
Semantische Sprachverarbeitung

Carola Eschenbach
Universität Hamburg, FB Informatik
AB Wissens- und Sprachverarbeitung (WSV)

Sommersemester 2004

Semantische Sprachverarbeitung

Sitzung 4: Typen und Lambda-Kalkül

- Motivation, Gegliedertes Lexikon
 - Typen
 - Typisierter Lambda-Kalkül
 - (Skopusambiguitäten)
-

Korrespondenz Syntax - Semantik

Syntax

- Ausdrücke (Wörter, Konstituenten, Phrasen) gehören **syntaktischen Kategorien** an, die ihren grammatischen Positionsmöglichkeiten entsprechen.
- Verb (intransitiv, transitiv, di-transitiv, ...), Verbalphrase
- Nomen, Nominalphrase, Artikel, Determinator, Pronomen

Semantik

- Die Bedeutungen der Ausdrücke haben unterschiedliche **semantische Typen**, die ihrer Funktion im Bedeutungsaufbau entsprechen
 - Proposition, Menge / Eigenschaft, Relation, Quantor, ...
-

Montagues These

Starke Korrespondenz

- Jeder syntaktischen Kategorie entspricht genau ein semantischer Typ auf der Bedeutungsebene
- Bedeutungskombination besteht im wesentlichen aus der Anwendung von Funktionen auf Argumente

Beispiel (vereinfacht)

- Nomen: einstellige Prädikate / Eigenschaften
- Nominalphrasen: Quantoren: Eigenschaften von Eigenschaften (.. später mehr)

Bedeutungen von Lexemen kombinieren

- eigentlichen Inhalt (Prädikat, Relation) und
 - Typ-relevante Anteile
-

Grundidee (Sitzung 3)

die Bedeutung eines Ausdrucks

- beinhaltet eine eigentlichen, inhaltlichen Beitrag
- weist (wohldefinierte) Lücken auf, die der Kontext spezifiziert
- Beitrag und Lücken werden zu prädikatenlogischen Fragmenten zusammengesetzt

der Typ des Ausdrucks

- determiniert, welche Lücken vom Kontext zu füllen sind

Regeln

- determinieren, wie die Lücken gefüllt werden

Lambda-Kalkül

- Abstraktion und Applikation als Lücken-Bildung und Lücken-Füllung

C. Eschenbach: Semantische Sprachverarbeitung, Sommer 2004

4-5

Nutzen für Verarbeitung

Trennung von

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

Lexem

- Wortform → Syntaktische Kategorie + Lexembeitrag

semMacro

- Syntaktische Kategorie + Lexembeitrag → Semantischer Beitrag

Kombination

- mit spezifischen Grammatikregeln

C. Eschenbach: Semantische Sprachverarbeitung, Sommer 2004

4-6

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(roles, 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).
...

C. Eschenbach: Semantische Sprachverarbeitung, Sommer 2004

4-7

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).

C. Eschenbach: Semantische Sprachverarbeitung, Sommer 2004

4-8

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

C. Eschenbach: Semantische Sprachverarbeitung, Sommer 2004

4 – 9

Typen

Semantische Basistypen

- **e**: Entity: Ausdruck denotiert ein Objekt (im weiten Sinn)
- **t**: Truth-value: Ausdruck gibt eine Proposition wieder

Komplexe Typen

- Wenn α und β Typen sind, dann ist $\langle \alpha, \beta \rangle$ ein komplexer Typ.
 - Ausdrücke von Typ $\langle \alpha, \beta \rangle$ denotieren Abbildungen von α -Denotaten auf β -Denotate.
- Das sind alle Typen.

Beispiel

- $\langle e, t \rangle$: Prädikat, Eigenschaft
- $\langle e, \langle e, t \rangle \rangle$: zweistellige Relation
- $\langle \langle e, t \rangle, t \rangle$: gen. Quantor, Abbildung von Eigenschaften auf Propositionen

C. Eschenbach: Semantische Sprachverarbeitung, Sommer 2004

4 – 10

Typen-gesteuerte Grammatik

Typenzuordnung

- jedem Symbol wird ein Typ zugeordnet

Bildung komplexer Ausdrücke

- Wenn α und β Typen sind, **a** ein Ausdruck vom Typ α und **f** ein Ausdruck vom Typ $\langle \alpha, \beta \rangle$, dann ist die Applikation von **f** auf **a**, geschrieben **f(a)**, ein (wohlgeformter) Ausdruck vom Typ β .

