Welcome to my colloquium titled Extensible Dependency Grammar, a modular grammar formalism based on multigraph description.

Extensible Dependency Grammar: A Modular Grammar Formalism Based On Multigraph Description

Ralph Debusmann

Programming Systems Lab, Saarbrücken, Germany

Promotionskolloquium, November 3, 2006

What the thesis is about

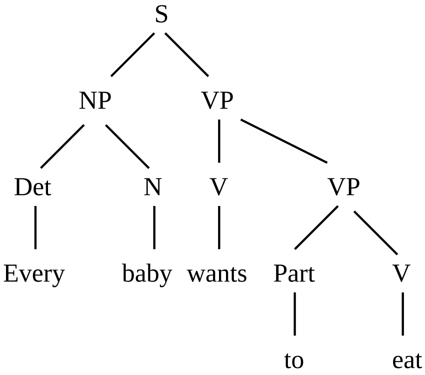
- Extensible Dependency Grammar (XDG)
- new grammar formalism for natural language
- explores the combination of:
  1. dependency grammar
  2. model theory
  3. parallel architecture
- results:
  1. modularity: grammars can be extended by any linguistic aspect, each modeled independently
  2. emergence: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

Overview

Overview

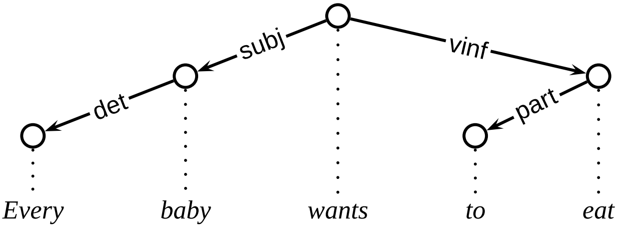
Dependency Grammar

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



Dependency Grammar

- dependency grammar (Tesnière 1959):
  - hierarchically arranges words
  - edges labeled by grammatical functions
  - mothers: heads, daughters: dependents

