
Semantische Sprachverarbeitung

Carola Eschenbach

Universität Hamburg, MIN-Fakultät, Dept. Informatik
AB Wissens- und Sprachverarbeitung (WSV)

Sommersemester 2008

Semantische Sprachverarbeitung

Sitzung 6: Fortsetzung Typen und Lambda-Kalkül

- Wiedereinstieg Typisierter Lambda-Kalkül
 - Lambda in Prolog
 - Skopusambiguitäten
-

Wiedereinstieg typisierte Lambda-Kalkül

Syntax-Semantik-Korrespondenz

- Syntaktische Kategorien (Nominalphrase, Verbalphrase, ...)
- Semantische Typen (Quantor, Prädikat, ...)
- determinieren den Aufbau komplexer Ausdrücke
- Ausdrucksbedeutung: eigentlicher Beitrag + Typ-Beitrag

Semantische Typen

- Basistypen (e , t), komplexe Typen ($\langle \alpha, \beta \rangle$) für Funktionen

(Typisierte) Lambda-Ausdrücke

- Symbolen wird ein Typ zugeordnet.
- Wenn a vom Typ α und f vom Typ $\langle \alpha, \beta \rangle$ ist, dann ist $f(a)$ vom Typ β .
- Wenn x eine Variable vom Typ α und b vom Typ β ist, dann ist $\lambda x [b]$ vom Typ $\langle \alpha, \beta \rangle$.

Äquivalenzen

Definition

- Zwei λ -Terme A , B sind genau dann (*logisch*) äquivalent (\equiv), wenn für jede Interpretation I gilt: $I(A) = I(B)$

Äquivalenzen für λ -Terme

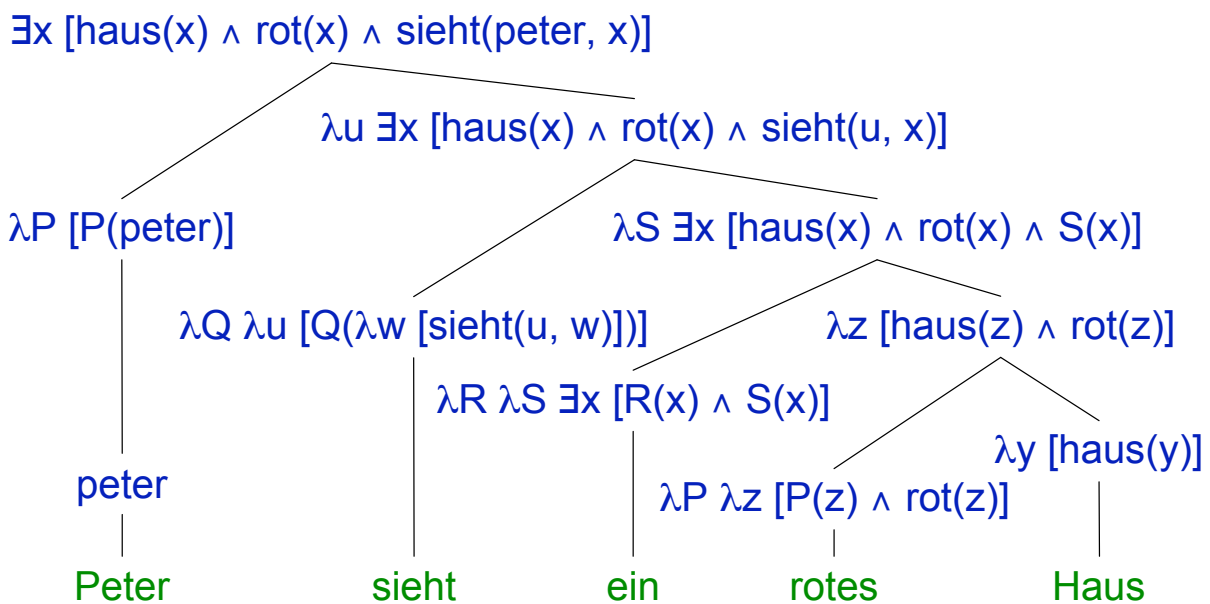
- Seien x und y Variablen und B ein Ausdruck vom selben Typ.
- Sei A ein Ausdruck, in dem y nicht vorkommt und der mit B keine gemeinsamen Variablen hat
- Gebundene Umbenennung
 - $\lambda x [A]$ ist äquivalent mit $\lambda y [A\{x / y\}]$
- Applikation nach Abstraktion
 - $\lambda x [A](B)$ ist äquivalent mit $A\{x / B\}$
- Abstraktion nach Applikation
 - $\lambda y [A(y)]$ ist äquivalent mit A

Peter sieht ein rotes Haus.