- [Zur besseren Lesbarkeit werden aber normalerweise alternative Klammerregeln verwendet]

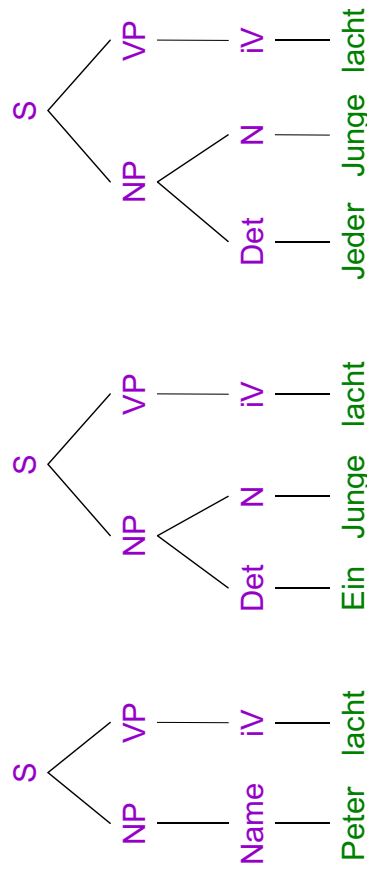
Beispiel

- Prädikat $\langle \langle e, t \rangle \rangle$ und Terme $\langle e \rangle$ bilden Formeln $\langle t \rangle$.
- Negation $\langle \langle t, t \rangle \rangle$ und Formel bilden eine Formel
- \rightarrow Signatur-gesteuerte Sprachdefinition

C. Eschenbach: Semantische Sprachverarbeitung, Sommer 2004

4 – 11

Beispiel: Typen in der natürlichen Sprache

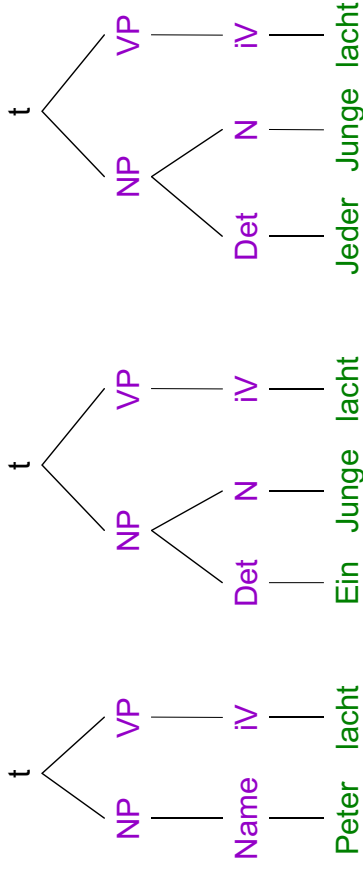


C. Eschenbach: Semantische Sprachverarbeitung, Sommer 2004

4 – 12

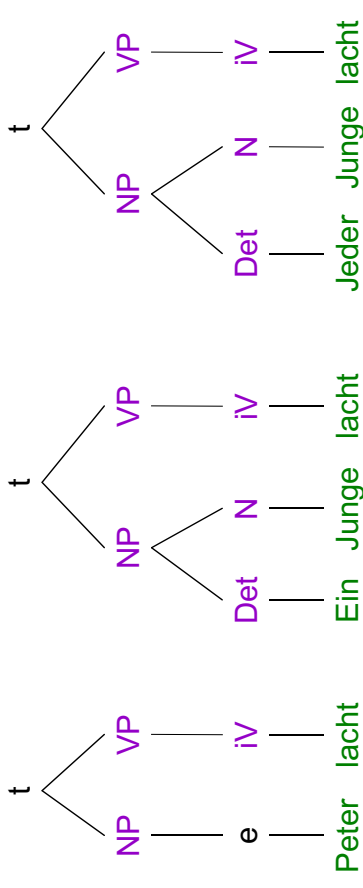
Beispiel: Typen in der natürlichen Sprache

S(ätze) sind vom Typ t



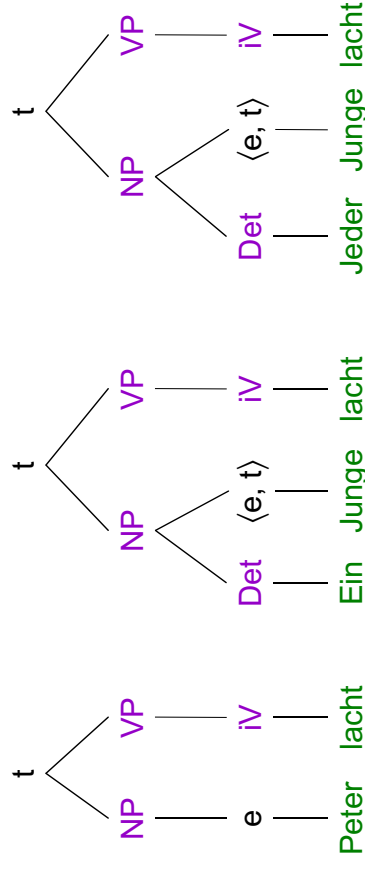
Beispiel: Typen in der natürlichen Sprache