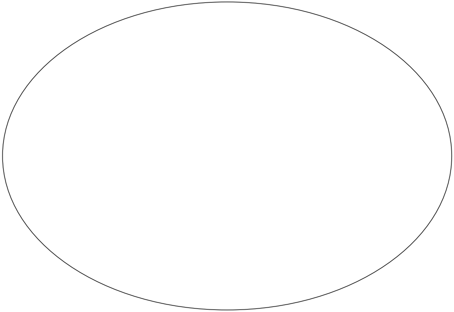


Advantages

- 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

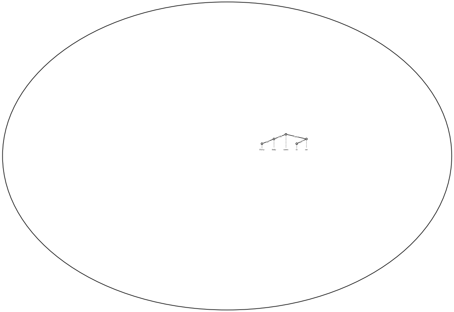
Model Theory

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



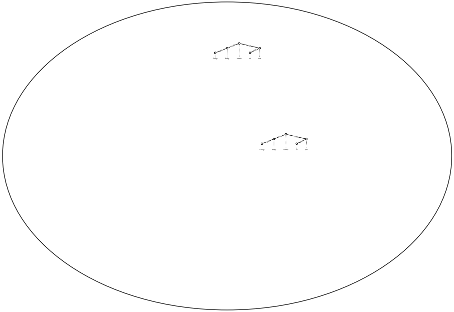
Model Theory

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



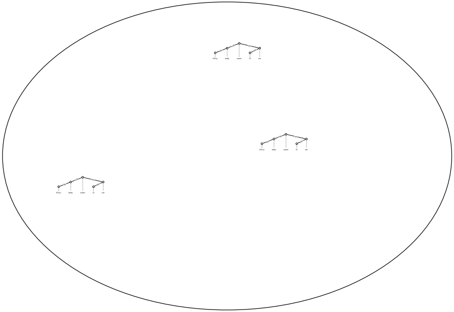
Model Theory

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



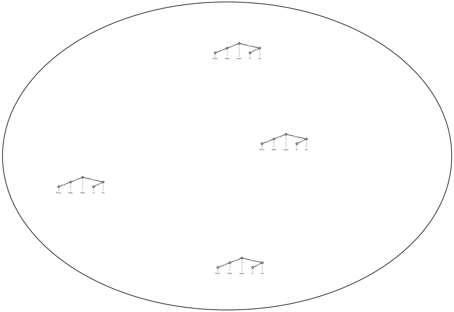
Model Theory

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



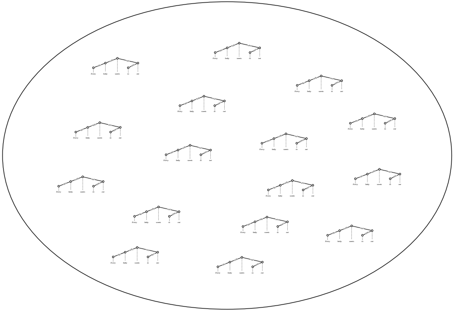
Model Theory

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



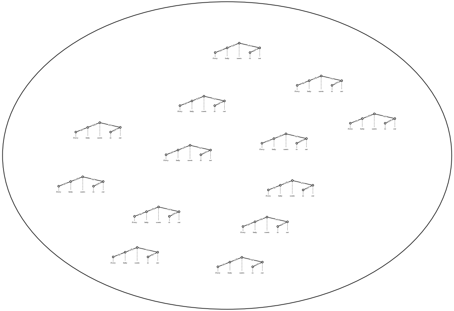
Model Theory

- 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



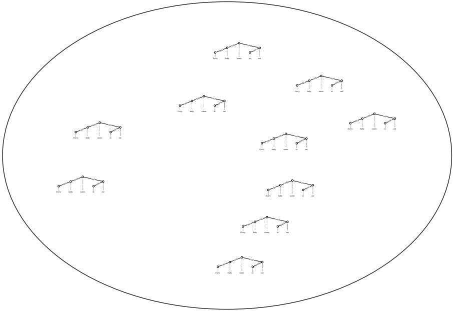
Model Theory

- 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



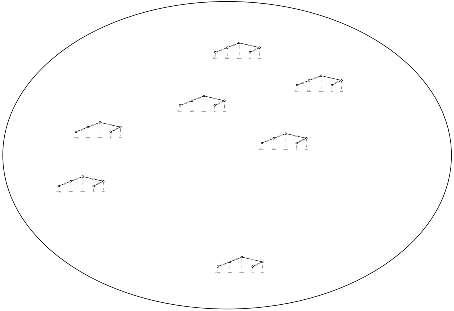
Model Theory

- 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



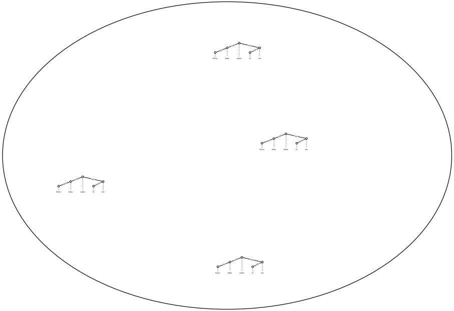
Model Theory

- 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

- 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

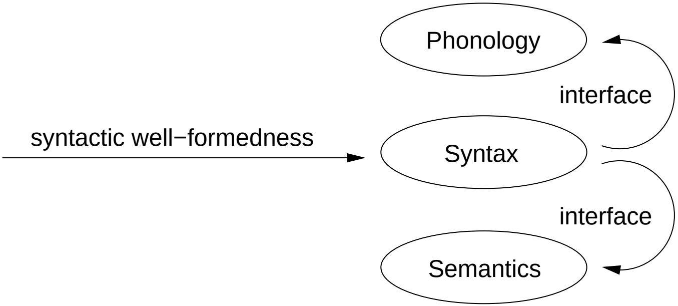


Advantage

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

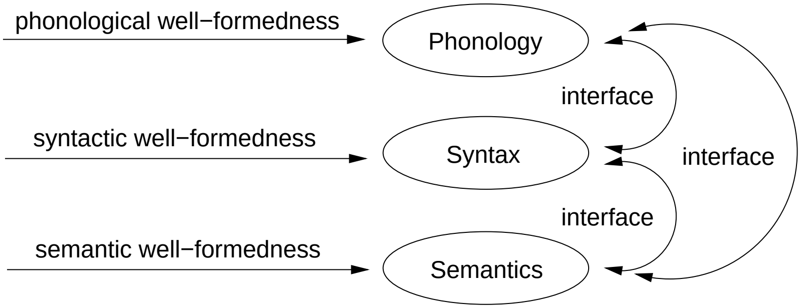
Parallel Architecture

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



Parallel Architecture

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



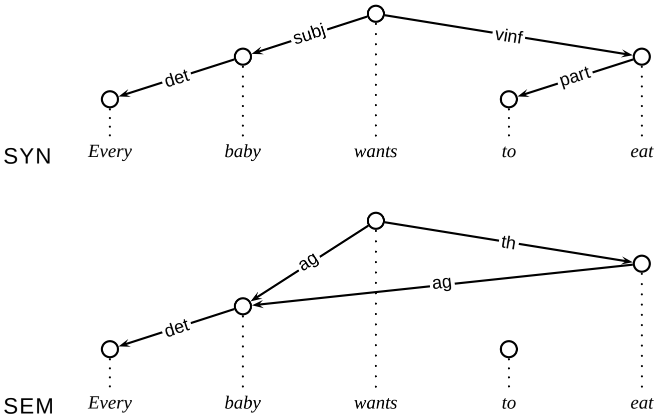
Advantages