Peter	e	peter
sieht	$\langle\langle\langle e, t \rangle, t \rangle, \langle e, t \rangle\rangle$	$\lambda Q \lambda u [Q(\lambda w [\text{sieht}(u, w)])]$
ein	$\langle\langle e, t \rangle, \langle\langle e, t \rangle, t \rangle\rangle$	$\lambda R \lambda S \exists x [R(x) \wedge S(x)]$
rotes	$\langle\langle e, t \rangle, \langle e, t \rangle\rangle$	$\lambda P \lambda z [P(z) \wedge \text{rot}(z)]$
Haus	$\langle e, t \rangle$	$\lambda y [\text{haus}(y)]$

rotes Haus $\lambda P \lambda z [P(z) \wedge \text{rot}(z)](\lambda y [\text{haus}(y)])$
 $\equiv \lambda z [P(z) \wedge \lambda y [\text{haus}(y)](z)]$
 $\equiv \lambda z [P(z) \wedge \text{haus}(z)]$

Wahrheitsbedingungen: Peter sieht ein rotes Haus.



λ -Terme in PROLOG

- λ -Terme werden durch PROLOG-Terme repräsentiert
- Variablen: PROLOG-Variablen

Kein Symbol-Zeichensatz in PROLOG

$\text{lambda}(X, F)$ anstelle von $\lambda x [F]$

Keine Zusammengesetzten Funktionsausdrücke in PROLOG

$F@T$ anstelle von $F(T)$

Leider keine korrekte PROLOG -Syntax

Applikation wird repräsentiert, nicht 'ausgeführt'

Vereinfachung als Nachverarbeitung:

'Ausführung' der Applikation

- Beta-Konversion: Nutzung von
 - $\lambda x [A](B)$ ist äquivalent mit $A\{x / B\}$

Grammatik von Fragment 1 mit Lambda-Ausdrücken

$\text{np}(\text{np}/[\text{Det}, \text{N}], \text{Dsem@Nsem}) \text{ ---> } \text{det}(\text{Det}, \text{Dsem}), \text{n}(\text{N}, \text{Nsem}).$

$\text{n}(\text{n}/[\text{Adj}, \text{N}], \text{Asem@Nsem}) \text{ ---> } \text{adj}(\text{Adj}, \text{Asem}), \text{n}(\text{N}, \text{Nsem}).$

$\text{vp}(\text{vp}/[\text{TV}, \text{NP}], \text{Vsem@NPsem}) \text{ ---> } \text{tv}(\text{TV}, \text{Vsem}), \text{np}(\text{NP}, \text{NPsem}).$

$\text{s}(\text{s}/[\text{NP}, \text{VP}], \text{NPsem@VPsem}) \text{ ---> } \text{np}(\text{NP}, \text{NPsem}), \text{vp}(\text{VP}, \text{VPsem}).$

$\text{s}(\text{s}/[\text{S1}, \text{C}, \text{S2}], (\text{Csem@S1sem})@\text{S2sem}) \text{ ---> } \text{s}(\text{S1}, \text{S1sem}), \text{conj}(\text{C}, \text{Csem}), \text{s}(\text{S2}, \text{S2sem}).$

$\text{vp}(\text{vp}/[\text{IV}], \text{Vsem}) \text{ ---> } \text{iv}(\text{IV}, \text{Vsem}).$

$\text{np}(\text{np}/[\text{PN}], \text{lambda}(\text{P}, \text{P@Nsem})) \text{ ---> } \text{pn}(\text{PN}, \text{Nsem}).$

$\text{vp}(\text{vp}/[\text{VP1}, \text{C}, \text{VP2}], \text{lambda}(\text{X}, (\text{Cs}@\text{VP1s@X})@\text{VP2s@X})) \text{ ---> } \text{vp}(\text{VP1}, \text{VP1s}), \text{conj}(\text{C}, \text{Cs}), \text{vp}(\text{VP2}, \text{VP2s}).$

Zusammenfassung

typisierter Lambda-Kalkül

- Basis: Korrespondenz von syntaktischer Kategorie und semantischem Typ
- Systematische Spezifikation
 - des Kombinationspotentials der Lexeme
 - des semantischen Beitrags der Grammatik-Regeln
- Anwendung von Funktionen auf Argumente als basales Kombinationsmittel
- ergänzt um Typen-Anhebung

Ausprobieren ?