Namen sind vom Typ e



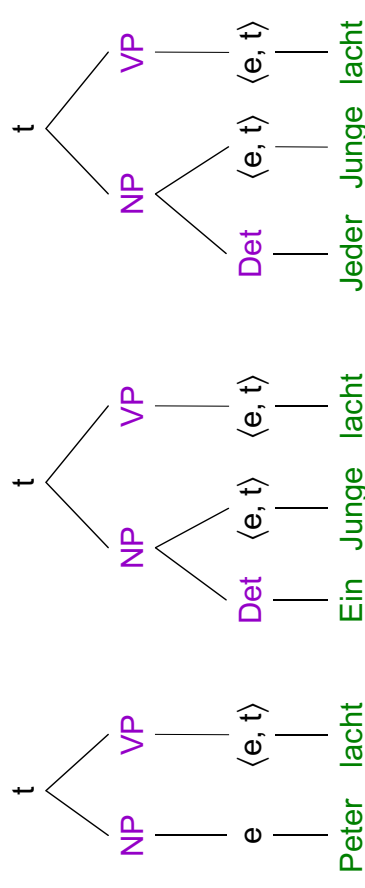
Beispiel: Typen in der natürlichen Sprache

N(omen) sind vom Typ <e, t>



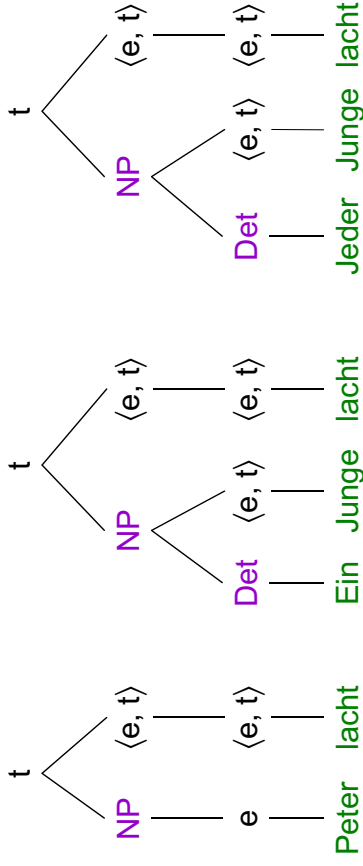
Beispiel: Typen in der natürlichen Sprache

i(ntransitive) V(erben) sind vom Typ <e, t>



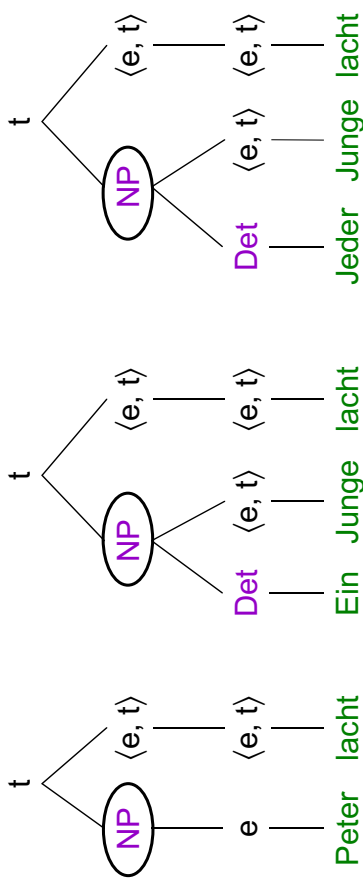
Beispiel: Typen in der natürlichen Sprache

V(erb)alP(hrasen) sind vom Typ $\langle e, t \rangle$



Beispiel: Typen in der natürlichen Sprache

Sind N(ominal)P(hrasen) vom Typ e ?



Nominalphrasen: Generalisierte Quantoren

Jeder Junge lacht

- Jeder Junge \rightarrow die Eigenschaften, die jeder Junge hat
- \rightarrow die Eigenschaft zu lachen gehört zu den Eigenschaften, die jeder Junge hat

Ein Junge lacht

- Ein Junge \rightarrow die Eigenschaften, die mind. ein Junge hat
- \rightarrow die Eigenschaft zu lachen gehört zu den Eigenschaften, die mind. ein Junge hat

Viele Jungen lachen

- Ein Junge \rightarrow die Eigenschaften, die viele Jungen haben
- \rightarrow die Eigenschaft zu lachen gehört zu den Eigenschaften, die viele Jungen haben