- 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



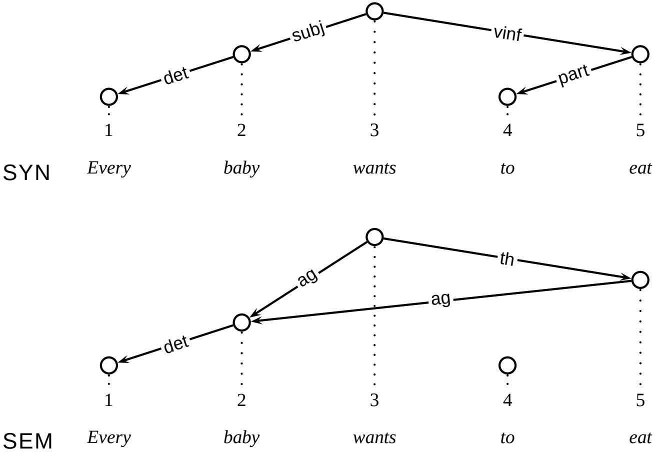
Related Work

- phrase structure grammar:
  - Generative Grammar (Chomsky 1957, 1965, 1981, 1995)
  - 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

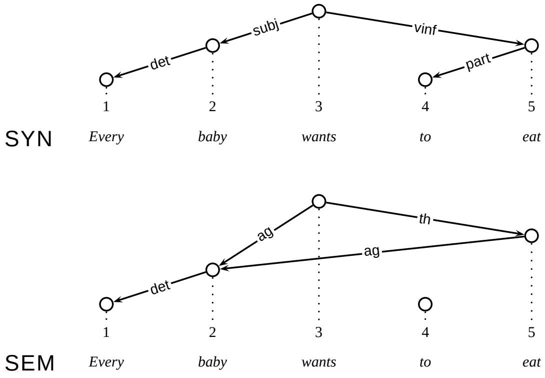
Dependency Multigraphs

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



Relations

- 3 relations:
  1. labeled edge
  2. strict dominance
  3. precedence



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

Models and String Language

- 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

Principles

- 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

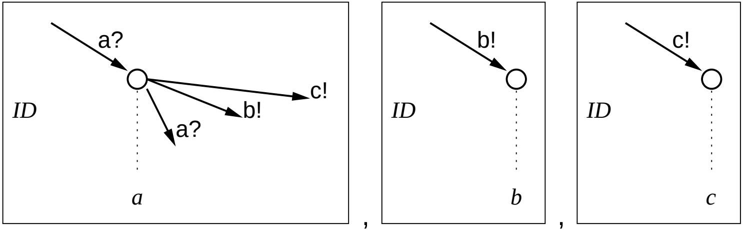
Tree Principle

- 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

$$
\begin{array}{ll}
\forall v : \neg(v \longrightarrow_d^+ v) & \wedge \\
\exists^1 v : \neg\exists v' : v' \longrightarrow_d v & \wedge \\
\forall v : \neg\exists v' : v' \longrightarrow_d v \ \vee \ \exists^1 v' : v' \longrightarrow_d v &
\end{array}
$$

Valency Principle

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

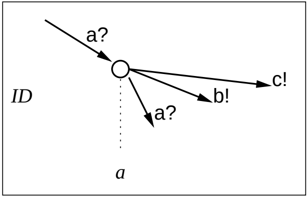


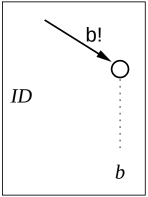
Grammar 1

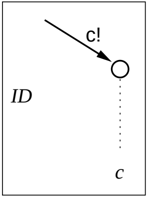
- together with the tree principle, the fragments yield our first grammar
- string language: equally many as, bs and cs in any order:

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

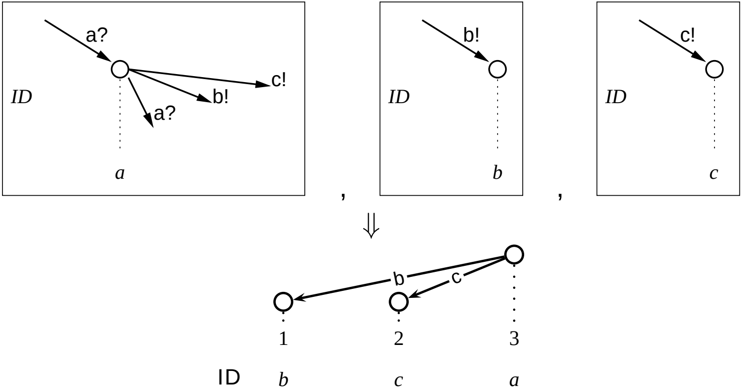
- why? as arranged in a chain, each a has precisely one outgoing edge to b and one to c:



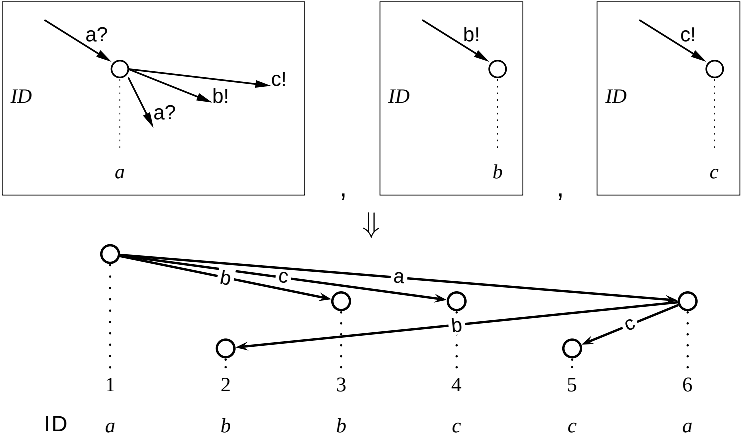
,


,


Example Analyses

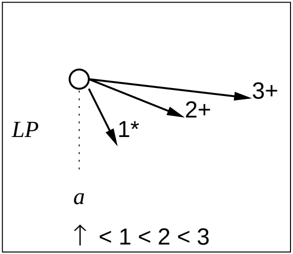


Example Analyses



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.:

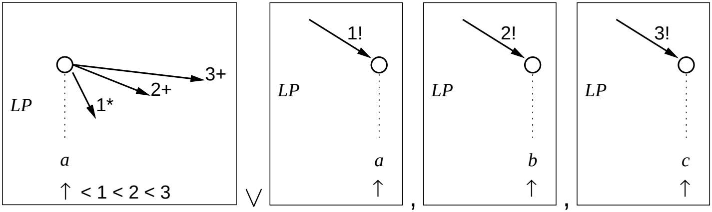


Grammar 2

- string language: one or more a followed by one or more bs followed by one or more cs:

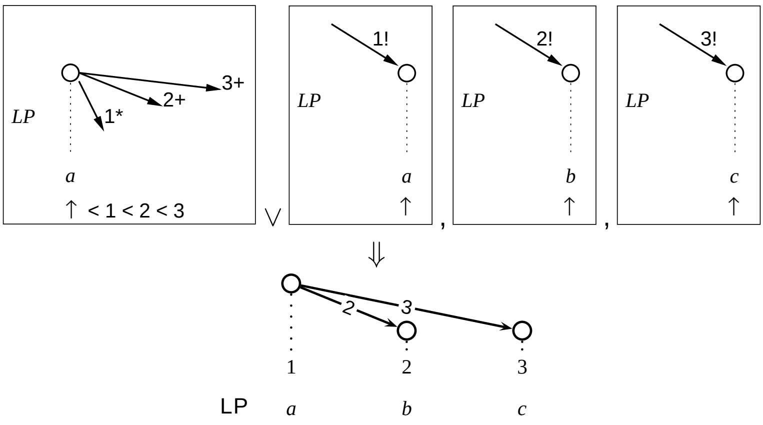
$$L_2 = \{w \in a^+ b^+ c^+\}$$

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

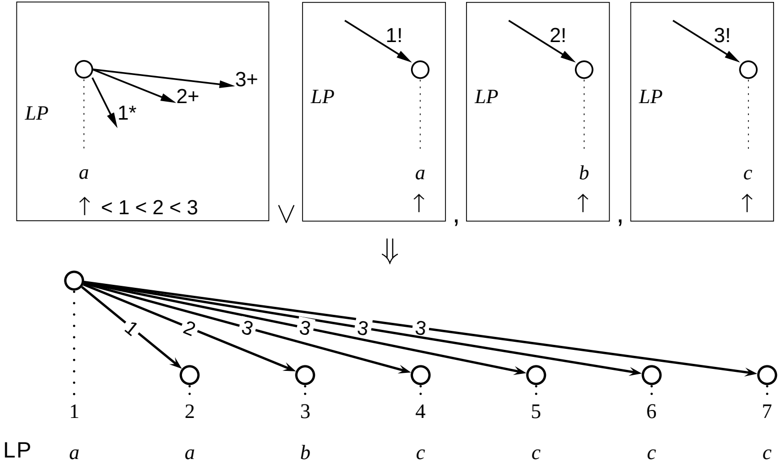


- idea: a is always root, licensing zero or more outgoing edges labeled 1 to as, and one or more labeled 2 to bs and 3 to cs, where the as precede the bs precede the cs:

Example Analyses



Example Analyses



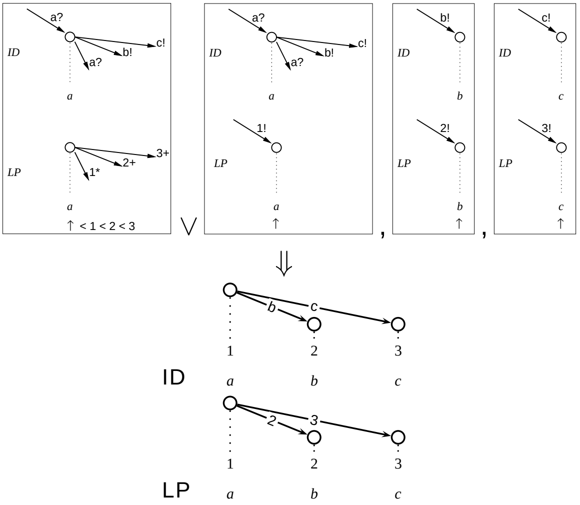
Intersection of Dimensions

- intersection of the two languages L_1 and L_2 yields the string language of n as followed by n bs followed by n cs:

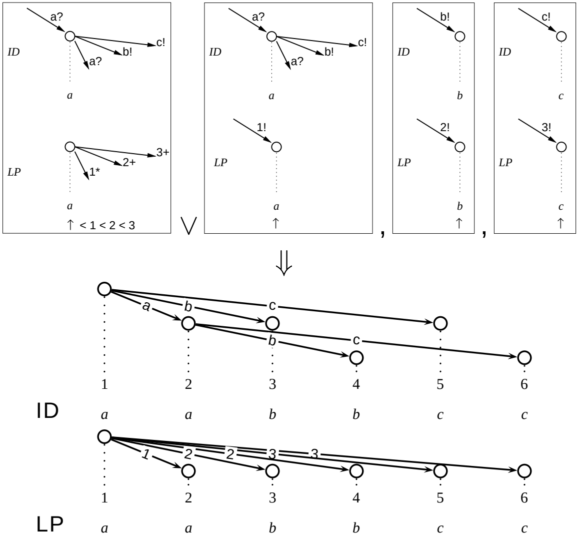
$$L_1 \cap L_2 \quad = \quad \{w \in a^n b^n c^n \mid n \geq 1\}$$

- modeled by intersecting dimensions:
  1. ID dimension of grammar 1 ensures that there are equally many as, bs and cs
  2. LP dimension of grammar 2 orders the as before the bs before the cs

Example Analyses



Example Analyses



Scrambling

- German subordinate clauses: nouns followed by the verbs:

(dass)	*ein*	*Mann*	*Cecilia*	*die*	*Nilpferde*	*füttern*	*sah*
(that)	*a*	*man*	*Cecilia*	*the*	*hippos*	*feed*	*saw*

“(that) a man saw Cecilia feed the hippos”

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

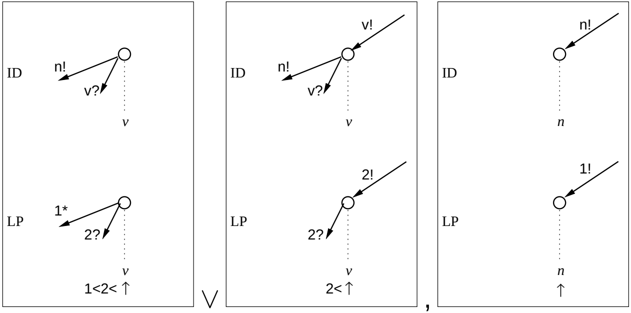
(dass) ein Mann die Nilpferde Cecilia füttern sah
(dass) die Nilpferde ein Mann Cecilia füttern sah
(dass) die Nilpferde Cecilia ein Mann füttern sah
(dass) Cecilia ein Mann die Nilpferde füttern sah
(dass) Cecilia die Nilpferde ein Mann füttern sah

Idealization

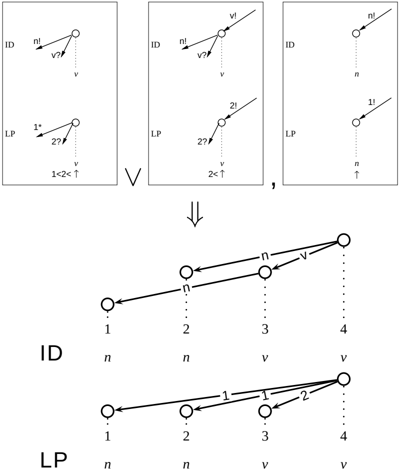
- idealized language:

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

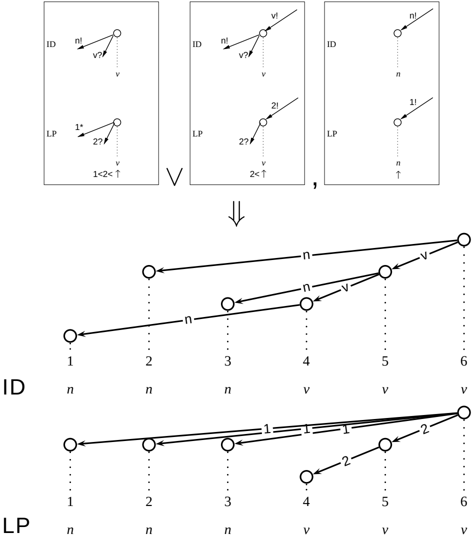
- grammar: ID dimension pairs verbs and nouns, LP dimension orders nouns before verbs



