Dependenzgrammatik als Beschreibung von Multigraphen

Ralph Debusmann

Programming Systems Lab, Saarbrücken

CHORUS-Treffen, 15. Februar 2005

Überblick

Überblick

Linguistische Aspekte

- es gibt viele verschiedene linguistische Aspekte:
  - Syntax
    - unmittelbare Dominanz
    - lineare Präzedenz
  - Semantik
    - Prädikat-Argument-Struktur
    - Quantoren-Skopus
  - andere:
    - Informationsstruktur
    - prosodische Struktur
- Psychologie (Jackendoff 2002): allesamt eigenständige Module, werden gleichzeitig verarbeitet

Bisherige Grammatikformalismen

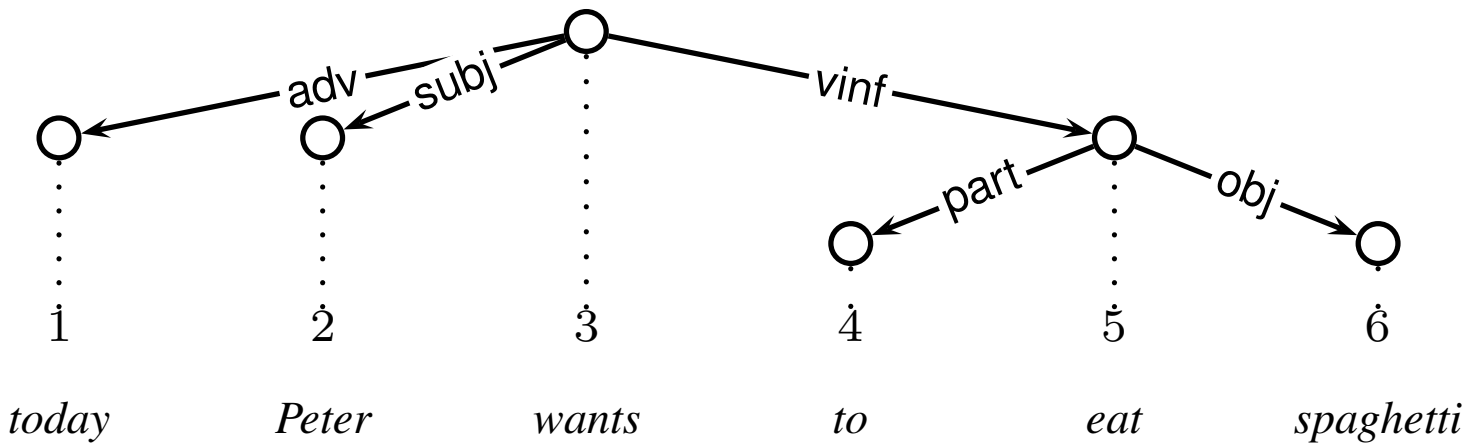
- Ausgangspunkt Syntax
- Ableitung der anderen Aspekte aus Syntax
- nur Syntax eigenständiges Modul
- keine gleichzeitige Verarbeitung

Extensible Dependency Grammar

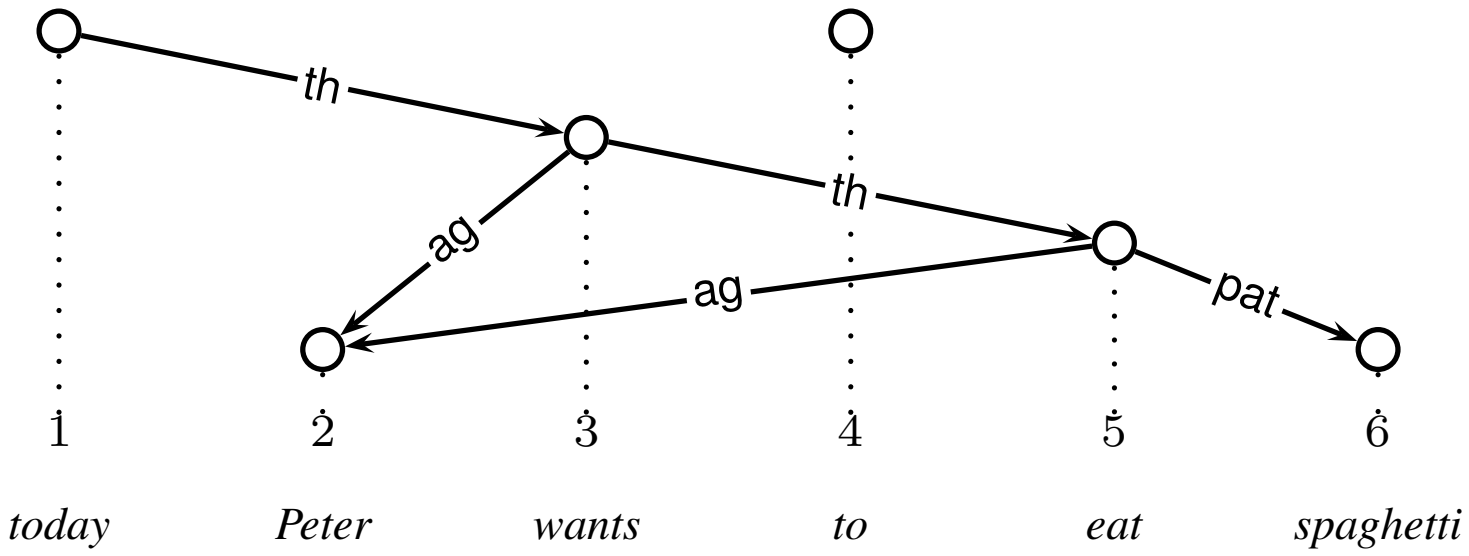
- neuer Ansatz: Extensible Dependency Grammar (XDG)
- jeder linguistische Aspekt repräsentiert durch eigenen Dependenzgraph
- eigenständige Module
- gleichzeitige Verarbeitung
- diese Präsentation: Formalisierung von XDG