Paket Lambda

main.pl

- Grammatik mit Erzeugung der prädikatenlogischen Formeln unter Verwendung des Lambda-Kalküls
-

Skopusambiguitäten

- Jeder Christ verehrt eine Frau.
- Eine Frau wird von jedem Christen verehrt.

(1) ... und zwar seine Mutter

- 'jeder Christ' hat weiten Skopus
- 'eine Frau' hat engen Skopus

$\forall x [\text{christ}(x) \Rightarrow \exists y [\text{frau}(y) \wedge \text{verehrt}(x, y)]]$

(2) ... und zwar die Jungfrau Maria

- 'jeder Christ' hat engen Skopus
- 'eine Frau' hat weiten Skopus

$\exists y [\text{frau}(y) \wedge \forall x [\text{christ}(x) \Rightarrow \text{verehrt}(x, y)]]$

Beide Lesarten sind möglich

- Präferenzen von viele Faktoren abhängig (Betonung, Wortstellung, Kontext, ...)

Skopusambiguität: Beispiele

Welche Lesarten gibt es?

Welche Formeln drücken die Lesarten aus?

Welche Lesarten sind präferiert?

- Ein Namensschild lag neben jedem Teller.
- Ein Schüler war in jedem Klassenzimmer.
- Ein Fremdenführer führt jeden Besucher in zwei Museen.
- Eine Fahne hängt an der Vorderseite jeden Hauses.
- Jeder Christ verehrt eine Frau.
- Es ist nicht der Fall, dass jeder Christ eine Frau verehrt.

Weitere Beispiele?

Sind Skopusambiguitäten real?

Aus Lesart (2) folgt Lesart (1)

- (wenn Maria von allen Christen verehrt wird, dann verehrt jeder Christ (wenigstens) eine Frau)
- Lesart (1) ist allgemeiner als Lesart (2)

Hypothese

- Es gibt keine Mehrdeutigkeit
- (1) ist die Satzbedeutung
- (2) ein Spezialfall

Aufgabe

- Bestimmung der allgemeinsten Lesart

Drei Ansätze zur Skopusbestimmung

Der Skopus ergibt sich durch lineare Abfolge

- EIN Fremdenführer führt jeden Besucher in zwei Museen.
- aber: Ein Namensschild lag neben jedem Teller.

Skopus ergibt sich durch hierarchische Struktur

- EIN Fremdenführer führt jeden Besucher in zwei Museen.
- aber: Eine Fahne hängt an der Vorderseite jeden Hauses.

Skopus ist lexikalisch determiniert

- Jeder Christ verehrt eine Frau.
- $\forall x [\text{christ}(x) \Rightarrow \exists y [\text{frau}(y) \wedge \text{verehrt}(x, y)]]$
- Es ist nicht der Fall, dass jeder Christ eine Frau verehrt.
- $\neg \forall x [\text{christ}(x) \Rightarrow \exists y [\text{frau}(y) \wedge \text{verehrt}(x, y)]]$
- $\equiv \exists x [\text{christ}(x) \wedge \forall y [\text{frau}(y) \Rightarrow \neg \text{verehrt}(x, y)]]$
(dies ist die speziellere Interpretation)

Skopusambiguitäten sind real !

- Die Hypothese, dass die Bestimmung quantorenhaltiger Sätze genau der allgemeinsten Lesart entspricht, widerspricht der Kompositionalitätsannahme.