Typenanhebung

ohne Anhebung

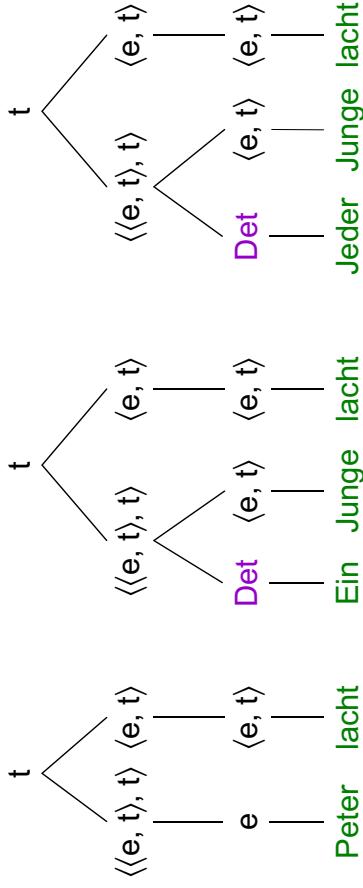
- Peter \rightarrow das Individuum namens ‚Peter‘ e
- lacht \rightarrow die Eigenschaft zu lachen $\langle e, t \rangle$
- Peter lacht \rightarrow Peter hat die Eigenschaft zu lachen t

mit Anhebung

- Peter \rightarrow die Eigenschaft, eine Eigenschaft von Peter zu sein $\langle\langle e, t \rangle, t \rangle$
- lacht \rightarrow die Eigenschaft zu lachen $\langle e, t \rangle$
- Peter lacht \rightarrow die Eigenschaft zu lachen gehört zu den Eigenschaften von Peter t

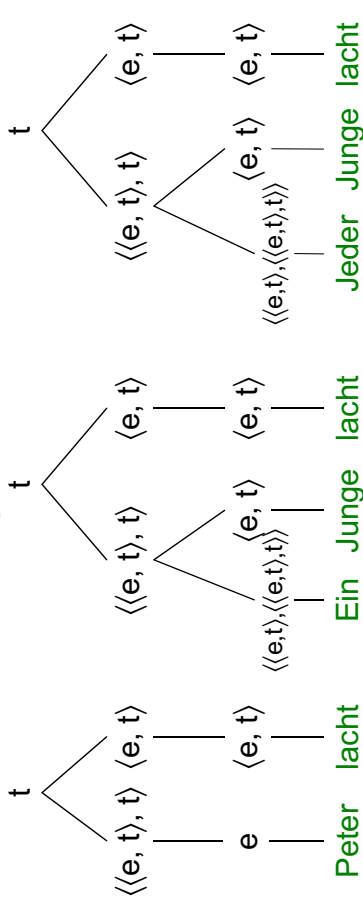
Beispiel: Typen in der natürlichen Sprache

N(ominal)P(hrasen) sind vom Typ $\langle\langle e, t \rangle, t\rangle$
Generalisierte Quantoren



Beispiel: Typen in der natürlichen Sprache

Det(erminatoren) sind vom Typ $\langle\langle e, t \rangle, \langle\langle e, t \rangle, t\rangle\rangle$
Relationen zwischen Eigenschaften



Determinatoren: Relationen zwischen Eigenschaften bzw. Mengen

ein

- die beiden Eigenschaften sind an einem gemeinsamen Objekt realisiert
- die beiden Mengen haben einen nicht-leeren Durchschnitt

jeder

- die N-Eigenschaft ist spezifischer als die V-Eigenschaft
- die N-Menge ist in der V-Menge enthalten

kein

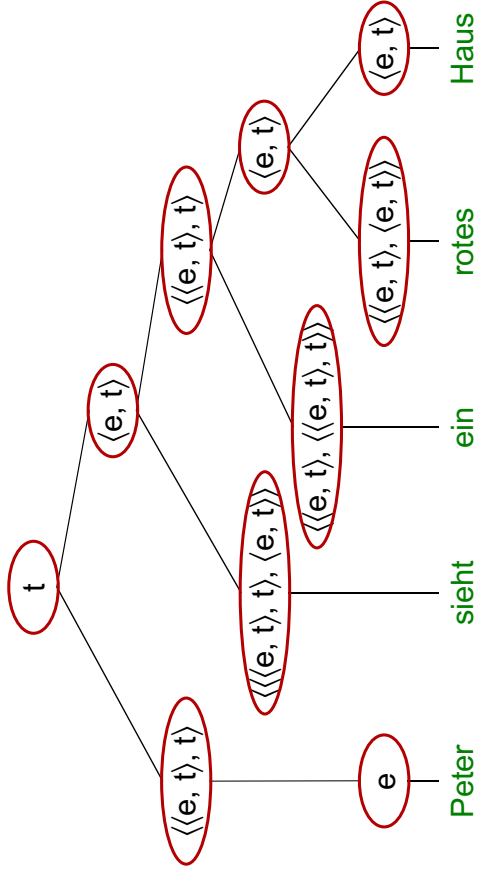
- die beiden Eigenschaften sind nicht an einem gemeinsamen Objekt realisiert
- der Durchschnitt der beiden Mengen ist leer

Wozu der Aufwand?

