

Semantik und Pragmatik

15. Mai 2007

Gerhard Jäger

Lambda-Notation mit Angabe des Definitionsbereichs

- Funktionen haben Definitionsbereich:

$$\{\langle x, x^2 \rangle \mid x \in N\} \neq \{\langle x, x^2 \rangle \mid x \in R\}$$

- Notation $\lambda x. x^2$ deshalb unvollständig
- vollständige Notation: Angabe des Definitionsbereichs im Lambda-Präfix:
 - $\lambda x \in N. (x^2)$
 - $\lambda x \in R. (x^2)$
- allgemeines Format:
 $\lambda \text{ Variable} \in \text{Definitionsbereich. (Beschreibung des Wertes der Variable)}$

Lambda-Notation mit Angabe des Definitionsbereichs

- Beispiel
 - $(\lambda x \in R.(x^2 + 3x + 2))(-10) = 72$
 - $(\lambda x \in N.(x^2 + 3x + 2))(-10)$ ist nicht definiert
- Angabe des Definitionsbereichs sowie Klammern um Beschreibung des Wertes werden häufig weggelassen, wenn dadurch keine Ambiguität auftritt

Variablen-Konventionen

- Schreibweise mit explizitem Definitionsbereich ist umständlich
- Vereinfachung durch Variablen-Konventionen:
 - Jeder Variablen-Name ist per Konvention mit bestimmten Definitionsbereich assoziiert:
 - $x, y, z, \dots: E$ (Menge der Individuen)
 - $s, s', s_1, s_2, \dots: S$ (Menge der Situationen)
 - $P, Q, P', \dots: S \times E$ (Menge der Relationen zwischen Situationen und Individuen)
 - $R, S, \dots: S \times E \times E$ (Menge der Relationen zwischen Situationen und Paaren von Individuen)
 - $p, q, \dots: POW(S)$ (Menge der Mengen von Situationen)

Variablen-Konventionen

- soweit nicht anders angegeben, wird implizit angenommen, dass jede Variable nur in dem Bereich Werte nehmen kann, mit dem der Variablenname assoziiert ist

- es gilt also:

$\lambda x.\phi$ ist eine Abkürzung für $\lambda x \in E.\phi$

$\lambda s'.\phi$ ist eine Abkürzung für $\lambda s' \in S.\phi$

$\lambda P.\phi$ ist eine Abkürzung für $\lambda P \in S \times E.\phi$

$\lambda p.\phi$ ist eine Abkürzung für $\lambda p \in POW(S).\phi$

usw.

Funktionen mit Funktionen als Argumente

- Argument einer Funktion kann komplex sein:
 - Argument ist eine Menge
 - $\lambda X \in POW(N).(X \cap \{1, 2, 3\})$
 - $(\lambda X \in POW(N).(X \cap \{1, 2, 3\}))(\{2, 3, 4\}) = \{2, 3, 4\} \cap \{1, 2, 3\} = \{2, 3\}$
 - $(\lambda X \in POW(N).(X \cap \{1, 2, 3\}))(\{4, 5, 6\}) = \{4, 5, 6\} \cap \{1, 2, 3\} = \emptyset$
 - $(\lambda X \in POW(N).(X \cap \{1, 2, 3\}))(Isaak)$ ist nicht definiert
 - Argument ist selbst eine Funktion
 - $\lambda f \in N \mapsto N.(f(3))$
 - $(\lambda f \in N \mapsto N.(f(3)))(\lambda x \in N.(x^2)) = (\lambda x \in N.x^2)(3) = 3^2 = 9$

Funktionen mit Funktionen als Argumente

Weitere Beispiele:

$$\begin{aligned}(\lambda f.(f(3) + f(4)))(\lambda x.x^2 + x + 1) &= (\lambda x.x^2 + x + 1)(3) + (\lambda x.x^2 + x + 1)(4) \\ &= 3^2 + 3 + 1 + 4^2 + 4 + 1 \\ &= 34\end{aligned}$$

$$\begin{aligned}(\lambda f.f(f(3) - 9))(\lambda x.x^2 + x + 1) &= (\lambda x.x^2 + x + 1)((\lambda x.x^2 + x + 1)(3) - 9) \\ &= (\lambda x.x^2 + x + 1)((3^2 + 3 + 1) - 9) \\ &= (\lambda x.x^2 + x + 1)(4) \\ &= 4^2 + 4 + 1 \\ &= 21\end{aligned}$$

Funktionen mit Funktionen als Wert

Gleichermaßen können Funktionen auch Werte haben, die wieder Funktionen sind, z.B.