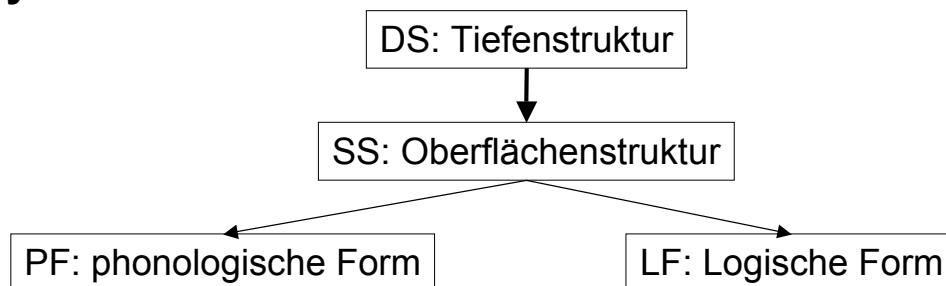
Annahmen

- Ein Satz, der quantifizierende NPs enthält, ist grundsätzlich hinsichtlich der Skopusbeziehungen mehrdeutig.
- Pragmatische Faktoren und lexikalische Eigenschaften können Lesarten für einzelne Sätze ausschließen.
- Zu jeder Skopusanordnung kann ein Satz gefunden werden, der die entsprechende Lesart zulässt.
- Es muss aber nicht unbedingt einen Satz geben, der alle Lesarten zulässt.

Syntaktische Strukturebenen: Transformationen

- Es gibt verschiedene Syntaktische Strukturebenen

Syntaxmodell



- jede syntaktische Ebenen hat eigenständige Wohlgeformtheitsbedingungen
- Generierungsabfolge entsprechend Transformationen / Konstituentenbewegungen

Syntaktische Transformationen: Beispiele DS → SS

Englisch: Topikalisierung

- John likes Mary → Mary₁, John likes t₁

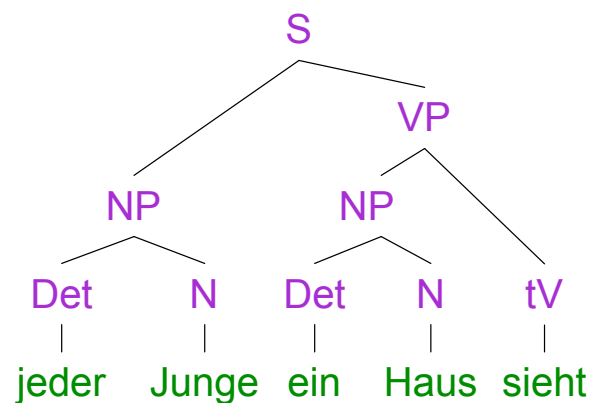
Deutsch: Fragesatz / Relativsatz

- Peter Laura sieht? → sieht₁₁ Peter Laura t₁₁?
- Laura den sieht → den₁₀ Laura t₁₀ sieht
- Laura wen sieht? → sieht₅ Laura wen t₅? → Wen₄ sieht₅ Laura t₄ t₅?

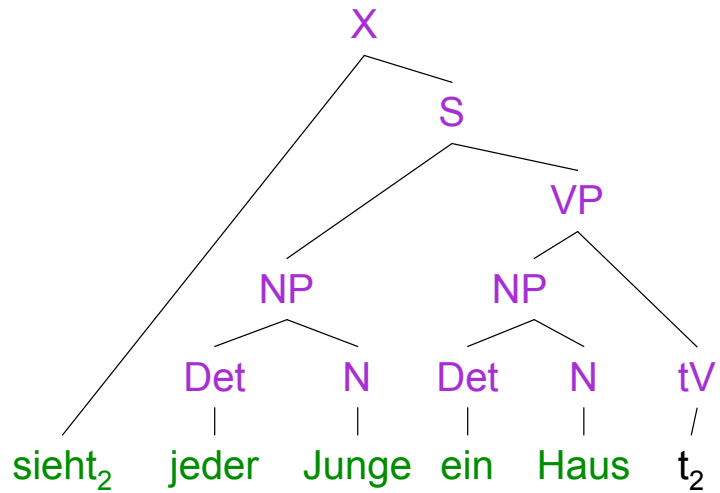
Deutsch: Hauptsatzstellung

- Peter Laura sieht → sieht₃ Peter Laura t₃ → Peter₂ sieht₃ t₂ Laura t₃
- Peter um-blättert → blättert₇ Peter um t₇ → Peter₆ blättert₇ t₆ um t₇
- Peter Laura gesehen hat → hat₉ Peter Laura gesehen t₉ → Peter₈ hat₉ t₈ Laura gesehen t₉