Überblick

Dependenzgraph-Beispiel (Syntax)



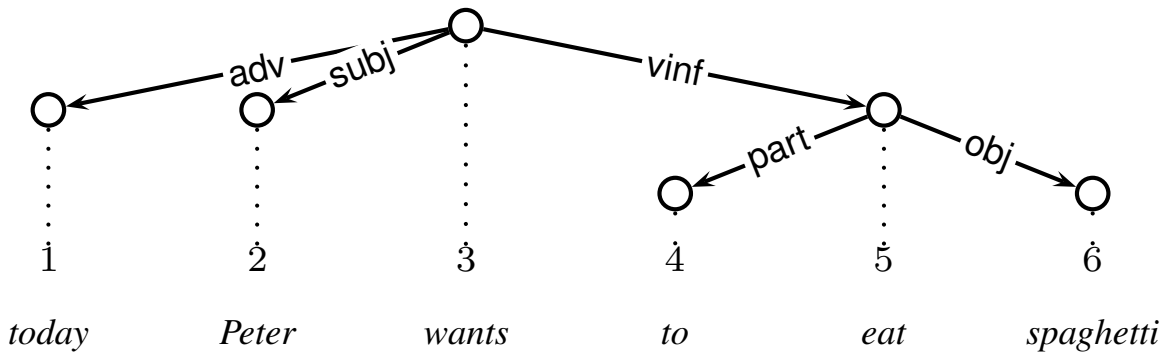
Dependenzgraph-Beispiel (Semantik)

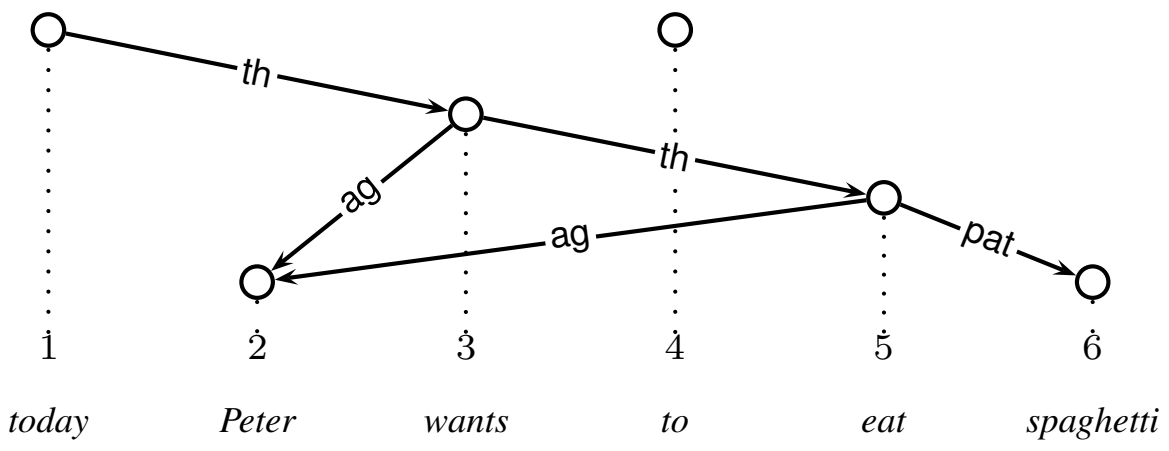


Multigraphen

- multi-dimensionale Dependenzgraphen aus beliebig vielen Dependenzgraphen (Dimensionen)
- dieselbe Knotenmenge

Zweidimensionaler Multigraph





Formalisierung

- ein Multigraph ist ein Tupel (V, D, W, w, L, E, A, a)

Komponenten

1. endliche Menge V von Knoten (endlicher Intervall der natürlichen Zahlen beginnend mit 1, deshalb total geordnet)
2. endliche Menge D von Dimensionen
3. endliche Menge W von Wörtern
4. Knoten-Wort-Abbildung $w \in V \to W$
5. endliche Menge L von Kantenmarkierungen
6. Menge $E \subseteq V \times V \times D \times L$ von markierten gerichteten Kanten
7. endliche Menge A von Attributen
8. Knoten-Attribute-Abbildung $a \in V \to D \to A$

Relationen

- jede Dimension $d \in D$ induziert zwei Relationen:

Zwei Relationen

1. markierte Kante: $\dot{\longrightarrow}_d$
2. Präzedenz: $v \prec v'$

Überblick

Typen

- $T \in Ty$ gegeben Menge At von Atomen (beliebige Symbole):

Typen

$T ::=$	B	Boolescher Wert
$\mid$	V	Knoten
$\mid$	$T_1 \to T_2$	Funktion
$\mid$	$\{a_1, \ldots, a_n\}$	endlicher Bereich
$\mid$	$\{a_1 : T_1, \ldots, a_n : T_n\}$	Record

Terme

- $t \in Te$ definiert gegeben eine Menge At von Atomen und Con von Konstanten:

Terme

$t \in Te ::=$	x	Variable
$\mid$	c	Konstante
$\mid$	a	Atom
$\mid$	$\lambda x : T.t$	Abstraktion
$\mid$	$t_1\ t_2$	Applikation
$\mid$	$\{a_1 = t_1, \ldots, a_n = t_n\}$	Record
$\mid$	$t.a$	Record-Auswahl

Multigraph-Typ

- Multigraphen können aufgrund ihrer Dimensionen, Wörter, Markierungen und Attribute unterschieden werden
- Multigraph-Typ: Tupel $MT = (Dim, Word, lab, attr)$

Multigraph-Typ

1. $Dim \in Ty$ endlicher Bereich der Dimensionen
2. $Word \in Ty$ endlicher Bereich der Wörter
3. $lab \in Dim \rightarrow Ty$ Funktion von Dimensionen zu Markierungstypen, d.h. zum endlichen Bereich der Kantenmarkierungen auf dieser Dimension
4. $attr \in Dim \rightarrow Ty$ Funktion von Dimensionen zu Attributtypen, d.h. zum (beliebigen) Typ der Attribute auf dieser Dimension

