-formed models
model theory: eliminative perspective (Rogers 1996):

1. start with the set of all possible models
2. use well-formedness conditions to eliminate all non-well-formed models
model theory: eliminative perspective (Rogers 1996):
1. start with the set of all possible models
2. use well-formedness conditions to eliminate all non-well-formed models
model theory: eliminative perspective (Rogers 1996):

1. start with the set of all possible models
2. use well-formedness conditions to eliminate all non-well-formed models
Advantage

- declarativity: constraints describe the well-formed models independently of any underlying mechanisms
Parallel Architecture

- traditional: syntacto-centric architecture (Chomsky 1965):
  - only syntax modeled independently
  - other linguistic aspects obtained by functional interfaces

Phonology

Syntax

Semantics

syntactic well-formedness

interface

interface

interface
parallel architecture (Jackendoff 2002), (Sadock 1991):
  - all linguistic aspects modeled independently
  - relational interfaces

phonological well-formedness

syntactic well-formedness

semantic well-formedness

Phonology

Syntax

Semantics
Advantages

- modularity: linguistic aspects can be modeled largely independently of each other
- emergence: complex phenomena emerge as the intersection of the linguistic aspects
Extensible Dependency Grammar (XDG) combines:

1. flexibility from dependency grammar
2. declarativity from model theory
3. modularity and emergence from the parallel architecture

- models: dependency multigraphs, i.e. tuples of dependency graphs
- share the same set of nodes
- arbitrary many components called dimensions
Example Multigraph

SYN

Every  baby  wants  to  eat

SEM

Every  baby  wants  to  eat
Related Work

- phrase structure grammar:
  - Tree Adjoining Grammar (Joshi 1987)
  - Combinatory Categorial Grammar (Steedman 2000)
  - Head-driven Phrase Structure Grammar (Pollard/Sag 1994)
  - Lexical-Functional Grammar (Bresnan 2001)

- dependency grammar:
  - Functional Generative Description (Sgall et al. 1986)
  - Meaning Text Theory (Mel’čuk 1988)
  - Constraint Dependency Grammar (Menzel/Schröder 1998)
  - Topological Dependency Grammar (Duchier/Debusmann 2001)
Overview

1. Introduction
2. Formalization
3. Implementation
4. Application
5. Conclusions
Dependency Multigraphs

- tuples \((V, D, W, w, L, E, A, a)\)

**SYN**
- Every
- baby
- wants
- to
- eat

**SEM**
- Every
- baby
- wants
- to
- eat
3 relations:
1. labeled edge
2. strict dominance
3. precedence

SYN

SEM
Extensible Dependency Grammar

Formalization

Dependency Multigraphs

Grammar

\[ G = (MT, P), \] characterizes set of multigraphs:

1. \( MT \): multigraph type determining dimensions, words, edge labels
2. \( P \): set of principles constraining the set of well-formed multigraphs of type \( MT \)

- principles \( P \) formulated in a higher order logic
- signature determined by \( MT \)
the models of $G = (MT, P)$ are all multigraphs which:
1. have multigraph type $MT$
2. satisfy all principles $P$

the string language of a grammar $G$ are all strings $s$ such that:
1. there is a model of $G$ with as many nodes as words in $s$
2. concatenation of the words of the nodes yields $s$
formulas in higher order logic

characterize the well-formed multigraphs of a specific multigraph type

predefined principle library from which grammars can be built like with lego bricks (Debusmann et al. 2005 FGMOL), e.g.:

- tree principle
- valency principle
- order principle
Tree Principle

- given a dimension $d$, there must be:
  1. no cycles
  2. precisely one node without an incoming edge (the root)
  3. each node must have at most one incoming edge

\[
\forall v : \neg (v \xrightarrow{d^+} v) \\
\exists^1 v : \neg \exists v' : v' \xrightarrow{d} v \\
\forall v : \neg \exists v' : v' \xrightarrow{d} v \lor \exists^1 v' : v' \xrightarrow{d} v
\]
Valency Principle

- lexically constrains the incoming and outgoing edges of the nodes
- characterized by fragments, e.g.:

```
ID

a

a?
                        b!
                  c!

ID

b

ID

c
```
Grammar 1

- together with the tree principle, the fragments yield our first grammar
- string language: equally many as, bs and cs in any order:

\[ L_1 = \{ w \in (a \cup b \cup c)^+ \mid |w|_a = |w|_b = |w|_c \} \]

- why? as arranged in a chain, each a has precisely one outgoing edge to b and one to c:
Example Analyses

Extensible Dependency Grammar

- Formalization
- Principles

ID

a?

b!

c!