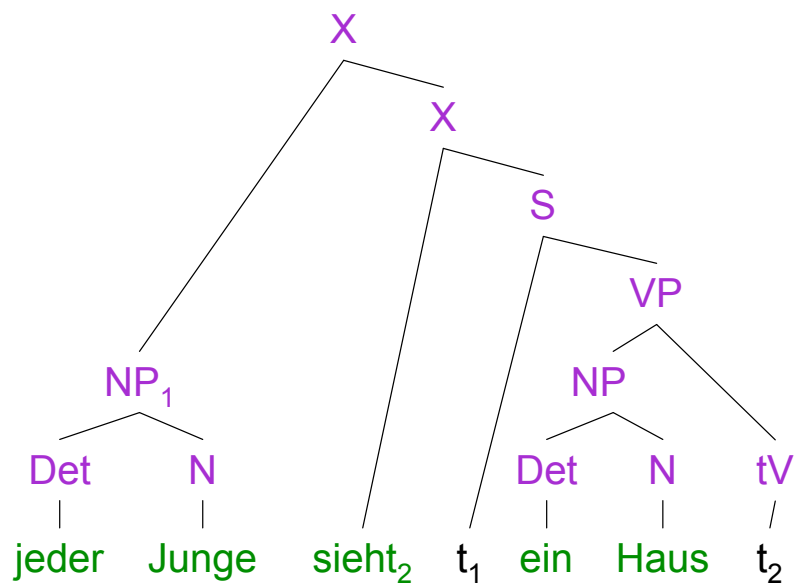
DS: Jeder Junge sieht ein Haus.



Jeder Junge sieht ein Haus.



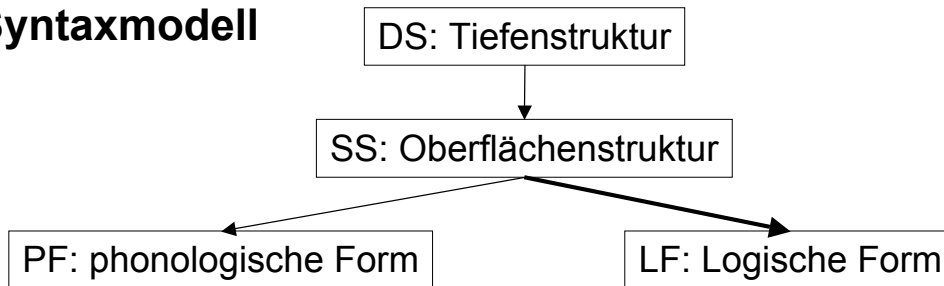
SS: Jeder Junge sieht ein Haus.



Logische Form

- Eine abstrakte (syntaktische) Strukturebene ist für Skopus verantwortlich).

Syntaxmodell



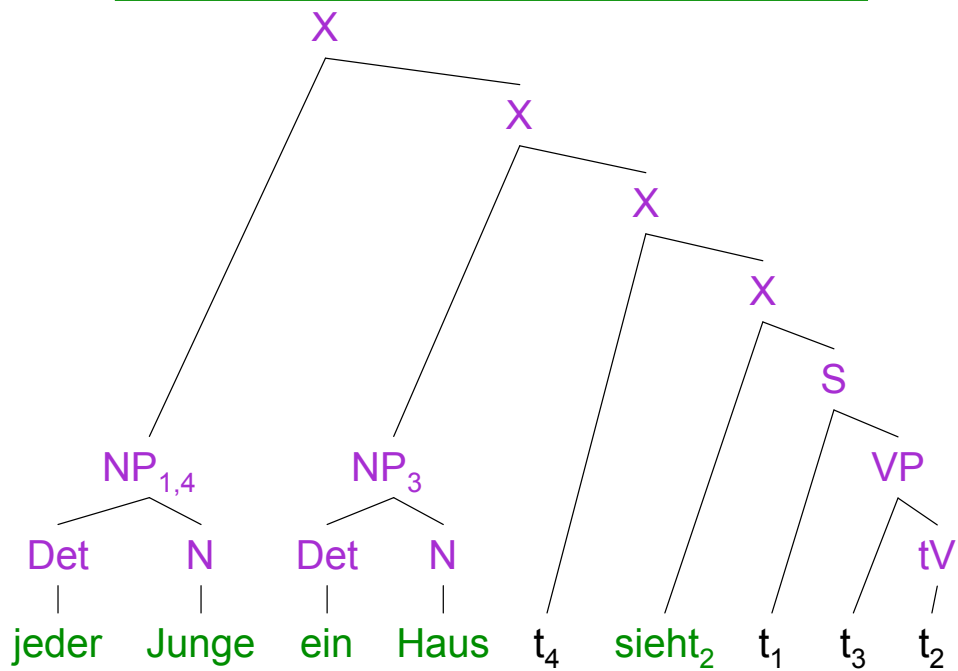
- syntaktische Ebenen mit eigenständigen Wohlgeformtheitsbedingungen