Example Analyses



Example Analyses



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

- recognition problem: NP-hard (reduction to SAT in thesis) (Debusmann/Smolka 2006)
- 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

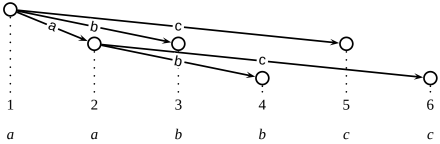
Overview

Implementation

- 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.:



$$1 \mapsto \left\{ \begin{array}{rcl} eq & = & \{1\} \\ mothers & = & \{\} \\ up & = & \{\} \\ daughters & = & \{2,3,5\} \\ down & = & \{2,3,4,5,6\} \\ \dots & & \end{array} \right\}$$

- 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:

```
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
```

Features

- 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

Grammar Development Kit

- extensive grammar development kit built around the constraint parser (35000 code lines): XDG Development Kit (XDK) (Debusmann et al. 2004 MOZ)
- example 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
- 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

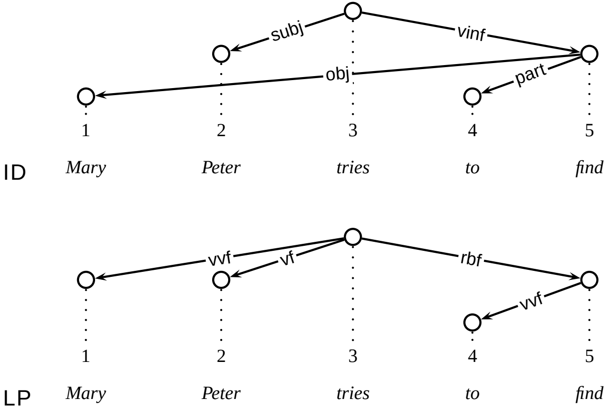
Application to Natural Language

- 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

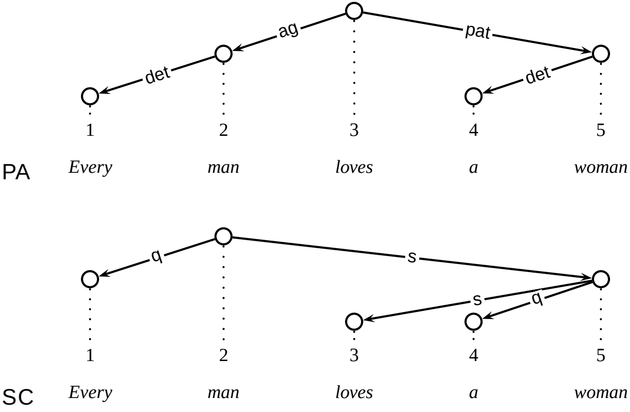
Topicalization



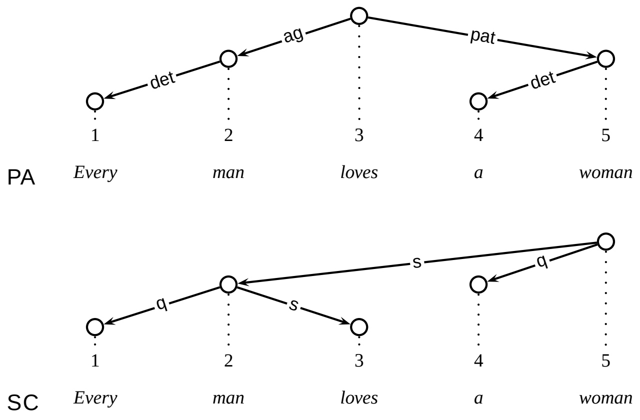
Semantics

- 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)

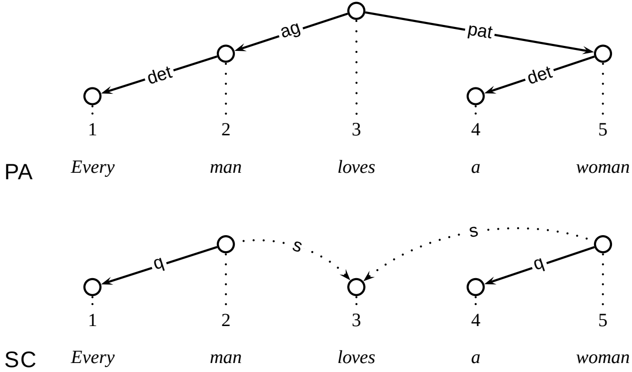
Example (Weak Reading)



Example (Strong Reading)



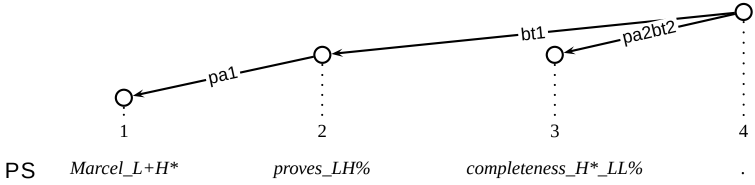
Example (Underspecification)