,  

ID

b!

b

,  

ID

c!

c

ID

⇓

ID

b

,  

ID

c

ID

1 2 3

b c a

ID
Example Analyses
Order Principle

- lexically constrains:
  1. the order of the outgoing edges of the nodes depending on their edge labels
  2. the order of the mother with respect to the outgoing edges, also depending on their edge labels

- characterized by ordered fragments, e.g.:

```
< 1 < 2 < 3
```

```
_ L P 

↑ < 1 < 2 < 3
```

```
_ a 

1* 2+ 3+
```
Grammar 2

- String language: one or more $a$ followed by one or more $b$s followed by one or more $c$s:

$$L_2 = \{ w \in a^+ b^+ c^+ \}$$

- Tree, valency and order principles and the fragments below:

- Idea: $a$ is always root, licensing zero or more outgoing edges labeled 1 to $a$s, and one or more labeled 2 to $b$s and 3 to $c$s, where the $a$s precede the $b$s precede the $c$s:
Example Analyses
Example Analyses
Intersection of Dimensions

The intersection of the two languages $L_1$ and $L_2$ yields the string language of $n$ $a$s followed by $n$ $b$s followed by $n$ $c$s:

$$L_1 \cap L_2 = \{ w \in a^n b^n c^n \mid n \geq 1 \}$$

modeled by intersecting dimensions:

1. **ID** dimension of grammar 1 ensures that there are equally many $a$s, $b$s and $c$s
2. **LP** dimension of grammar 2 orders the $a$s before the $b$s before the $c$s
Example Analyses
Example Analyses
Scrambling

- German subordinate clauses: nouns followed by the verbs:

  \[(dass) \text{ ein Mann Cecilia die Nilpferde füttern sah}\]
  \[(that) \text{ a man Cecilia the hippos feed saw}\]
  “(that) a man saw Cecilia feed the hippos”

- all permutations of the nouns grammatical, i.e., also:

  \[(dass) \text{ ein Mann die Nilpferde Cecilia füttern sah}\]
  \[(dass) \text{ die Nilpferde ein Mann Cecilia füttern sah}\]
  \[(dass) \text{ die Nilpferde Cecilia ein Mann füttern sah}\]
  \[(dass) \text{ Cecilia ein Mann die Nilpferde füttern sah}\]
  \[(dass) \text{ Cecilia die Nilpferde ein Mann füttern sah}\]
Idealization

- idealized language:

\[
SCR = \{ \sigma(n^{[1]}, \ldots, n^{[k]})v^{[k]} \ldots v^{[1]} \mid k \geq 1 \text{ and } \sigma \text{ a permutation} \}
\]

- grammar: ID dimension pairs verbs and nouns, LP dimension orders nouns before verbs
Example Analyses
Example Analyses

Extensible Dependency Grammar

Formalization

Principles
Expressivity

- can model lexicalized context-free grammar (constructive proof in thesis)
- can go far beyond context-free grammar:
  - $a^n b^n c^n$ already non-context free
  - can model TAG (Debusmann et al. 2004 TAG+7): mildly context-sensitive
  - cross-serial dependencies (thesis): also mildly context-sensitive
  - scrambling: beyond the mildly context-sensitive Linear Context-Free Rewriting Systems (LCFRS) (Becker et al. 1992)
- put to use in an elegant account of German word order phenomena in (Duchier/Debusmann 2001), (Debusmann 2001), (Bader et al. 2004)
Complexity

- restrictions on principles:
  - first-order: upper bound in PSPACE
  - polynomially testable: upper bound in NP
- all principles written so far first-order
- all principles implemented as polynomially testable constraints in Mozart/Oz
- i.e., practical upper bound: in NP
Overview

1. Introduction
2. Formalization
3. Implementation
4. Application
5. Conclusions
how to process an NP-hard problem?

constraint programming (Schulte 2002), (Apt 2003): solving of constraint satisfaction problems (CSPs)

CSPs stated in terms of:
1. constraint variables, here on finite sets of integers
2. constraints on them

solutions of a CSP determined by two interleaving processes:
1. propagation: application of deterministic inference rules
2. distribution: non-deterministic choice

XDG parsing regarded as a CSP in Mozart/Oz (Smolka 1995), based on techniques developed in (Duchier 1999, 2003)
Modeling Dependency Multigraphs