Generalisierung

- Einheitlichen syntaktischen Strukturen ($[_{S[NP]}] [_{VP}]$) entsprechen einheitlichen semantischen Kombinationsregeln (Anwendung der NP-Bedeutung auf VP-Bedeutung)

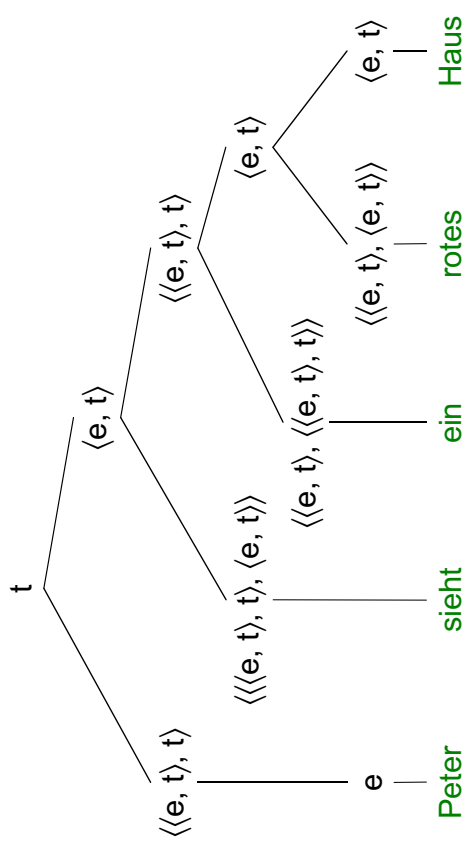
Typen: Peter sieht ein rotes Haus.



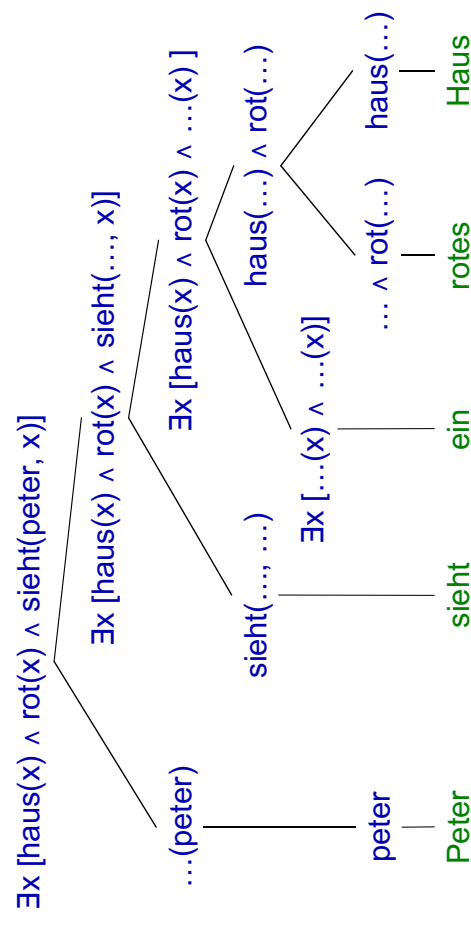
Typen: Peter sieht ein rotes Haus.

Peter	e	peter
sieht	$\langle\langle e, t \rangle, t \rangle, \langle e, t \rangle\rangle$	sieht(..., ...)
ein	$\bullet\langle\langle e, t \rangle, \langle e, t \rangle, t \rangle\rangle$	$\bullet\exists [\dots(x) \wedge \dots(x)]$
rotes	$\langle\langle e, t \rangle, \langle e, t \rangle\rangle$	rotes(...) \wedge ...
Haus	$\langle e, t \rangle$	haus(...)

Typen: Peter sieht ein rotes Haus.



Wahrheitsbedingungen: Peter sieht ein rotes Haus.



Typisierte λ -Ausdrücke

Typenzuordnung

- jedem Symbol (auch den Variablen) wird ein Typ zugeordnet

Bildung komplexer Ausdrücke

- Wenn α und β Typen sind, a ein Ausdruck vom Typ α und f ein Ausdruck vom Typ $\langle \alpha, \beta \rangle$, dann ist die Applikation von f auf a , geschrieben $f(a)$, ein (wohlgeformter) Ausdruck vom Typ β .
- Wenn α und β Typen sind, x eine Variable vom Typ α und b ein Ausdruck vom Typ β , dann ist $\lambda x [b]$, ein (wohlgeformter) Ausdruck (Abstraktion) vom Typ $\langle \alpha, \beta \rangle$.