Phonology

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

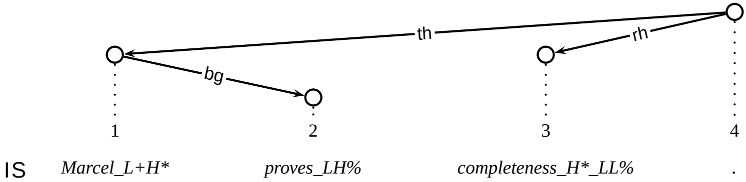
Example



Information Structure

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

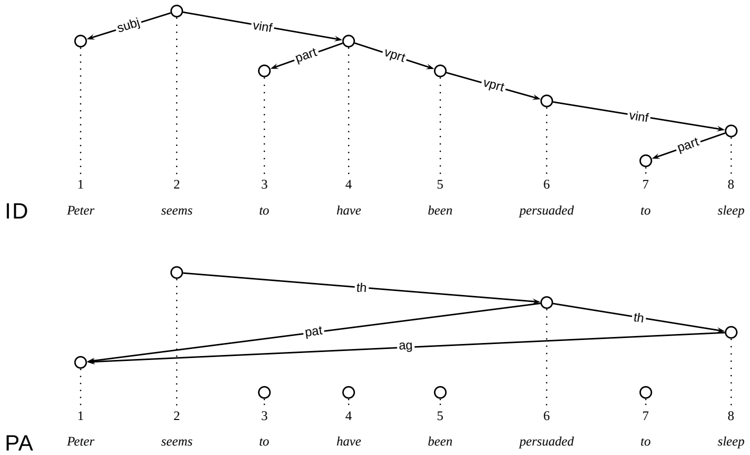
Example



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

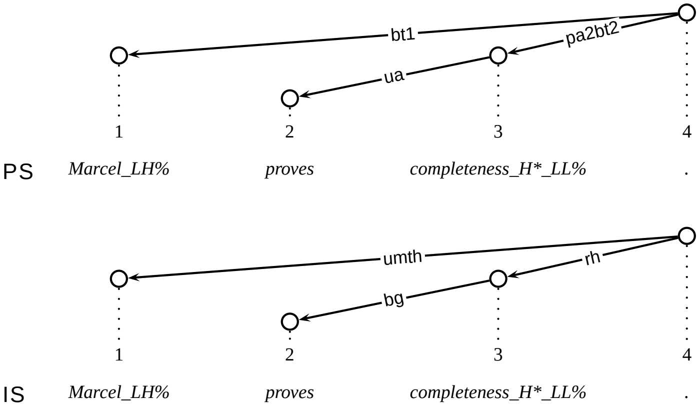
Control/Raising and Passive Example



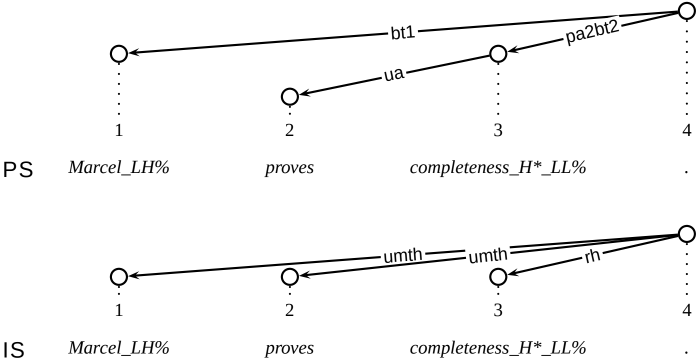
Phonology-Semantics Interface

- 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



Unmarked Theme Example



Overview

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 I

- Ralph Debusmann, Denys Duchier, Alexander Koller, Marco Kuhlmann, Gert Smolka, and Stefan Thater.
  A relational syntax-semantics interface based on dependency grammar.
  In *Proceedings of COLING 2004*, Geneva/CH, 2004.
- Ralph Debusmann, Denys Duchier, Marco Kuhlmann, and Stefan Thater.
  TAG as dependency grammar.
  In *Proceedings of TAG+7*, Vancouver/CA, 2004.

Selected Publications II

- Ralph Debusmann, Denys Duchier, and Joachim Niehren.
  The XDG grammar development kit.
  In *Proceedings of the MOZ04 Conference*, volume 3389 of *Lecture Notes in Computer Science*, pages 190–201, Charleroi/BE, 2004. Springer.
- Ralph Debusmann, Denys Duchier, and Andreas Rossberg.
  Modular grammar design with typed parametric principles.
  In *Proceedings of FG-MOL 2005*, Edinburgh/UK, 2005.
- Ralph Debusmann, Oana Postolache, and Maarika Traat.
  A modular account of information structure in Extensible Dependency Grammar.
  In *Proceedings of the CICLING 2005 Conference*, Mexico City/MX, 2005. Springer.