Skopuseffekte: Transformationen SS → LF

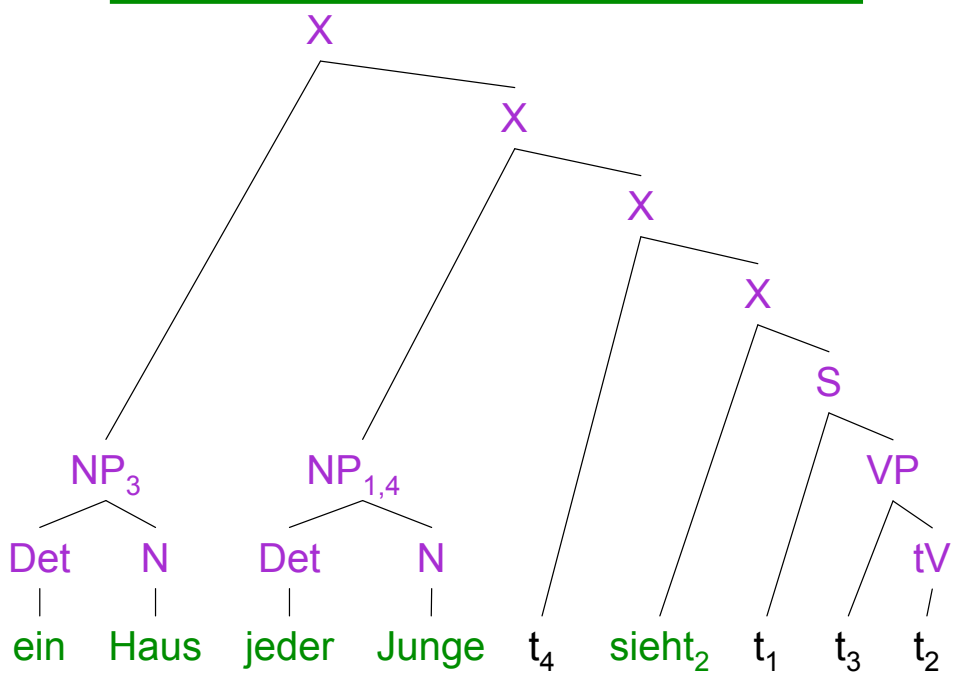
Quantifikationelle NPs werden angehoben

- [jeder Junge]₁ sieht₂ t₁ ein Haus t₂
→ [ein Haus]₃ [jeder Junge]_{1,4} t₄ sieht₂ t₁ t₃ t₂
 $\exists y [\text{haus}(y) \wedge \forall x [\text{junge}(x) \Rightarrow \text{sieht}(x, y)]]$
- [jeder Junge]₁ sieht₂ t₁ ein Haus t₂
→ [jeder Junge]_{1,4} [ein Haus]₃ t₄ sieht₂ t₁ t₃ t₂
 $\forall x [\text{junge}(x) \Rightarrow \exists y [\text{haus}(y) \wedge \text{sieht}(x, y)]]$

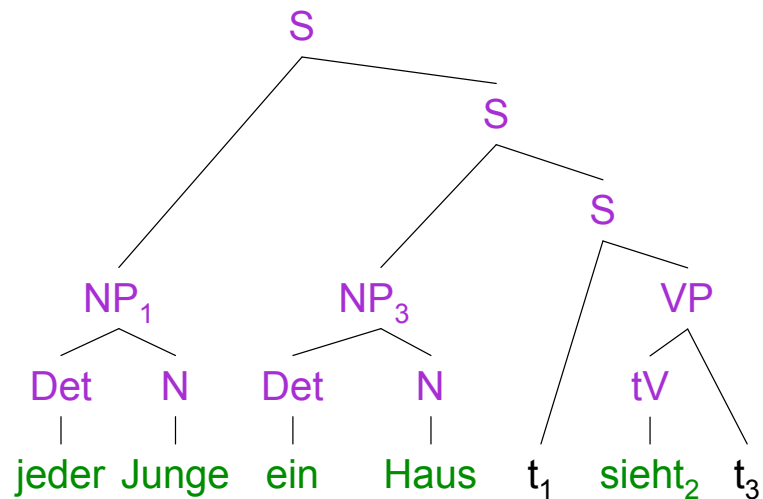
LF: Jeder Junge sieht ein Haus.