Multigraphen und Multigraph-Typen

- ein Multigraph $M = (V, D, W, w, L, E, A, a)$ hat Multigraph-Typ $MT = (Dim, Word, lab, attr)$ genau dann wenn:

Bedingungen

1. die Dimensionen dieselben sind
2. die Wörter dieselben sind
3. die Kanten in E die richtigen Kantenmarkierungen für ihre Dimension haben gemäß lab
4. die Knoten die richtigen Attribute für ihre Dimension haben gemäß $attr$

Signatur

- zwei Arten von Konstanten:
  1. logische Konstanten
  2. Multigraph-Konstanten

Logische Konstanten

- beinhalten Typkonstante B und die folgenden Termkonstanten:

Logische Konstanten

0	:	B	falsch
$\Rightarrow$	:	$\mathsf{B} \to \mathsf{B} \to \mathsf{B}$	Implikation
$\doteq_T$	:	$T \to T \to \mathsf{B}$	Gleichheit
$\exists_T$	:	$(T \to \mathsf{B}) \to \mathsf{B}$	Quantifizierung

- andere logische Konstanten wie üblich abgeleitet

Multigraph-Konstanten

- festgelegt durch Multigraph-Typ
- beinhalten die Typkonstante V und die folgenden Termkonstanten:

Multigraph-Konstanten

$\dot{\longrightarrow}_d$	:	$V \to V \to lab\ d \to B$	markierte Kante
$\prec$	:	$V \to V \to B$	Präzedenz
$(w\ \cdot)$	:	$V \to Word$	Wort
$(d\ \cdot)$	:	$V \to attr\ d$	Attribute

- andere Multigraph-Konstanten wie $\to_d$, $\to_d^+$, $\to_d^*$ abgeleitet

Grammatik

- eine XDG Grammatik $G = (MT, P)$ is definiert durch:
  1. einen Multigraph-Typ MT
  2. eine Menge P von Formeln, genannt Prinzipien
- jedes Prinzip muss gemäß der Signatur MT formuliert sein

Modelle

- die Modelle einer Grammatik $G = (MT, P)$ sind alle Multigraphen, die
  1. Multigraph-Typ MT haben
  2. alle Prinzipien P erfüllen

Stringsprache

- gegeben eine Grammatik $G = (MT, P)$ ist die Stringsprache $L(G)$ ist die Menge aller Strings $s = w_1 \ldots w_n$ so dass:
  1. es ein Modell von G mit gleichvielen Knoten wie Wörtern gibt:
     $$V \quad = \quad \{1, \ldots, n\}$$
  2. die Konkatenation der Wörter der Knoten dieses Modells s ergibt:
     $$(w\ 1) \ldots (w\ n) \quad = \quad s$$

Überblick

Dependenzgrammatik

- Sammlung von Ideen, meist zitiert: (Tesniere 1959), (Melcuk 1988)
  - 1:1-Abbildung Knoten:Wörter
  - Kopf-Dependent-Asymmetrie
  - Lexikalisierung
  - Valenz
  - Ordnung
  - Projektivität
  - Mehrdimensionalität

1:1-Abbildung Wörter:Knoten

- Knoten-Wort-Abbildung
- Multigraphen waren definiert als Tupel:
  (V, D, W, w, L, E, A, a)

Komponenten

1. ...
2. die Knoten-Wort Abbildung $w \in V \rightarrow W$
3. ...

Kopf-Dependent-Asymmetrie

- markierte gerichtete Kanten
- Multigraphen waren Tupel: (V, D, W, w, L, E, A, a)

Komponenten

1. ...
2. eine Menge $E \subseteq V \times V \times D \times L$ von markierten gerichteten Kanten
3. ...

Lexikalisierung

- Idee: Verhalten der Knoten hängt von assoziierten Wörtern ab
- um Lexikalisierung in XDG zu modellieren, teilen wir die Attribute auf in:
  1. lexikalische Attribute
  2. nicht-lexikalische Attribute

Attribute

- die Attribute $attr\ d$ jeder lexikalisierten Dimension d müssen ein Record vom folgenden Typ sein:

$$attr\ d \quad = \quad \left\{ \begin{array}{ccc} lex & : & L \\ a_1 & : & \ldots \\ & \ldots & \\ a_n & : & \ldots \end{array} \right\}$$

- wobei lex die lexikalischen Attribute vom Typ L beherbergt, und $a_1, \ldots a_n$ die nicht-lexikalischen Attribute sind

Lexikon

- das Lexikon ist eine endliche Menge von Lexikoneinträgen vom Typ E:

$$E = \left\{ \begin{array}{ccc} word & : & Word \\ d_1 & : & L_1 \\ & \ldots & \\ d_m & : & L_m \end{array} \right\}$$

- wobei jeder Lexikoneintrag mit einem Wort assoziiert ist und die lexikalischen Attribute der Dimensionen $d_1, \ldots, d_m$ spezifiziert

Lexikalisierungs-Prinzip

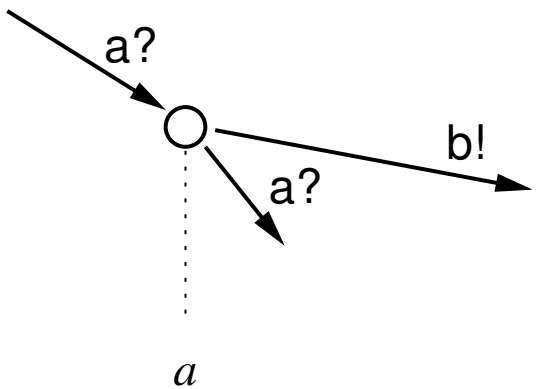
1. für jeden Knoten muss ein Lexikoneintrag e aus dem Lexikon $lexicon$ ausgewählt werden
2. e muss mit dem gleichen Wort assoziiert sein, mit dem der Knoten assoziiert ist
3. e bestimmt die lexikalischen Attribute