Selected Publications III

- Ralph Debusmann and Gert Smolka.
  Multi-dimensional dependency grammar as multigraph description.
  In *Proceedings of FLAIRS-19*, Melbourne Beach/US, 2006. AAAI.
- Alexander Koller, Ralph Debusmann, Malte Gabsdil, and Kristina Striegnitz.
  Put my galakmid coin into the dispenser and kick it: Computational linguistics and theorem proving in a computer game.
  Journal of Logic, Language and Information, 13(2):187–206, 2004.

Selected Publications IV

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

Selected Publications by Other Authors I

- Ondrej Bojar.
  Problems of inducing large coverage constraint-based dependency grammar.
  In *Proceedings of the International Workshop on Constraint Solving and Language Processing*, Roskilde/DK, 2004.
- Peter Dienes, Alexander Koller, and Marco Kuhlmann.
  Statistical A* dependency parsing.
  In *Prospects and Advances in the Syntax/Semantics Interface*, Nancy/FR, 2003.
- Alexander Koller and Kristina Striegnitz.
  Generation as dependency parsing.
  In *Proceedings of ACL 2002*, Philadelphia/US, 2002.

Selected Publications by Other Authors II

Christian Korthals.
Unsupervised learning of word order rules, 2003.
Diploma thesis.

Pierre Lison.
Implémentation d'une interface sémantique-syntaxe basée sur des grammaires d'unification polarisées.
Master's thesis, Univesité Catholique de Louvain, 2006.

Jorge Pelizzoni and Maria das Gracas Volpe Nunes.
N:M mapping in XDG - the case for upgrading groups.
In *Proceedings of the International Workshop on Constraint Solving and Language Processing*, Sitges/ES, 2005.

Future Work

- formalization:
  - strengthen relation to other grammar formalisms
  - formalize XDG in a weaker logic than HOL, e.g. MSO
- implementation:
  - make use of new constraint technology, e.g. Gecode (Schulte/Tack 2005)
  - automatically generate principle propagators from EMSO-specifications (Tack et al. 2006)

Thank you!

References I

Krzysztof R. Apt.
Principles of Constraint Programming.
Cambridge University Press, 2003.

Tilman Becker, Owen Rambow, and Michael Niv.
The derivational generative power, or, scrambling is beyond LCFRS.
Technical report, University of Pennsylvania, 1992.

Joan Bresnan.
Lexical Functional Syntax.
Blackwell, 2001.

Noam Chomsky.
Syntactic Structures.
Janua linguarum. Mouton, The Hague/NL, 1957.

Noam Chomsky.
Aspects of the Theory of Syntax.
MIT Press, Cambridge/US, 1965.

Noam Chomsky.
Lectures on Government and Binding: The Pisa Lectures.
Foris Publications, 1981.

References II

Noam Chomsky.
The Minimalist Program.
MIT Press, 1995.

Denys Duchier.
Axiomatizing dependency parsing using set constraints.
In *Proceedings of MOL 6*, Orlando/US, 1999.

Denys Duchier.
Configuration of labeled trees under lexicalized constraints and principles.
Research on Language and Computation, 1(3–4):307–336, 2003.

Markus Egg, Alexander Koller, and Joachim Niehren.
The Constraint Language for Lambda Structures.
Journal of Logic, Language, and Information, 2001.

Ray Jackendoff.
Foundations of Language.
Oxford University Press, 2002.

Aravind K. Joshi.
An introduction to tree-adjoining grammars.
In Alexis Manaster-Ramer, editor, *Mathematics of Language*, pages 87–115. John Benjamins, Amsterdam/NL, 1987.

References III

- Igor Mel'čuk.
  Dependency Syntax: Theory and Practice.
  State Univ. Press of New York, Albany/US, 1988.

- Wolfgang Menzel and Ingo Schröder.
  Decision procedures for dependency parsing using graded constraints.
  In *Proceedings of the COLING/ACL 1998 Workshop Processing of Dependency-based Grammars*, Montréal/CA, 1998.

- Carl Pollard and Ivan A. Sag.
  Head-Driven Phrase Structure Grammar.
  University of Chicago Press, Chicago/US, 1994.

- James Rogers.
  A model-theoretic framework for theories of syntax.
  In *Proceedings of ACL 1996*, 1996.

- Jerrold M. Sadock.
  Autolexical Syntax.
  University of Chicago Press, 1991.

- Christian Schulte.
  Programming Constraint Services, volume 2302 of *Lecture Notes in Artificial Intelligence*.
  Springer-Verlag, 2002.

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.
D. Reidel, Dordrecht/NL, 1986.

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.
MIT Press, Cambridge/US, 2000.

Guido Tack, Christian Schulte, and Gert Smolka.
Generating propagators for finite set constraints.
In Fréderic Benhamou, editor, *12th International Conference on Principles and Practice of Constraint Programming*, volume 4204 of *Lecture Notes in Computer Science*, pages 575–589. Springer, 2006.

Lucien Tesnière.
Eléments de Syntaxe Structurale.
Klincksiek, Paris/FR, 1959.