- $\lambda x \lambda y. x + y$
 - $((\lambda x (\lambda y. x + y))(2))(3) =$
 - $= (\lambda y. 2 + y)(3)$
 - $= 2 + 3 = 5$
- in der Notation haben solche funktionswertige Funktionen mehrere Lambda-Operatoren hintereinander
- es gilt die Konvention:
 - Lambda-Operatoren werden von links nach rechts geklammert
 - Argumente werden von rechts nach links geklammert
 - erstes Lambda gehört zu erstem Argument, zweites Lambda zu zweitem Argument usw.

Funktionen mit Funktionen als Wert

$$(\lambda x_1. \dots . \lambda x_n. \alpha)(a_1) \dots (c_n)$$

ist eine Abkürzung für

$$(((\lambda x_1. (\dots . (\lambda x_n. (\alpha)(a_1)))))) \dots)(c_n))$$

Skopus, Variablenbindung, Variablenumbenennung

- λ -Operator verhält sich in vielerlei Hinsicht wie ein Quantor in der Prädikatenlogik
- wie in der Prädikatenlogik ist die Wahl des Variablennamens unwesentlich:

$$\begin{aligned}\forall x(P(x) \rightarrow Q(x)) &= \forall y(P(y) \rightarrow Q(y)) \\ \lambda x.x^2 + 3x + 4 &= \lambda w.w^2 + 3w + 4\end{aligned}$$

- wichtig ist nur welche Variablenvorkommen gleichnamig sind und welche ungleichnamig

Skopus, Variablenbindung, Variablenumbenennung

- **Skopus** eines Lambda-Operators: Ausdruck in Klammern, der der Variable und dem Punkt folgt

$$(\lambda f. \underbrace{f(f(3) - 9)})(\lambda x. \underbrace{x^2 + x + 1})$$

- alle freien Vorkommen der Variablen, die nach dem λ steht, im Skopus des λ -Operators werden durch den Operator **gebunden**
- eine Variable, die nicht gebunden ist (weder durch ein λ noch durch einen Quantor), heißt **frei**
- Eine Variable im λ -Operator kann umbenannt werden, wenn
 - alle Variablenvorkommen, die durch den Operator gebunden werden, ebenfalls umbenannt werden
 - die Bindungsrelationen dadurch nicht zerstört werden

Charakteristische Funktionen in Lambda-Notation

- Charakteristische Funktion χ_M einer Menge M :
 - Wertebereich: $\{0, 1\}$
 - Bildungsvorschrift: $\chi_M(x) = 1$ gdw. $x \in M$, sonst 0
- Bedeutung von meta-sprachlichen Sätzen ist immer „wahr“ (bzw. 1) oder „falsch“ (bzw. 0)
- deshalb kann charakteristische Funktion als λ -Term ausgedrückt werden:

$$\lambda x. x \in M$$

- Beispiel:
 - angenommen, $M = \{x \mid x \text{ ist ein Mensch}\}$
 - dann: $\chi_M = \lambda x. x \text{ ist ein Mensch}$

Mengen können generell als Lambda-Terme dargestellt werden.

Darstellung von Bedeutungen in Lambda-Notation

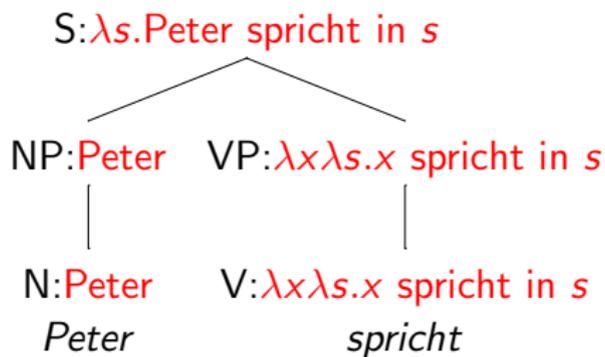
- Ob ein Individuum eine Eigenschaft hat oder nicht, ist situationsabhängig
- Situationsabhängigkeit muss deshalb in lexikalischer Bedeutung verankert sein:
 - $\|Pferd\| = \lambda x \lambda s. x \text{ ist ein Pferd in } s$
 - $\|rot\| = \lambda x \lambda s. x \text{ ist rot in } s$
 - $\|spricht\| = \lambda x \lambda s. x \text{ spricht in } s$
 - $\|Peter \text{ spricht}\| = \lambda s. Peter \text{ spricht in } s$