- dependency graphs: nodes identified with integers, each node associated with a set of finite set of integers variables, e.g.:

```
1 ↦
  
  eq = {1}
  mothers = {}
  up = {}
  daughters = {2, 3, 5}
  down = {2, 3, 4, 5, 6}
  ...
```

- dependency multigraphs: variables duplicated for each dimension
Modeling Principles

- principles can now be transformed into constraints on finite sets of integers
- e.g. the tree principle:

```plaintext
for Node in Nodes do
  %% no cycles
  {FS.disjoint Node.eq Node.down}

  %% one root
  {FS.card Roots}=:1

  %% at most one incoming edge
  {FS.card Node.mothers}=<:1
end
```
Constraint Parser

- concurrent: all dimensions processed in parallel
- reversible: can be used for parsing and generation (Koller/Striegnitz 2002), (Debusmann 2004)
- supports underspecification: e.g. of quantifier scope, PP attachment (Debusmann et al. 2004 COLING)
- efficient for handcrafted grammars
- first successful experiments in large-scale parsing with the XTAG grammar (> 100,000 lexical entries) after thesis submission
extensive grammar development kit (35000 code lines): XDG Development Kit (XDK) (Debusmann et al. 2004 MOZ)

elementary grammars (24000 additional lines):
  - German grammar developed in (Debusmann 2001)
  - Arabic grammar developed in (Odeh 2004)
  - toy grammars for Czech, Dutch and French
  - implementations of all example grammars in the thesis
  - imported XTAG grammar

graphical user interface

complete documentation (200+ pages)

application:
  - successfully used for teaching (ESSLLI 2004, FoPra)
  - module in an engine for interactive fiction (Koller et al. 2004)
Overview

1. Introduction
2. Formalization
3. Implementation
4. Application
5. Conclusions
English example grammar developed in the thesis models fragments of:
- syntax
- semantics
- phonology
- information structure

interfaces:
- relational syntax-semantics interface (Korthals/Debusmann 2002), (Debusmann et al. 2004 COLING)
- relational phonology-information structure interface (Debusmann et al. 2005 CICLING)
Syntax

- based on topological analysis of German
  (Duchier/Debusmann 2001)
- ID dimension: models grammatical functions
- LP dimension: models word order using topological fields
- intersection of ID/LP leads to the emergence of complex
  English word order phenomena:
  - topicalization
  - wh-questions
  - pied piping
Topicalization

ID

1 Mary
2 Peter
3 tries
4 to
5 find

LP

1 Mary
2 Peter
3 tries
4 to
5 find
PA dimension: models predicate-argument structure
SC dimension: models quantifier scope
supports scope underspecification
interface to the Constraint Language for Lambda Structures (CLLS) (Egg et al. 2001)
Example (Weak Reading)

PA

Every man loves a woman

SC

Every man loves a woman
Example (Strong Reading)

PA

Every
man
loves
a
woman

SC

Every
man
loves
a
woman
Example (Underspecification)

PA

Every man loves a woman

SC

Every man loves a woman
Phonology

- PS dimension: models prosody
- sentence divided into prosodic constituents marked by boundary tones
- prosodic constituents headed by pitch accents
Example

\[
\begin{align*}
\text{PS} &\quad Marcel_L+H^* &\quad proves_LH\% &\quad completeness_H^*LL\% &\quad . \\
1 &\quad \rightarrow &\quad 2 &\quad \rightarrow &\quad 3 &\quad \rightarrow &\quad 4 &\quad \rightarrow \\
\end{align*}
\]
Information Structure

- IS dimension
- sentence divided into information structural constituents using the theme/rheme dichotomy
- information structural constituents: divided into focus and background
Example

Extended Dependency Grammar

Application

Information Structure

IS  Marcel\_L+H*  proves\_LH\%  completeness\_H*\_*LL\%
Syntax-Semantics Interface

- relational interface between ID and PA dimensions
- modular modeling: independent of word order (LP) and quantifier scope (SC)
- intersection of ID/PA leads to the emergence of:
  - control/raising
  - auxiliary constructions (e.g. passives)
- supports underspecification of PP-attachment
Control/Raising and Passive Example

Peter seems to have been persuaded to sleep.
relational interface between PS and IS dimensions
modular modeling: independent of any other linguistic aspect
based on the prosodic account of information structure developed in (Steedman 2000)
intersection of PS and IS dimensions leads e.g. to the emergence of the unmarked theme ambiguity phenomenon
Unmarked Theme Example

