Semantik und Pragmatik

SS 2005

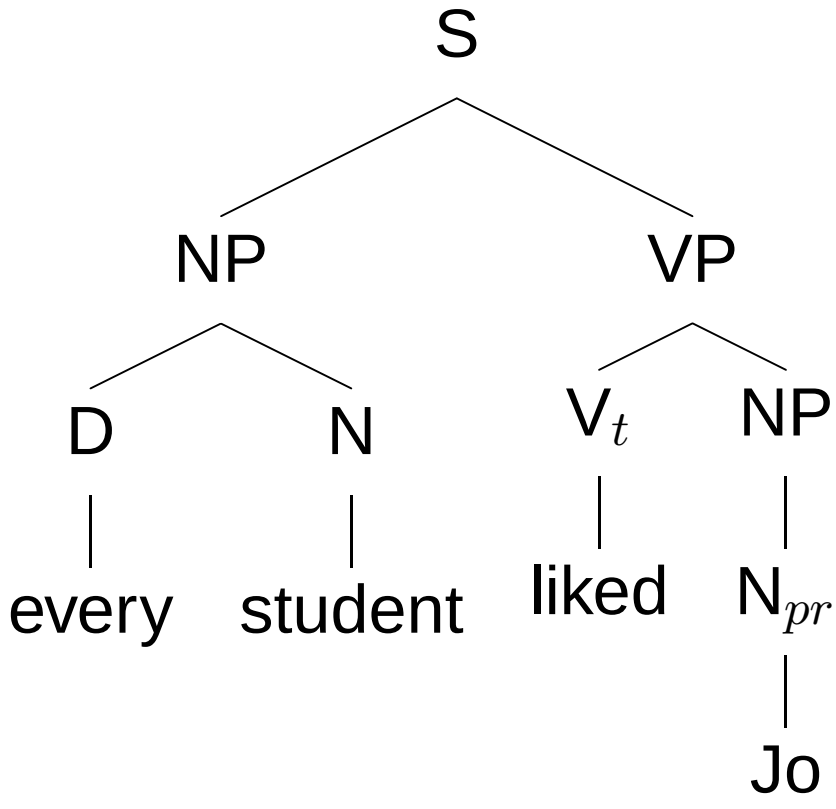
Universität Bielefeld

Teil 10, 24. Juni 2005

Gerhard Jäger

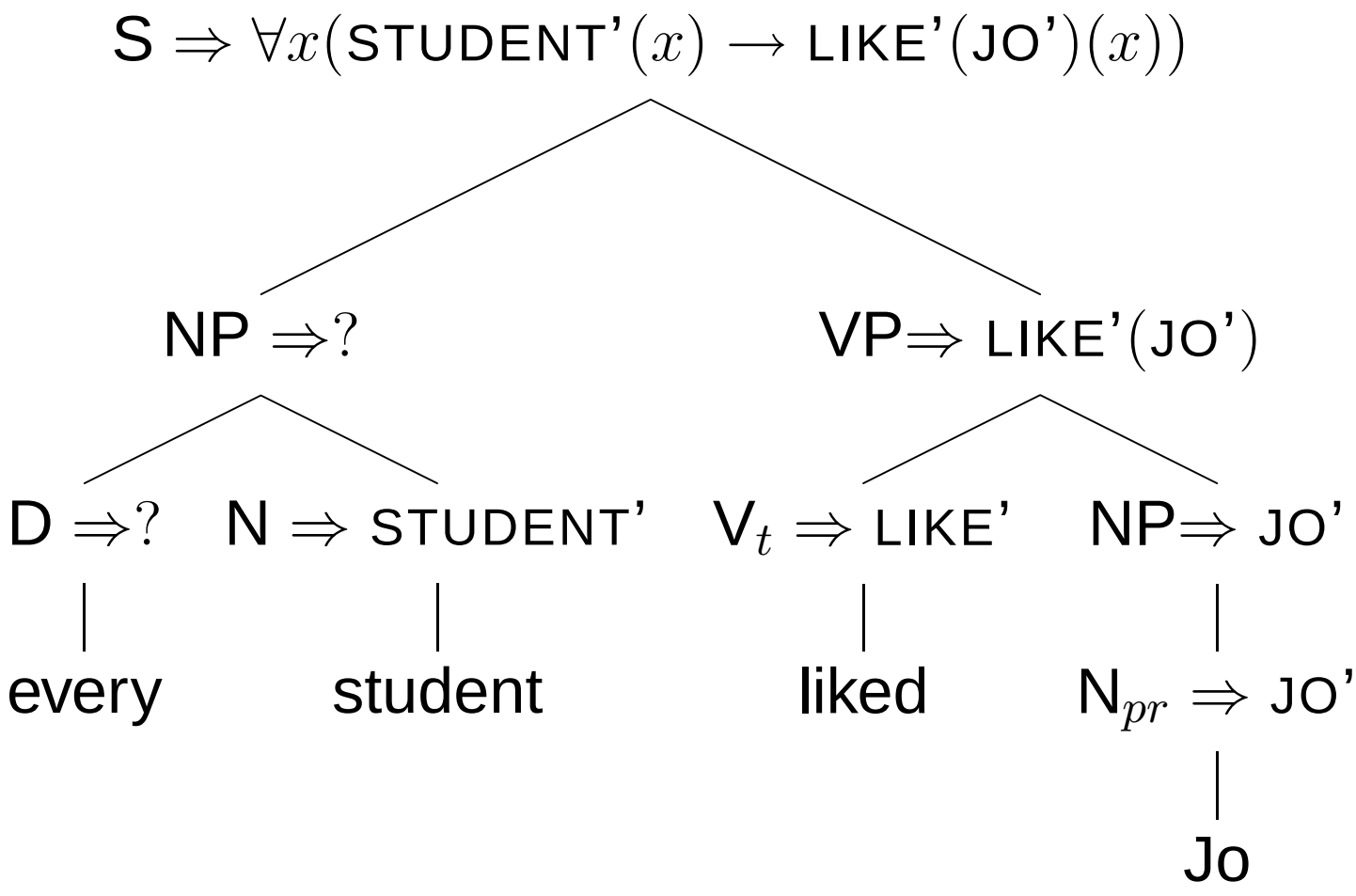
Quantifikation und Kompositionalität

(1) Every student liked Jo.



- bei kompositionaler Übersetzung sollte jeder Knoten übersetzbar sein

Quantifikation und Kompositionalität



Quantifikation und Kompositionalität

- erster Versuch: höherstufige Variablen
  - *every* $\Rightarrow \forall x(P_{\langle e,t\rangle}(x) \rightarrow Q_{\langle e,t\rangle}(x))$
  - *student* $\Rightarrow$ STUDENT'
  - *every student* $\Rightarrow \forall x(\text{STUDENT'}_{\langle e,t\rangle}(x) \rightarrow Q_{\langle e,t\rangle}(x))$
  - *liked Jo* $\Rightarrow$ LIKE'(JO')
  - *every student liked Jo*
    $\Rightarrow \forall x(\text{STUDENT'}(x) \rightarrow \text{LIKE'}(\text{JO'})(x))$

Quantifikation und Kompositionalität

- entsprechende Grammatik müsste folgendermaßen aussehen:
  - $S \rightarrow NP, VP$
  - $S \Rightarrow [VP'/Q]NP'$
  - $NP \rightarrow D, N$
  - $NP \Rightarrow [N'/P]D'$
- würde funktionieren, ist aber unplausibel, weil
  - die Übersetzung von *every* und *every student* dann Formeln vom Typ t wären
  - diese Ausdrücke also wahrheitswert-fähig sein müssten.

Quantifikation und Kompositionalität