- Satzbedeutung = lexikalische Bedeutungen + Syntax

- Beispiel:

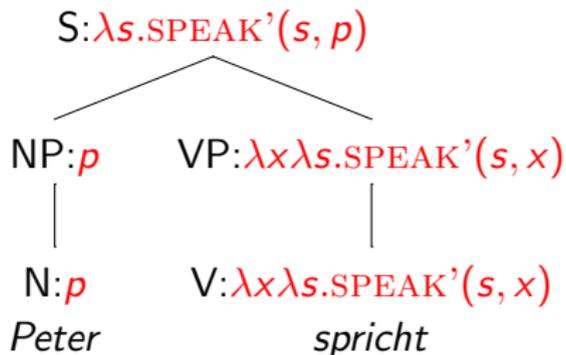
Peter spricht.

- Satzbedeutung: $\lambda s.Peter$ spricht in s
- lexikalische Bedeutungen:
 - $\|Peter\| = Peter$
 - $\|spricht\| = \lambda x \lambda s.x$ spricht in s
- Syntax: $[_S [_{NP} [_N Peter]] [_{VP} [_V spricht]]]$



Kompositionalität

- Bis jetzt wurde als Meta-Sprache Deutsch + Lambda-Notation verwendet
- Prädikatenlogik ist präziser als Deutsch und deshalb als Meta-Sprache vorzuziehen
- beachte: alle Prädikate haben (anders als in der Standard-Übersetzung) ein zusätzliches Argument für Situationen



- Bedeutung des Mutterknotens ergibt sich eindeutig aus Bedeutungen der Tochterknoten:
 - bei nicht-verzweigenden Knoten sind Bedeutung von Mutter- und Tochterknoten identisch
 - bei NP-VP-Struktur wird Bedeutung der VP (eine Funktion) auf Bedeutung der NP angewendet
- Annahme: diese Korrespondenz zwischen Syntax und Semantik gilt für alle Sätze des Deutschen (wobei die korrekte Syntax des Deutschen natürlich viel komplexer ist, aber das gehört nicht hierher)

- formal: für jede Syntax-Regel gibt es eine korrespondierende semantische Regel
- bis jetzt sind das:
 - $S \rightarrow NP, VP :: \|S\| = \|VP\|(\|NP\|)$
 - $NP \rightarrow N :: \|NP\| = \|N\|$
 - $VP \rightarrow V :: \|VP\| = \|V\|$

Schönfinkelisierung

- Bedeutung transitiver Verben: **zweistellige Relation**
- z.B.: *lieben* \rightsquigarrow $\{\langle x, y \rangle \mid \text{LOVE}'(x, y)\}^1$
- Darstellung als charakteristische Funktion:

$$\lambda \langle x, y \rangle \in E \times E. \text{LOVE}'(x, y)$$

- Lambda-Konversion:

$$(\lambda \langle x, y \rangle \in E \times E. \text{LOVE}'(x, y))(\langle a, h \rangle) = \text{LOVE}'(a, h)$$

¹Wir ignorieren für den Moment die Situationsabhängigkeit.