Lexikalisierungsprinzip

$$
\begin{aligned}
&lexicalization_{lexicon,d_1,\ldots,d_m} = \\
&\exists e \in lexicon \wedge \\
&e.word \doteq (w\ v) \wedge \\
&(d_1\ v).lex \doteq e.d_1 \wedge \ldots \wedge (d_n\ v).lex \doteq e.d_m
\end{aligned}
$$

Valenz

- Idee: lexikalische Spezifikation der lizensierten eingehenden und ausgehenden Kanten
- führt zu Begriff der Konfiguration von Fragmenten, die zu Analysen zusammengebaut werden müssen
- Beispielfragment:

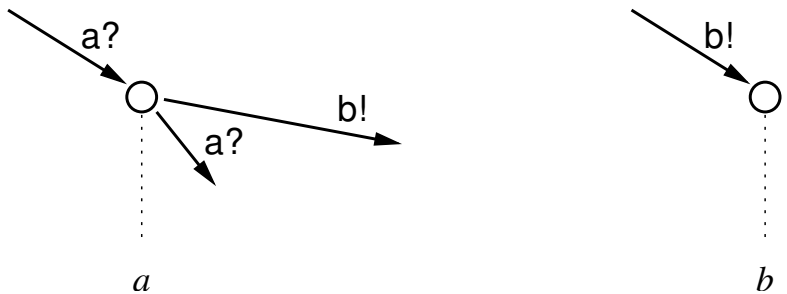


Beispielgrammatik

- Beispielgrammatik:

$$\text{EQAB} \quad = \quad \{w \in (a \cup b)^+ \mid |w|_a = |w|_b\}$$

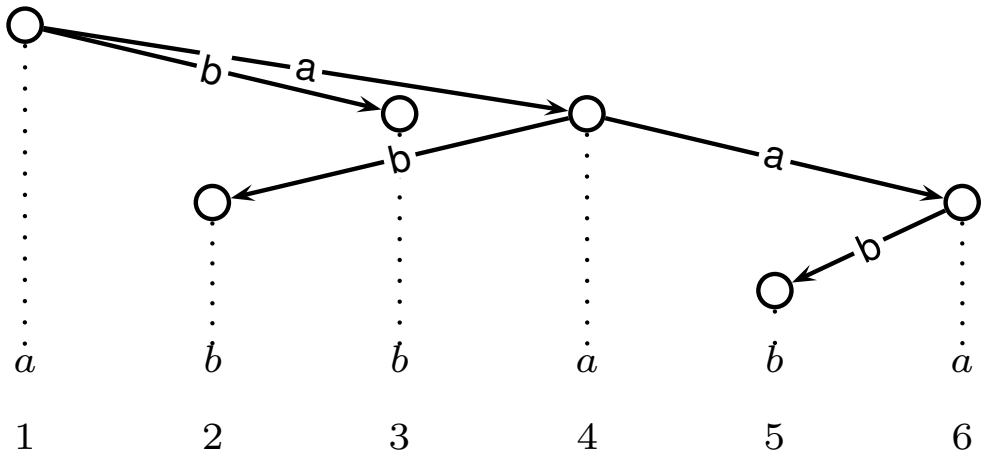
- Idee: benutze die folgenden Fragmente:



- zusätzliche Einschränkung: Modelle müssen Bäume sein

Beispielanalyse

- Beispielanalyse:



- intuitiv: as in Kette angeordnet, jedes a muss eine ausgehende Kante zu einem b haben
- Ergebnis: gleichviele as und bs

Valenz in XDG

- keine eingehenden Kanten mit Markierung l für Knoten v auf Dimension d:

$$in0_d\ v\ l \quad = \quad \neg\exists v' : v' \xrightarrow{l}_d v$$

- genau eine eingehende Kante mit Markierung l:

$$in1_d\ v\ l \quad = \quad \exists^1 v' : v' \xrightarrow{l}_d v$$

- höchstens eine eingehende Kante mit Markierung l:

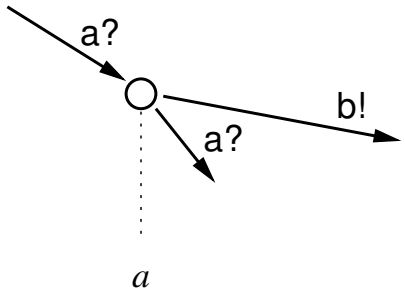
$$in0or1_d\ v\ l \quad = \quad (in0_d\ v\ l) \vee (in1_d\ v\ l)$$

Lexikalisierung von Valenz

- Idee: drücke Fragmente durch Lexikoneinträge aus
- zwei lexikalische Attribute: in und out
- bilden Kantenmarkierungen auf Kardinalitäten ab:
  - ! (genau eine Kante)
  - ? (höchstens eine Kante)
  - $*$ (beliebig viele Kanten)
  - 0 (keine Kanten)

Lexikalisierung von Valenz: Beispiel

- Beispielfragment:



- dazugehöriger Lexikoneintrag:

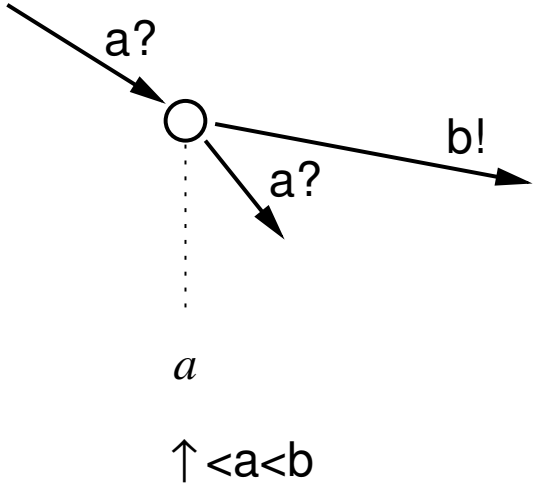
$$\left\{ \begin{array}{rl} word = & a \\ \mathsf{ID} = & \left\{ \begin{array}{rl} in = & \{a =?, b = 0\} \\ out = & \{a =?, b =!\} \end{array} \right\} \end{array} \right\}$$