PS  Marcel_LH%  proves  completeness_H\ast_LL%  

IS  Marcel_LH%  proves  completeness_H\ast_LL%  

Unmarked Theme Example

\[ \text{PS} \quad \text{Marcel}_LH\% \quad \text{proves} \quad \text{completeness}_H^*_{LL}\% \]

\[ \text{IS} \quad \text{Marcel}_LH\% \quad \text{proves} \quad \text{completeness}_H^*_{LL}\% \]
Overview

1. Introduction
2. Formalization
3. Implementation
4. Application
5. Conclusions
Summary

- with XDG, explored combination of dependency grammar, model theory and parallel architecture

formalization:
  - higher order logic
  - expressivity: far beyond context-free grammar
  - practical complexity: NP-complete

implementation:
  - parser based on constraint programming in Mozart/Oz
  - comprehensive grammar development kit (XDK)

application:
  - example grammar modeling fragments of natural language syntax, semantics, phonology and information structure

main results:
  1. new degree of modularity
  2. phenomena emerge by the intersection of individual dimensions, without further stipulation
Selected Publications

Ralph Debusmann, Denys Duchier, Alexander Koller, Marco Kuhlmann, Gert Smolka, and Stefan Thater.
A relational syntax-semantics interface based on dependency grammar.

Ralph Debusmann, Denys Duchier, Marco Kuhlmann, and Stefan Thater.
TAG as dependency grammar.
### Selected Publications II


Christian Korthals and Ralph Debusmann. 
Linking syntactic and semantic arguments in a dependency-based formalism. 
Ondrej Bojar. 
Problems of inducing large coverage constraint-based dependency grammar.

Peter Dienes, Alexander Koller, and Marco Kuhlmann. 
Statistical A* dependency parsing.

Alexander Koller and Kristina Striegnitz. 
Generation as dependency parsing.
Selected Publications by Other Authors II

Christian Korthals.
Diploma thesis.

Pierre Lison.
Implémentation d’une interface sémantique-syntaxe basée sur des grammaires d’unification polarisées.

Jorge Pelizzoni and Maria das Gracas Volpe Nunes.
N:M mapping in XDG - the case for upgrading groups.
Future Work

- **formalization:**
  - strengthen relation to other grammar formalisms, e.g. FGD, TAG
  - formalize XDG in a weaker logic, e.g. MSO
- **implementation:**
  - make use of new constraint technology, e.g. Gecode (Schulte/Tack 2005), (Schulte et al. 2006)
  - automatically generate principle implementations
  - integration of ideas for segmentation (Kubon et al. 2006)
Thank you!
References I

Krzysztof R. Apt.  
*Principles of Constraint Programming.*  

Tilman Becker, Owen Rambow, and Michael Niv.  
The derivational generative power, or, scrambling is beyond LCFRS.  

Joan Bresnan.  
*Lexical Functional Syntax.*  

Noam Chomsky.  
*Syntactic Structures.*  

Noam Chomsky.  
*Aspects of the Theory of Syntax.*  

Noam Chomsky.  
*Lectures on Government and Binding: The Pisa Lectures.*  
References II

Noam Chomsky. 
*The Minimalist Program.*

Denys Duchier. 
Axiomatizing dependency parsing using set constraints. 

Denys Duchier. 
Configuration of labeled trees under lexicalized constraints and principles. 

Markus Egg, Alexander Koller, and Joachim Niehren. 
The Constraint Language for Lambda Structures. 

Ray Jackendoff. 
*Foundations of Language.*

Aravind K. Joshi. 
An introduction to tree-adjoining grammars. 
References


References IV

Christian Schulte and Guido Tack.
Views and iterators for generic constraint implementations.
In Christian Schulte, Fernando Silva, and Ricardo Rocha, editors,
Proceedings of the Fifth International Colloquium on Implementation of
Constraint and Logic Programming Systems, pages 37–48, Sitges/ES,
2005.

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

Gert Smolka.
The Oz programming model.
In Jan van Leeuwen, editor, Computer Science Today, Lecture Notes in
Computer Science, vol. 1000, pages 324–343. Springer-Verlag, Berlin/DE,
1995.

Mark Steedman.
The Syntactic Process.

Guido Tack, Christian Schulte, and Gert Smolka.
Generating propagators for finite set constraints.
In Frédéric Benhamou, editor, 12th International Conference on Principles
and Practice of Constraint Programming, volume 4204 of Lecture Notes in

Lucien Tesnière.
Eléments de Syntaxe Structurale.