Beispiel

$\lambda x [(junge(x) \wedge lacht(x))]$ ist vom Typ $\langle e, t \rangle$.
 $\lambda P \lambda x [(P(x) \wedge lacht(x))]$ ist vom Typ $\langle \langle e, t \rangle, \langle e, t \rangle \rangle$.
 $\lambda Q \lambda P \lambda x [(P(x) \wedge Q(x))]$ ist vom Typ $\langle \langle e, t \rangle, \langle \langle e, t \rangle, \langle e, t \rangle \rangle \rangle$.

Interpretation von Applikation und Abstraktion

- Sei D eine Menge und I eine Interpretation in D
- Seien α und β Typen, a ein Ausdruck vom Typ α , x eine Variable vom Typ α , b ein Ausdruck vom Typ β und f ein Ausdruck vom Typ $\langle \alpha, \beta \rangle$

Applikation

- $I(f(a)) = I(f)(I(a))$
- Funktionsausdruck und Argumentsausdruck werden durch I ausgewertet, dann wird die resultierende Funktion auf das resultierende Argument angewendet.

Abstraktion

- $I(\lambda x [b]) = f$, wobei für alle $d \in D_\alpha$ ist $f(d) = I_{[x \rightarrow d]}(b)$
- Es resultiert eine Funktion f , die α -Objekte auf β Objekte abbildet, wobei der Funktionswert danach variiert, wie b abhängig von der Belegung von x interpretiert wird.

Interpretation von Typen

Sei D eine Menge

Semantische Basistypen

- e : Entity: Ausdruck denotiert ein Objekt (im weiten Sinn)
- $D_e = D$
- t : Truth-value: Ausdruck gibt eine Proposition wieder
- $D_t = \{\text{wahr, falsch}\}$

Komplexe Typen

- Wenn α und β Typen sind, dann ist $\langle \alpha, \beta \rangle$ ein komplexer Typ.
- Ausdrücke von Typ $\langle \alpha, \beta \rangle$ denotieren Abbildungen von α -Denotaten auf β -Denotat.
- $D_{\langle \alpha, \beta \rangle} = D_\alpha \rightarrow D_\beta$

Substitution (1)

Definition

Seien x eine Variable und B ein Ausdruck vom selben Typ
Die **Substitution** $\{x / B\}$ ist eine Abbildung von λ -Termen auf λ -Terme, bei der die freien Vorkommen von x durch den Term B ersetzt werden.

Üblicherweise werden Substitutionen hinter ihr Argument geschrieben. Also heißt $F \{x / B\}$, dass die Substitution $\{x / B\}$ auf den Ausdruck F angewendet wird.

Substitution (2)

Definition

Seien x eine Variable und B ein Ausdruck vom selben Typ
Die **Substitution** $\{x / B\}$ ist rekursiv definiert durch:

x wird auf B abgebildet:

- $x \{x / B\} = B$
andere Variablen und Konstanten werden auf sich selbst abgebildet:
- $y \{x / B\} = y$, falls y eine Variable und $y \neq x$
- $c \{x / B\} = c$, falls c eine Konstante ist

Bei der Applikation wird die Substitution auf Funktor und Argument angewendet:

- $[A(C)] \{x / B\} = A\{x / B\}(C\{x / B\})$

Substitution (3)

Definition

Seien x eine Variable und B ein Ausdruck vom selben Typ
Die **Substitution** $\{x / B\}$ ist rekursiv definiert durch:

Ist x durch eine Abstraktor oder Quantor gebunden, dann stoppt die Ersetzung:

- $(\lambda x [A])\{x / B\} = \lambda x [A]$
- $(\exists x [A])\{x / B\} = \exists x [A]$ $[\forall x [A]]\{x / B\} = \forall x [A]$

Ansonsten wird die Ersetzung bei dem eingebetteten Ausdruck fortgesetzt

- $(\lambda y [A])\{x / B\} = \lambda y [A\{x / B\}]$, falls $y \neq x$
- $(\exists y [A])\{x / B\} = \exists y [A\{x / B\}]$, falls $y \neq x$
- $(\forall y [A])\{x / B\} = \forall y [A\{x / B\}]$, falls $y \neq x$

Substitution (2)

Definition

Seien x eine Variable und B ein Ausdruck vom selben Typ
Die **Substitution** $\{x / B\}$ ist rekursiv definiert durch:

x wird auf B abgebildet:

- $x \{x / B\} = B$
andere Variablen und Konstanten werden auf sich selbst abgebildet:
- $y \{x / B\} = y$, falls y eine Variable und $y \neq x$
- $c \{x / B\} = c$, falls c eine Konstante ist

Bei der Applikation wird die Substitution auf Funktor und Argument angewendet:

- $[A(C)] \{x / B\} = A\{x / B\}(C\{x / B\})$

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

rotes Haus

$$\begin{aligned}
 & \lambda P \lambda z [\text{rotes}(z) \wedge P(z)] (\lambda y [\text{haus}(y)]) \\
 \equiv & \lambda z [\text{rotes}(z) \wedge \lambda y [\text{haus}(y)](z)] \\
 \equiv & \lambda z [\text{rotes}(z) \wedge \text{haus}(z)]
 \end{aligned}$$

Peter sieht ein rotes Haus.