Valenz-Prinzip

$$
\begin{aligned}
&valency_d = \\
&\forall l : \\
&\quad (d\ v).lex.in.l \doteq 0 \Rightarrow in0\ v\ l \\
&\quad (d\ v).lex.in.l \doteq ! \Rightarrow in1\ v\ l \\
&\quad (d\ v).lex.in.l \doteq ? \Rightarrow in0or1\ v\ l \\
&\\
&\quad (d\ v).lex.in.l \doteq 0 \Rightarrow out0\ v\ l \\
&\quad (d\ v).lex.in.l \doteq ! \Rightarrow out1\ v\ l \\
&\quad (d\ v).lex.in.l \doteq ? \Rightarrow out0or1\ v\ l
\end{aligned}
$$

Ordnung

- Idee: Modelle bisher ungeordnet, nun möchten wir sie ordnen
- lexikalische Ordnung auf den Dependenten jedes Knotens, ordnet außerdem jeden Kopf relativ zu seinen Dependenten
- zusätzliche Kantenmarkierung $\uparrow$ für den Kopf:

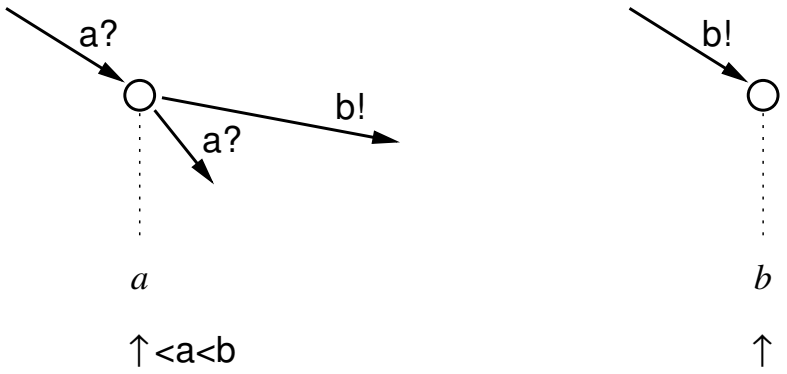


Beispielgrammatik

- Beispielgrammatik:

$$\text{ANBN} \quad = \quad \{w \in a^n b^n \mid n \geq 1\}$$

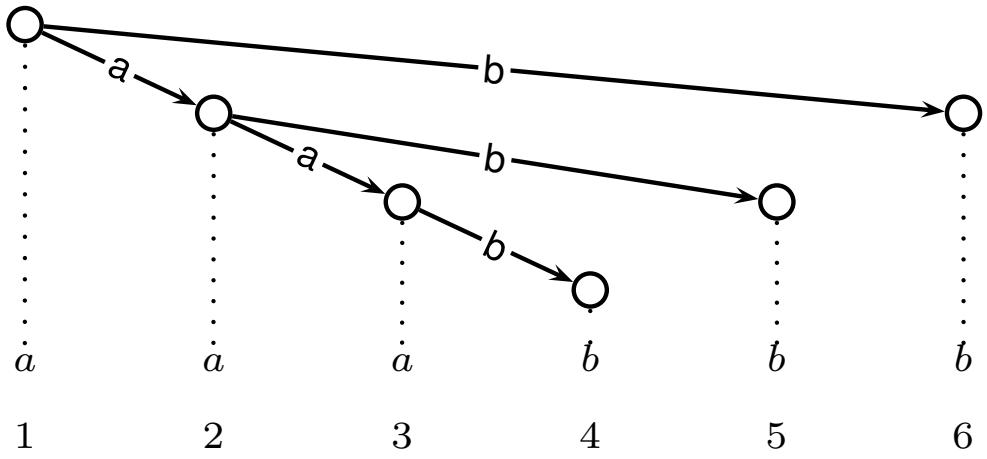
- Idee: benutze die folgenden geordneten Fragmente:



- Modelle müssen Bäume sein und projektiv (s.u.)

Beispielanalyse

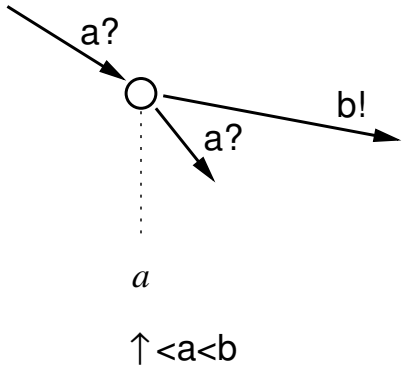
- Beispielanalyse:



- wie vorher: as in Kette angeordnet, und jedes a muss eine ausgehende Kanten zu einem b haben (gleichviele as und bs)
- dazu: alle as müssen vor allen bs stehen, und alle Köpfe vor ihren Dependenten

Lexikalisierung von Ordnung

- lexikalisches Attribut $order$: Menge von Paaren von Kantenmarkierungen, repräsentiert strikte partielle Ordnung
- z.B. geordnetes Fragment:



- dazugehöriger Lexikoneintrag:

$$\left\{ \begin{array}{l} word = a \\ \quad d = \left\{ \begin{array}{r} in = \{a = ?, b = 0\} \\ out = \{a = ?, b = !\} \\ order = \{(\uparrow, a), (\uparrow, b), (a, b)\} \end{array} \right\} \end{array} \right\}$$