- eleganter ist es, wenn Funktionsanwendung die normale semantische Operation ist
- intuitiv einleuchtende Grammatik-Regeln
  - $S \rightarrow NP, VP$
    $S \Rightarrow NP'(VP')$
  - $NP \rightarrow D, N$
    $NP \Rightarrow D'(N')$
- was man also bräuchte:
  - $VP' = \textsc{like'}(\textsc{jo'})$
  - $NP'(\textsc{like'}(\textsc{jo'})) = \forall x(\textsc{student'}(x) \rightarrow \textsc{like'}(\textsc{jo'})(x))$

Der Lambda-Operator

- λ-Operator: Erweiterung der Typentheorie
- erlaubt es, derartige Gleichungen zu lösen:
  - $NP'(\text{LIKE'}(\text{JO'})) = \forall x(\text{STUDENT'}(x) \rightarrow \text{LIKE'}(\text{JO'})(x))$
  - $NP' = \lambda P \forall x(\text{STUDENT'}(x) \rightarrow P(x))$
- allgemeine Definition der **Lambda-Konversion:**

$$(\lambda v \varphi)(\alpha) = [\alpha / v]\varphi$$

- Seitenbedingung (spielt für unsere Anwendungen keine Rolle): alle Variablen, die auf der linke Seite der Gleichung frei sind, sind auch auf der rechten Seite frei
- also:

$$(\lambda v_t \exists x v)(P(x)) \neq \exists x P(x)$$

Typ von Lambda-Ausdrücken

- wie Quantoren hat λ einen **Skopus** und **bindet eine Variable**
- λ-Ausdruck besteht aus drei Teilen:
  - λ
  - Variable
  - Skopus des λ-Operators
- λ-Operator kreiert Funktoren:
  - $\varphi : a$
  - $v : b$
  - $\lambda v \varphi : \langle b, a \rangle$