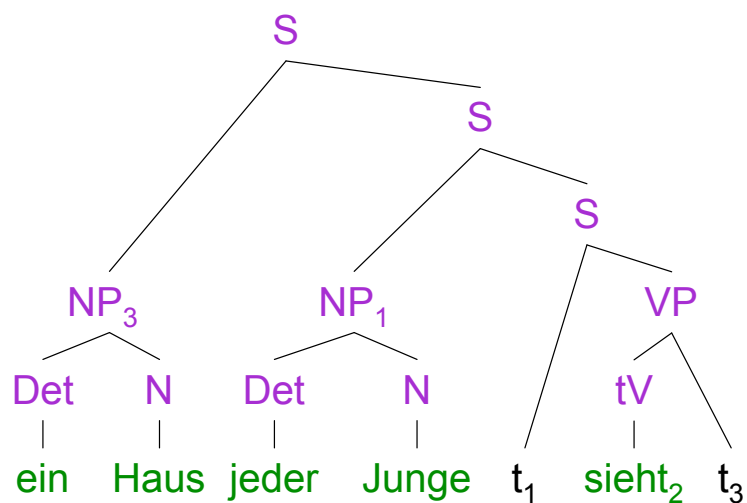
LF: Jeder Junge sieht ein Haus.



vereinfachte LF: Jeder Junge sieht ein Haus.



vereinfachte LF: Jeder Junge sieht ein Haus.



Grammatik von Fragment 1 mit Quantorenliste

Lexikalische Ebene: keine Quantorenliste

$n(n/[Adj, N], Asem@Nsem) \rightarrow adj(Adj, Asem), n(N, Nsem).$

Syntaktische Ebene (keine eingebetteten quantifizierenden NPs): leere Quantorenliste

$np(np/[PN], []/\lambda(P, P@Nsem)) \rightarrow pn(PN, Nsem).$

$vp(vp/[IV], []/Vsem) \rightarrow iv(IV, Vsem).$

$vp(vp/[VP1,C,VP2], []/\lambda(X, (Cs@(VP1s@X))@(VP2s@X))) \rightarrow$
 $vp(VP1, []/VP1s), conj(C, Cs), vp(VP2, []/VP2s).$
 $s(s/[S1,C,S2], []/(Csem@S1sem)@S2sem) \rightarrow$
 $s(S1, []/S1sem), conj(C, Csem), s(S2, []/S2sem).$

Grammatik von Fragment 1 mit Quantorenliste

Syntaktische Ebene (quantifizierenden NPs): Füllung der Quantorenliste

$np(np/[T], [bo(np/[Det, N, T], Dsem@Nsem, Var)]/\lambda(P, P@Var))$
 $\rightarrow det(Det, Dsem), n(N, Nsem), \{gen_trace(T)\}.$

Syntaktische Ebene (Satz): Leerung der Quantorenliste

$s(s/[NPSyn, SSyn], Q/(NPsem@\lambda(Var, Ssem))) \rightarrow$
 $s(SSyn, AllQ/Ssem),$
 $\{select(bo(NPSyn, NPsem, Var), AllQ, Q)\}.$

Syntaktische Ebene: Weiterreichung der Quantorenlisten

$vp(vp/[TV,NP], Q/Vsem@NPsem) \rightarrow tv(TV, Vsem),$
 $np(NP, Q/NPsem).$

$s(s/[NP, VP], Q/Vsem@NPsem) \rightarrow np(NP, QN/NPsem),$
 $vp(VP, QV/Vsem), \{append(QV, NV, Q)\}.$

Ausprobieren ?