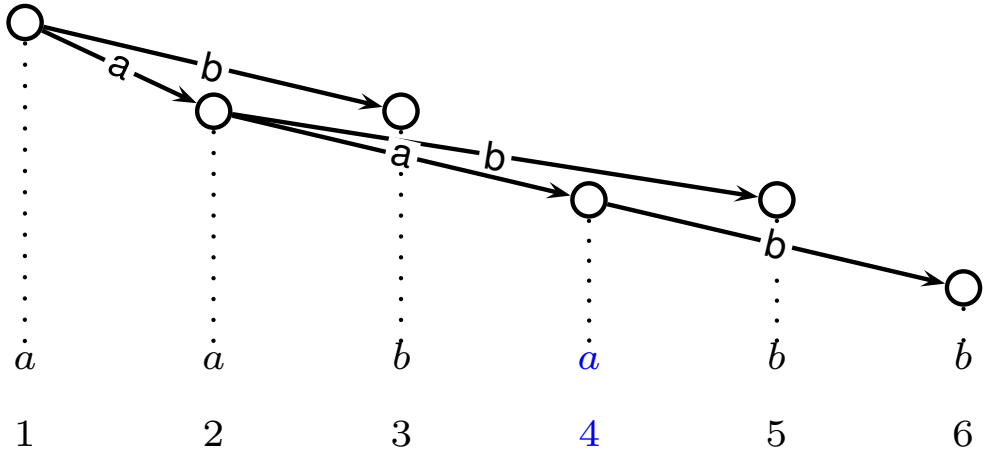
Ordnungs-Prinzip

- realisiert lexikalisierte Ordnung:

$$
\begin{aligned}
&order_d = \\
&\forall (l, l') \in (d\ v).lex.order : \\
&v \xrightarrow{l}_d v' \quad \wedge \quad v \xrightarrow{l'}_d v'' \quad \Rightarrow \quad v' \prec v'' \quad \wedge \\
&\qquad l \doteq \uparrow \quad \wedge \quad v \xrightarrow{l'}_d v' \quad \Rightarrow \quad v \prec v' \quad \wedge \\
&v \xrightarrow{l}_d v' \quad \wedge \quad l' \doteq \uparrow \quad \Rightarrow \quad v' \prec v
\end{aligned}
$$

Projektivität

- Problem: lokale Anordnung der Dependenten reicht nicht aus, um ANBN zu modellieren
- Gegenbeispiel: alle a-Dependenten stehen vor allen b-Dependenten, und alle Köpfe stehen vor ihren Dependenten, trotzdem stehen global nicht alle as vor allen bs:



Projektivität: Idee

- Idee: wir müssen nicht nur die Dependenten allein, sondern die Dependenten einschließlich ihrer Unterbäume ordnen
- das erreichen wir, indem wir verbieten, dass Kanten die Projektionskanten von Knoten, die weiter oben im Graph sind, schneiden

Projektivitäts-Prinzip

- realisiert Projektivität:

$$
\begin{aligned}
&projectivity_d = \lambda v. \\
&v \rightarrow_d v' \wedge v \prec v' \quad \Rightarrow \quad \forall v'' : v \prec v'' \wedge v'' \prec v' \quad \Rightarrow \quad v \rightarrow_d^+ v'' \quad \wedge \\
&v \rightarrow_d v' \wedge v' \prec v \quad \Rightarrow \quad \forall v'' : v' \prec v'' \wedge v'' \prec v \quad \Rightarrow \quad v \rightarrow_d^+ v''
\end{aligned}
$$

Mehrdimensionalität

- zusätzliche Ausdrucksstärke und Modularität, z.B., um andere Aspekte linguistischer Beschreibung zu modellieren (Semantik, Informationsstruktur)
- auch ermöglicht: elegante Behandlung von Wortstellungsproblemen
- Beispiel: Benutzung von zwei Dimensionen zur Modellierung der nicht-kontextfreien Sprache ANBNCN:

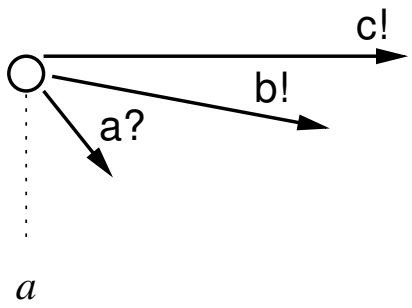
$$\text{ANBNCN} \quad = \quad \{w \in a^n b^n c^n \mid n \geq 1\}$$

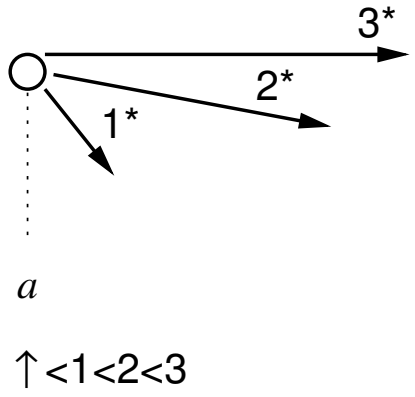
Grammatik für ANBNCN: Idee

- Idee: Zählen und Ordnen trennen:
  1. Immediate Dominance (ID)-Dimension für das Zählen, d.h. um sicherzustellen, dass für jedes a genau ein b und ein c vorhanden ist
  2. Linear Precedence (LP)-Dimension für das Ordnen, d.h. um sicherzustellen, dass alle as vor allen bs und die wiederum vor allen cs stehen
- ID dimension: ungeordneter Baum
- LP dimension: geordneter Baum