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

ein rotes Haus $\lambda R \lambda S \exists x (R(x) \wedge S(x)) (\lambda z [rotes(z) \wedge haus(z)])$

$\equiv \lambda S \exists x (\lambda z [rotes(z) \wedge haus(z)] (x) \wedge S(x))$

$\equiv \lambda S \exists x [rotes(x) \wedge haus(x) \wedge S(x)]$

Peter sieht ein rotes Haus.

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

sieht ein rotes Haus $\lambda Q \lambda u [Q(\lambda w [sieht(u, w)]) (\lambda S \exists x [rotes(x) \wedge haus(x) \wedge S(x)])]$

$\equiv \lambda u [\lambda S \exists x [rotes(x) \wedge haus(x) \wedge S(x)] (\lambda w [sieht(u, w)])]$

$\equiv \lambda u \exists x [rotes(x) \wedge haus(x) \wedge \lambda w [sieht(u, w)](x)]$

Peter sieht ein rotes Haus.

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

sieht ein rotes Haus $\equiv \lambda u \exists x [rotes(x) \wedge haus(x) \wedge \lambda w [sieht(u, w)](x)]$

$\equiv \lambda u \exists x [rotes(x) \wedge haus(x) \wedge sieht(u, x)]$

Peter sieht ein rotes Haus.

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

$[_{NP} [PN peter]] \lambda x \lambda P [P(x)] (peter)$ Type Shifting

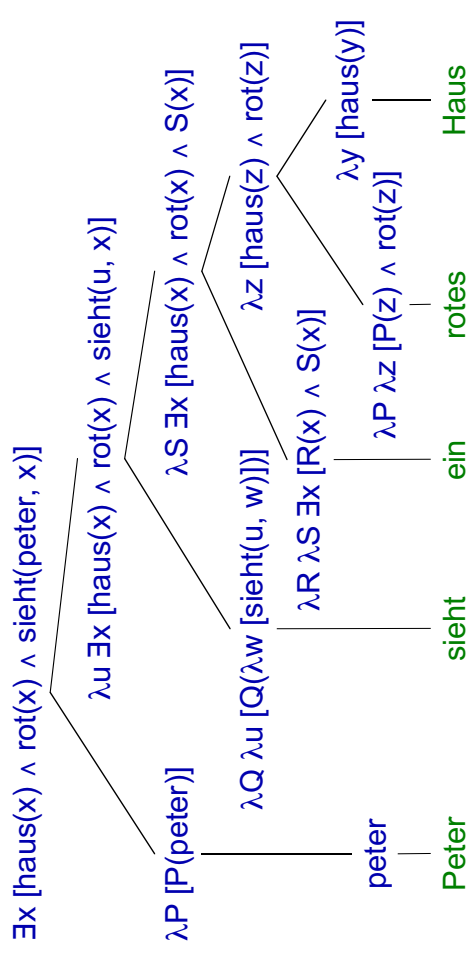
$\equiv \lambda P [P(peter)]$

Peter sieht ein rotes Haus.

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

$peter$ sieht ein rotes Haus
 $\lambda P [P(peter) (\lambda u \exists x [rotes(x) \wedge haus(x) \wedge sieht(u, x)])]$
 $\equiv \lambda u \exists x [rotes(x) \wedge haus(x) \wedge sieht(u, x)](peter)$
 $\equiv \exists x [rotes(x) \wedge haus(x) \wedge sieht(peter, x)]$

Wahrheitsbedingungen: Peter sieht ein rotes Haus.



Jeder Junge winkt.

jeder	$\langle\langle e, t \rangle, \langle\langle e, t \rangle, t \rangle\rangle$	$\bullet\lambda R \lambda S \forall z [R(z) \Rightarrow S(z)]$
Junge	$\langle e, t \rangle$	$\lambda y [junge(y)]$
winkt	$\langle e, t \rangle$	$\lambda x [winkt(x)]$

Diskussion

Jeder Junge zeigt Laura ein Haus.