Paket: Skopusmehrdeutigkeiten

- Grammatik mit Quantorenanhebung

Literatur

- Blackburn, Patrick & Johan Bos (1999). *Representation and Inference for Natural Language. A First Course in Computational Semantics*. Kapitel 2. Ms. **Auf Anfragen**

Neue Zugriffsmechanismen auf Lexikon

Weiterer Nutzen für Verarbeitung

Trennung von

- Wortform-Information / Lexembeitrag
- Kategorienbeitrag: Argumentstrukturinformation / Kombinationspotential

lexem

- Wortform → Syntaktische Kategorie + Lexembeitrag

semMacro

- Syntaktische Kategorie + Lexembeitrag → Semantischer Beitrag

Kombination

- mit Grammatikregeln in der Schnittstelle zum Lexikon

Beispiel

Anstelle von

- Lexikon

lexem(haus, n, [X]/(haus(X))).

lexem(junge, n, [X]/(junge(X))).

- Lexikon-Grammatik-Schnittstelle

n(n (-Lexem), Sem) --->
lexem(Lexem, n, Sem).

jetzt neu

- Lexikon

lexem(haus, n, haus).

lexem(junge, n, junge).

- Lexikon-Grammatik-Schnittstelle

n(n/ (- Phrase), Sem) --->
lexem(Phrase, n, LexSem),
semMacro(n, LexSem, Sem).

- semantisches Macro

semMacro(n, Predicate,
lambda(X, Predicate@X)).

Kombination von Lexem- und Kategorien-Beitrag

n(n/ (- Phrase), Sem) ---> lexem(Phrase, n, LexSem),
semMacro(n, LexSem, Sem).

adj(adj/ (- Phrase), Sem) ---> lexem(Phrase, adj, LexSem),
semMacro(adj, LexSem, Sem).

det(det/ (- Phrase), Sem) ---> lexem(Phrase, det, LexSem),
semMacro(det, LexSem, Sem).

pn(pn/ (- Phrase), Sem) ---> lexem(Phrase, pn, LexSem),
semMacro(pn, LexSem, Sem).

iv(iv/ (- Phrase), Sem) ---> lexem(Phrase, iv, LexSem),
semMacro(iv, LexSem, Sem).

tv(tv/ (- Phrase), Sem) ---> lexem(Phrase, tv, LexSem),
semMacro(tv, LexSem, Sem).

conj(conj/(- Phrase), Sem) ---> lexem(Phrase, coord, LexSem),
semMacro(coord, LexSem, Sem).

Formenlexikon

lexem(haus, n, haus).

lexem(junge, n, junge).

lexem(peter, pn, peter).

lexem(laura, pn, laura).

lexem(sieht, tv, sieht).

lexem(lacht, iv, lacht).

lexem(winkt, iv, winkt).

lexem(und, coord, conj).

lexem(oder, coord, disj).

lexem(roter, adj, intersective(rot)).

lexem(rote, adj, intersective(rot)).

lexem(rotes, adj, intersective(rot)).

lexem(roten, adj, intersective(rot)).

lexem(rotem, adj, intersective(rot)).

lexem(ein, det, indef).

lexem(eine, det, indef).

lexem(eines, det, indef).

lexem(einen, det, indef)

....

lexem(kein, det, negindef).

...

lexem(jeder, det, uni).

...

Kombination von Inhalt und Argumentstruktur

semMacro(iv, Predicate, Predicate).

semMacro(n, Predicate, Predicate).

semMacro(pn, Constant, Constant).

semMacro(coord, conj, lambda(F, lambda(G, F & G))).

semMacro(coord, disj, lambda(F, lambda(G, F v G))).

semMacro(det, indef, lambda(P, lambda(Q, exists(X, (P@X) & (Q@X)))).

semMacro(det, uni, lambda(P, lambda(Q, forall(X, (P@X) -> (Q@X)))).

semMacro(det, negindef, lambda(P, lambda(Q,
~ exists(X, (P@X) & (Q@X)))).

semMacro(adj, intersective(Predicate),
lambda(P, lambda(X, (Predicate@X) & (P@X))).

semMacro(tv, Rel, lambda(Q, lambda(X, Q@lambda(Z, (Rel@X)@Z))).

Nutzen der Trennung

Ergänzung eines Sprachfragments

- Ergänzung von Grammatikregeln und Lexemen
- semantische Makros, Regeln für Lexikonzugriff können unberührt bleiben

Variation der semantischen Makros

- zur Prüfung / Präsentation / Vergleich verschiedener Theorien
- bei konstanter Grammatik / Lexemmenge