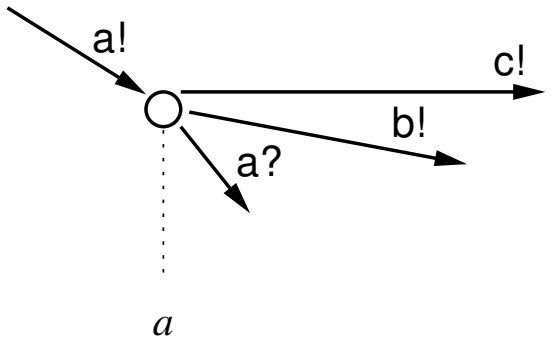
Grammatik für ANBNCN: a

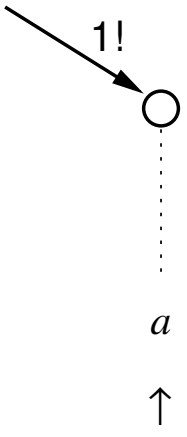
- a als Wurzel (links: ID, rechts: LP):





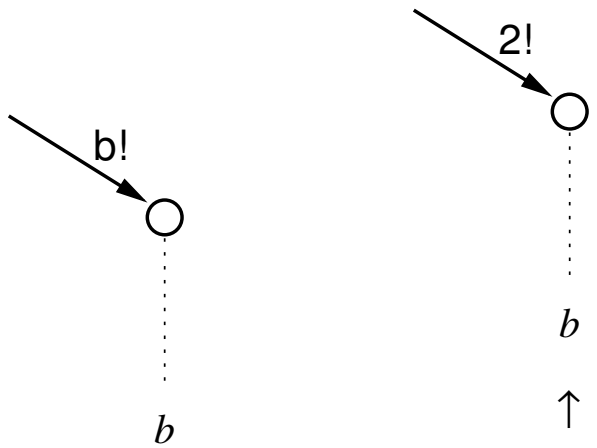
- a als Dependent:



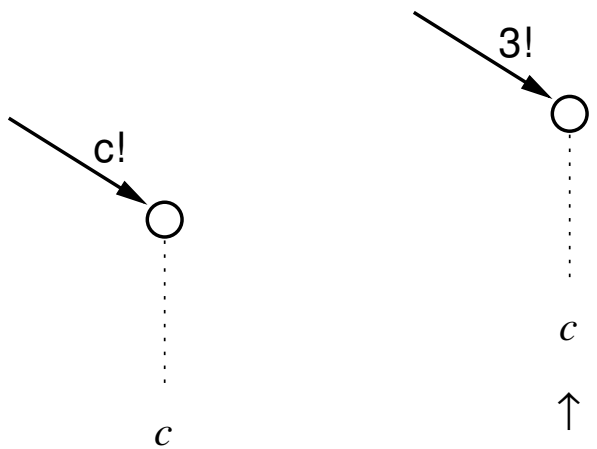


Grammatik für ANBNCN: b und c

- b:

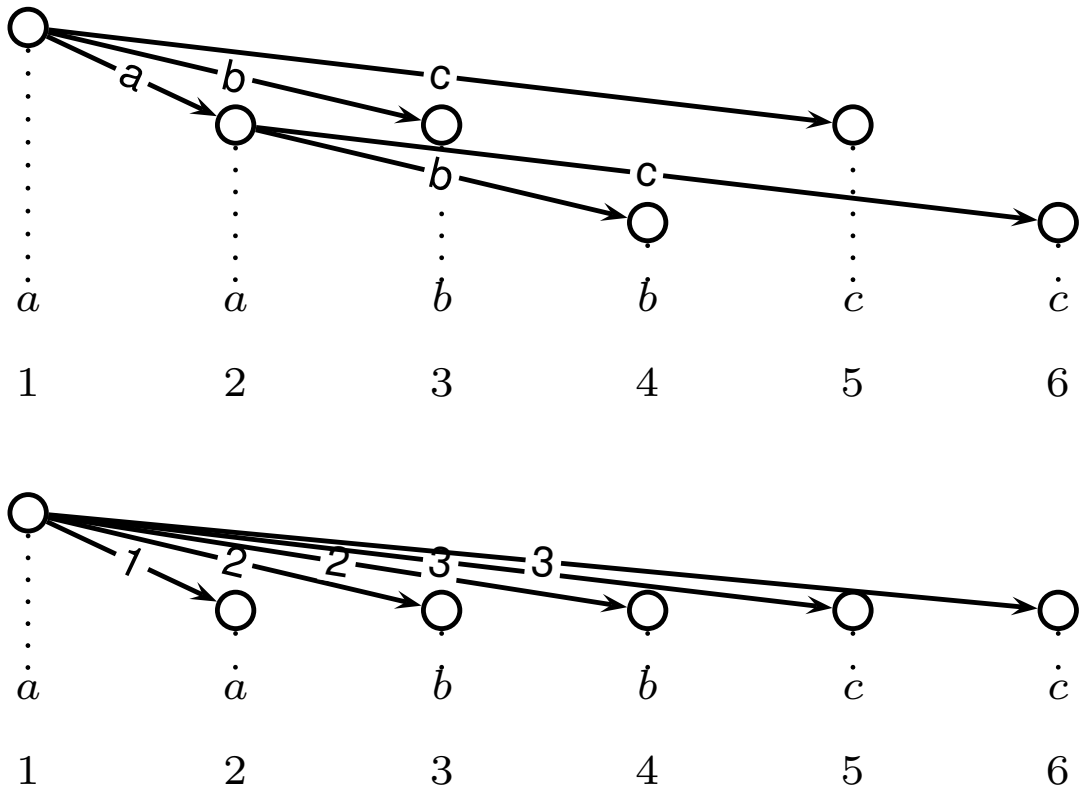


- c:



Beispielanalyse

- oben: ID, unten: LP:



Überblick

Stringsprachen

- kontextfreie Sprachen: kontextfreie Grammatik kann leicht kodiert werden (Beweis in Diss.)
- darüber hinaus: z.B. ANBNCN
- außerdem (Diss.): Cross-Serial Dependencies und Scrambling
- Scrambling: kann von keiner Instanz von LCFRS (Generalisierung von TAG) abgedeckt werden (Becker/Rambow/Niv 1992)

Linguistische Modellierung

- elegante Grammatik für Deutsche Wortstellungsspezialitäten, z.B. auch Scrambling (Duchier/Debusmann ACL 2001), (Bader/Foeldesi/Pfeiffer/Steigner Softwareprojekt 2004)
- verschiedene kleinere Grammatiken, auch von anderen Forschern (Arabisch, Französisch, Schwedisch)
- relationale Syntax-Semantik-Schnittstelle (Debusmann/Duchier/Koller/Kuhlmann/Smolka/Thater COLING 2004)
- modulare Version der prosodischen Theorie der Informationsstruktur von Mark Steedman (Debusmann/Postolache/Traat CICLING 2005)
- erstmalige Realisierung der parallelen Grammatikarchitektur von (Jackendoff 2002) (Diss.)