jeder	$\langle\langle e, t \rangle, \langle\langle e, t \rangle, t \rangle\rangle$	$\bullet\lambda R \lambda S \forall z [R(z) \Rightarrow S(z)]$
Junge	$\langle e, t \rangle$	$\lambda y [junge(y)]$
zeigt	$\langle\langle\langle e, t \rangle, t \rangle, t \rangle, \langle\langle e, t \rangle, t \rangle, \langle e, t \rangle\rangle$	$\lambda T \lambda Q \lambda u [T(\lambda v [Q(\lambda w [zeigt(u, v, w)]))]]$
Laura	e	dat <u>akk</u> laura
ein	$\bullet\langle\langle e, t \rangle, \langle\langle e, t \rangle, t \rangle\rangle$	$\bullet\lambda R \lambda S \exists x [R(x) \wedge S(x)]$
Haus	$\langle e, t \rangle$	$\lambda y [haus(y)]$

Diskussion

Jedes Haus ist rot. Peter ist ein Junge.

jedes	$\langle\langle e, t \rangle, \langle\langle e, t \rangle, t \rangle\rangle$	$\bullet \lambda R \lambda S \lambda z [R(z) \Rightarrow S(z)]$
Haus	$\langle e, t \rangle$	$\lambda y [\text{haus}(y)]$
ist		
rot		
Peter	e	peter
ist		
ein	$\bullet \langle\langle e, t \rangle, \langle\langle e, t \rangle, t \rangle\rangle$	$\bullet \lambda R \lambda S \lambda x [R(x) \wedge S(x)]$
Junge	$\langle e, t \rangle$	$\lambda y [\text{junge}(y)]$

Grammatik von Fragment 2 mit Lambda-Ausdrücken

$s(s([NP, VP], NPsem@VPsem) \text{--->} np(NP, NPsem), vp(VP, VPsem)).$
 $s(s([S1, C, S2], (Csem@S1sem)@S2sem) \text{--->} s(S1, S1sem), conj(C, Csem), s(S2, S2sem)).$
 $np(np([Det, N], Dsem@Nsem) \text{--->} det(Det, Dsem), n(N, Nsem)).$
 $np(np([PN], lambda(P, P@Nsem)) \text{--->} pn(PN, Nsem)).$
 $n(n([Adj, N], Asem@Nsem) \text{--->} adj(Adj, Asem), n(N, Nsem)).$
 $vp(vp([IV], Vsem) \text{--->} iv(IV, Vsem)).$
 $vp(vp([TV, NP], Vsem@NPsem) \text{--->} tv(TV, Vsem), np(NP, NPsem)).$
 $vp(vp([VP1, C, VP2], lambda(X, (Csem@(VP1sem@X))@(VP2sem@X)) \text{--->} vp(VP1, VP1sem), conj(C, Csem), vp(VP2, VP2sem)).$

λ -Terme in PROLOG

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

Kein Symbol-Zeichensatz in PROLOG

λ lambda (X, F) anstelle von $\lambda x [F]$

Keine Zusammengesetzten Funktionsausdrücke in PROLOG

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

Applikation wird repräsentiert, nicht 'ausgeführt'

Vereinfachung: 'Ausführung' der Applikation

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

Kombination von Inhalt und Argumentstruktur

$semMacro(iv, Predicate, lambda(X, Predicate@X)).$
 $semMacro(n, Predicate, lambda(X, Predicate@X)).$
 $semMacro(tv, Rel, lambda(Q, lambda(X, Q@lambda(Z, Rel@X@Z)))).$
 $semMacro(adj, intersective(Predicate), lambda(P, lambda(X, Predicate@X & (P@X))).)$
 $semMacro(pn, Constant, Constant).$
 $semMacro(det, indef, lambda(P, lambda(Q, exists(X, (P@X) & (Q@X)))).)$
 $semMacro(det, uni, lambda(P, lambda(Q, forall(X, (P@X) -> (Q@X)))).)$
 $semMacro(det, negindef, lambda(P, lambda(Q, ~ exists(X, (P@X) & (Q@X)))).)$
 $semMacro(coord, conj, lambda(F, lambda(G, F & G))).$
 $semMacro(coord, disj, lambda(F, lambda(G, F v G))).$

Ausprobieren ?

lambda_sem_frag2de.pl

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

Treiber-Prädikate

```
go(Ex):- example_S(Ex, Phrase), % Wähle Beispiel
write(Ex), write(' '), % Gib Beispiel-Name aus
writeln(Phrase), nl, nl, % Gib Wortfolge aus
analyze(Phrase).
```

```
parse:- readLine(Sentence), analyze(Sentence).
```

```
analyze(Phrase):- recognise(s(T, Sem), Phrase), % Parse
writeFormula(Sem),
pp_syn_sem(T, Sem),
betaConvert(Sem, SimpleSem), % vereinfache
writeFormula(SimpleSem),
pp_syn_sem(T, SimpleSem).
```

C. Eschenbach: Semantische Sprachverarbeitung, Sommer 2004

4- 50

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

C. Eschenbach: Semantische Sprachverarbeitung, Sommer 2004

4- 51

Literatur

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

C. Eschenbach: Semantische Sprachverarbeitung, Sommer 2004

4- 52