Syntax der Typentheorie

Definition 2 (Syntax der Typentheorie, endgültige Version)

1. *Von jedem Typ gibt es unendlich viele Variablen.*
2. *Von jedem Typ gibt es unendlich viele Konstanten.*
3. *Eine Konstante oder Variable eines Typs ist ein Ausdruck dieses Typs.*
4. *Wenn φ und ψ Ausdrücke vom Typ t sind, dann sind $\neg\varphi, \varphi \wedge \psi, \varphi \vee \psi, \varphi \rightarrow \psi, \varphi \leftrightarrow \psi$ auch Ausdrücke vom Typ t.*
5. *Wenn α ein Ausdruck vom Typ $\langle a, b\rangle$ ist und β ein Ausdruck vom Typ a, dann ist ein Ausdruck $\alpha(\beta)$ vom Typ b.*
6. *Wenn v eine Variable ist und φ ein Ausdruck vom Typ t, dann sind $\forall v(\varphi)$ und $\exists v(\varphi)$ ebenfalls Ausdrücke vom Typ t.*
7. *Wenn α ein Ausdruck vom Typ a ist und v eine Variable vom Typ b, dann ist $\lambda v\alpha$ ein Ausdruck vom Typ $\langle b, a\rangle$.*
8. *Nichts sonst ist ein Ausdruck.*

Semantik der Typentheorie

Definition 3 (Interpretation der Typentheorie (endgültige Version))
Sei $M = \langle E, F \rangle$ ein Modell für die Typentheorie, und g eine Belegungsfunktion für M.

- $[\alpha]_g^M = F(\alpha)$, *wenn α eine Konstante ist*
- $[v]_g^M = g(v)$, *wenn v eine Variable ist*
- $[\neg\varphi]_g^M = 1 - [\varphi]_g^M$
- $[\varphi \wedge \psi]_g^M = \min([\varphi]_g^M, [\psi]_g^M)$
- $[\varphi \vee \psi]_g^M = \max([\varphi]_g^M, [\psi]_g^M)$
- $[\varphi \rightarrow \psi]_g^M = \max(1 - [\varphi]_g^M, [\psi]_g^M)$
- $[\varphi \leftrightarrow \psi]_g^M = 1 - ([\varphi]_g^M - [\psi]_g^M)^2$
- $[\alpha(\beta)]_g^M = [\alpha]_g^M([\beta]_g^M)$
- $[\forall v_a(\varphi)]_g^M = \min(\{[\varphi]_{g[l/v]}^M | l \in Dom(a)\})$
- $[\exists v_a(\varphi)]_g^M = \max(\{[\varphi]_{g[l/v]}^M | l \in Dom(a)\})$
- $[\lambda v_a(\alpha)]_g^M = \{\langle l, [\alpha]_{g[l/v]}^M \rangle | l \in Dom(a)\}$

Semantik des Lambda-Operators

Theorem 1 *Wenn durch die Lambda-Konversion keine freien Variablen gebunden werden, gilt für alle Modelle* M *und Belegungsfunktionen* g*:*

$$[(\lambda v \alpha)\beta]_g^M = [[\beta/v]\alpha]_g^M$$

Auf der Ebene der typentheoretischen Übersetzung natürlichsprachlicher Ausdrücke darf also jederzeit Lambda-Konversion durchgeführt werden, ohne dass sich dadurch an der eigentlichen Semantik etwas ändert.

Beispiel-Grammatik: Syntax

1. $S \rightarrow NP, VP$
   $S \Rightarrow NP'(VP')$

2. $NP \rightarrow N_{pr}$
   $NP \Rightarrow \lambda P(P(N'_{pr}))$

3. $NP \rightarrow D, N$
   $NP \Rightarrow D'(N')$

4. $VP \rightarrow V_i$
   $VP \Rightarrow V'_i$

5. $VP \rightarrow V_t, NP$
   $VP \Rightarrow \lambda z(NP'(\lambda y(V'_t(y)(z))))$

Beispiel-Grammatik: Lexikon

- $N_{pr} \rightarrow$ John
  $N_{pr} \Rightarrow$ JOHN'
- $D \rightarrow$ some
  $D \Rightarrow \lambda P \lambda Q \exists x(P(x) \wedge Q(x))$
- $D \rightarrow$ every
  $D \Rightarrow \lambda P \lambda Q \forall w(P(w) \rightarrow Q(w))$
- $N \rightarrow$ student
  $N \Rightarrow$ STUDENT'
- $N \rightarrow$ book
  $N \Rightarrow$ BOOK'
- $V_i \rightarrow$ walked
  $V_i \Rightarrow$ WALK'
- $V_t \rightarrow$ read
  $V_t \Rightarrow$ READ'