Erkennungsproblem

- zwei Arten (Trautwein 1995):

Erkennungsproblem

1. universelles Erkennungsproblem: gegeben ein Paar (G, s), wobei G eine Grammatik ist und s ein String, ist s in $L(G)$?
2. festgelegtes Erkennungsproblem: sei G eine festgelegte Grammatik. Gegeben einen String s, ist s in $L(G)$?

Komplexität des Erkennungsproblems

- festgelegtes Erkennungsproblem: NP-hart
- Beweis: Reduktion des SAT-Problems (Diss., (Debusmann/Smolka 2006 FLAIRS))
- universelles Erkennungsproblem: folglich auch NP-hart
- keine obere Schranke bewiesen unter dieser Formalisierung: wahrscheinlich beide nicht in NP
- mit geeigneten Einschränkungen (z.B. auf first-order-Formeln): Vermutung: in NP

Parsen

- Constraint-basierter Parser in XDG Development Kit (XDK) (Debusmann/Duchier/Niehren 2004 MOZ)
- effizient für handgeschriebene Grammatiken trotz der hohen Komplexität
- bi-direktional: Parsen und Generieren mit derselben Grammatik, gleichzeitige Verarbeitung aller Dimensionen
- aber: bisherige Experimente mit Parsen von großen Grammatiken sind allesamt fehlgeschlagen

Überblick

Zusammenfassung

- XDG: sehr expressiver und modularer Grammatikformalismus
- eigenständige Dimensionen
- gleichzeitige Verabeitung
- erste vollständige Formalisierung im Lambda-Kalkül
- Realisierung der Ideen der Dependenzgrammatik in XDG

Zukünftige Forschung

- ermöglicht XDG effizientes Parsen auch von großen Grammatiken?
- falls nicht, warum?
- z.B. LFG und HPSG sind auch NP-hart (Barton/Berwick/Ristad 1987), (Trautwein 1995), trotzdem gibt es effiziente Parser
- zwei Richtungen

Theoretisch

- gibt es polynomielle Instanzen von XDG?
- Marcos Forschung legt sehr nahe, dass es welche gibt (zumindest mit einer Graph-Dimension)
- richtig spannend aber erst aus XDG-Sicht: Instanzen mit mehreren Dimensionen (z.B. TDG)
- formaler Vergleich mit anderen mehrdimensionalen Formalismen:
  - LFG (Bresnan/Kaplan 1982)
  - HPSG (Pollard/Sag 1994)
  - MMCCG (Kruijff/Baldridge 2004)
  - Synchronous TAG (Shieber/Schabes 1990)
  - Generalized Multitext Grammars (Melamed/Satta/Wellington 2004)
  - Hetrogeneous Relation Graphs (Taylor/Black/Caley 2001)

Praktisch

- neue Induktionsmethoden
- Profiling des bisherigen XDG Parsers
- Neuimplementierung des XDG-Parsers mit Gecode-Bibliothek statt Mozart
- Einbeziehung von Supertagging (lexikalische Präferenzen)
- Kodierung von XDG in besser verarbeitbare Formalismen

Danke für eure Aufmerksamkeit!

Literatur


Regine Bader, Christine Foeldesi, Ulrich Pfeiffer, and Jochen Steigner.
Modellierung grammatischer Phänomene der deutschen Sprache mit Topologischer Dependenzgrammatik.
Technical report, Saarland University, 2004.
Softwareprojekt.


G. Edward Barton, Robert Berwick, and Eric Sven Ristad.
Computational Complexity and Natural Language.
MIT Press, 1987.

Literatur


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


Joan Bresnan and Ronald Kaplan.
Lexical-Functional Grammar: A formal system for grammatical representation.
In Joan Bresnan, editor, *The Mental Representation of Grammatical Relations*, pages 173–281. The MIT Press, Cambridge/US, 1982.

Literatur


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, 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.

Literatur


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.


Ralph Debusmann and Gert Smolka.
Multi-dimensional dependency grammar as multigraph description.
In *Proceedings of FLAIRS-19*, Melbourne Beach/US, 2006. AAAI.

Literatur


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


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

Literatur


Geert-Jan M. Kruijff and Jason Baldridge.
Generalizing dimensionality in Combinatory Categorial Grammar.
In *Proceedings of COLING 2004*, Geneva/CH, 2004.


I. Dan Melamed, Giorgio Satta, and Benjamin Wellington.
Generalized Multitext Grammars.
In *Proceedings of ACL 2004*, Barcelona/ES, 2004.

Literatur


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


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

Literatur


Stuart M. Shieber and Yves Schabes.
Synchronous Tree Adjoining Grammars.
In *Proceedings of COLING 1990*, Helsinki/FI, 1990.


P. Taylor, A. Black, and R. Caley.
Hetrogeneous relation graphs as a mechanism for representing linguistic information.
Speech Communications, 33:153–174, 2001.

Literatur


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


Marten Trautwein.
The complexity of structure sharing in unification-based grammars.
In Walter Daelemans, Gert Durieux, and Steven Gillis, editors, *Computational Linguistics in the Netherlands 1995*, pages 165–179, 1995.