Schönfinkelisierung

- Was ist Bedeutung von *liebt Hans*? Die Menge der Individuen, die Hans lieben.

$$\| \textit{liebt Hans} \| = \{x \mid \text{LOVE}'(x, h)\} \approx \lambda x. \text{LOVE}'(x, h)$$

- *liebt* kann auch als Funktion aufgefasst werden, die Bedeutung von α auf Bedeutung von *liebt* α abbildet:

$$\| \textit{liebt} \| = \lambda y \lambda x. \text{LOVE}'(x, y)$$

Schönfinkelisierung

- zweistellige Relation $\{\langle x, y \rangle \mid \text{LOVE}'(x, y)\}$ wird also umgewandelt in zweistellige charakteristische Funktion $\lambda\langle x, y \rangle. \text{LOVE}'(x, y)$, und diese in eine einstellige Funktion, deren Wert eine einstellige charakteristische Funktion ist:

$$\lambda y \lambda x. \text{LOVE}'(x, y)$$

- generelle Technik:

$$\{\langle x, y \rangle \mid R(x, y)\} \rightsquigarrow \lambda\langle x, y \rangle. R(x, y) \rightsquigarrow \lambda y \lambda x. R(x, y)$$

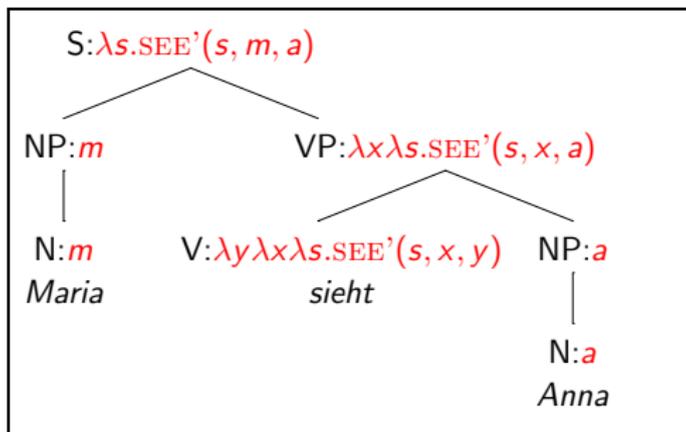
- auch auf mehrstellige Relationen anwendbar:

$$\{\langle x_1, \dots, x_n \rangle \mid S(x_1, \dots, x_n)\} \rightsquigarrow \lambda x_n. \dots \lambda x_1. S(x_1, \dots, x_n)$$

Beachte: Reihenfolge der Variablen im Lambda-Präfix ist Spiegelbild der Reihenfolge im Argumentraster der Relation!

- Beispiele: *lieben, kennen, sehen, helfen, ...*
- drücken **zweistellige Relationen** zwischen Individuen aus
- plus Situationsabhängigkeit: dreistellige Relation
- $\| \textit{Maria sieht Anna} \| = \lambda x. \text{SEE}'(s, m, a)$
- $\| \textit{sieht} \| = \lambda y \lambda x \lambda s. \text{SEE}'(s, x, y)$

Transitive Verben



Regeln:

- $S \rightarrow NP, VP ::$
 $\|S\| = \|VP\|(\|NP\|)$
- $NP \rightarrow N ::$
 $\|NP\| = \|N\|$
- $VP \rightarrow V ::$
 $\|VP\| = \|V\|$
- $VP \rightarrow V, NP ::$
 $\|VP\| = \|V\|(\|NP\|)$

Die kompositionale Analyse der Booleschen Operationen kann auch in dem neuen Format ausgedrückt werden:

Negation

- Logischer Operator der Negation kann auf zweierlei Weise im Dt. ausgedrückt werden:
 - *Es ist nicht der Fall, dass Peter spricht.*
 - *Peter spricht nicht.*
- semantischer Effekt ist in beiden Fällen
Komplementmengenbildung:

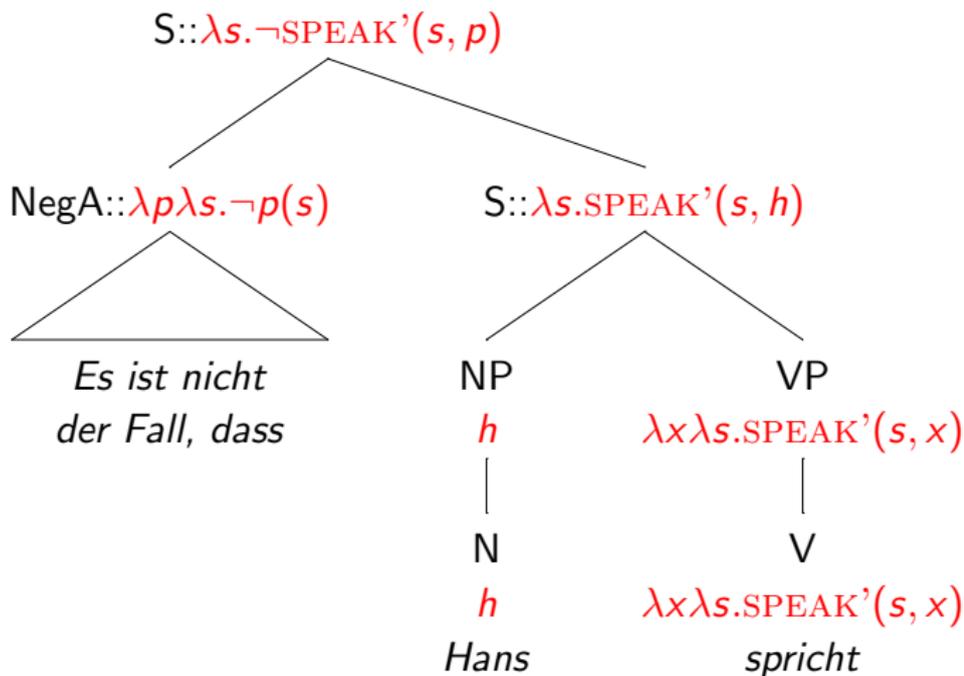
$$\| \textit{Peter spricht nicht} \| = \lambda s. \neg \text{SPEAK}'(s, p)$$

Negation

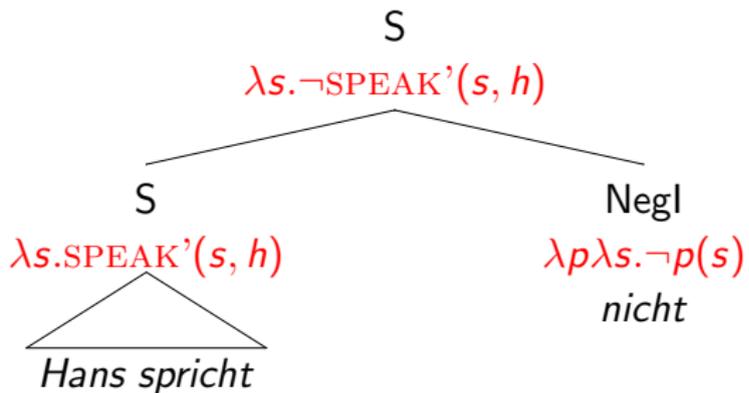
- Neue Regeln:
 - $S_1 \rightarrow \text{Neg}A, S_2 :: \|S_1\| = \|\text{Neg}A\|(\|S_2\|)$
 - $S_1 \rightarrow S_2, \text{Neg}I :: \|S_1\| = \|\text{Neg}I\|(\|S_2\|)$
 - $\text{Neg}A \rightarrow \text{Es ist nicht der Fall, dass} :: \|\text{Neg}A\| = \lambda p \lambda s. \neg p(s)$
 - $\text{Neg}I \rightarrow \text{nicht} :: \|\text{Neg}I\| = \lambda p \lambda s. \neg p(s)$

Boolsche Operatoren

Negation



Negation



Satz-Koordination

- Regeln:

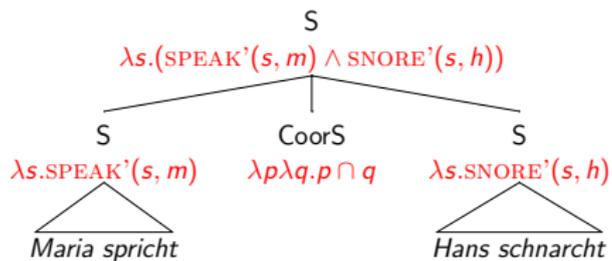
- $S_1 \rightarrow S_2, \text{CoorS}, S_3 :: \|S_1\| = \| \text{CoorS} \| (\|S_2\|) (\|S_3\|)$
- $\text{CoorS} \rightarrow \text{und} :: \lambda p \lambda q. p \cap q$
- $\text{CoorS} \rightarrow \text{oder} :: \lambda p \lambda q. p \cup q$

- Merke:

$$\lambda s. \phi \cap \lambda s. \psi = \lambda s. (\phi \wedge \psi)$$

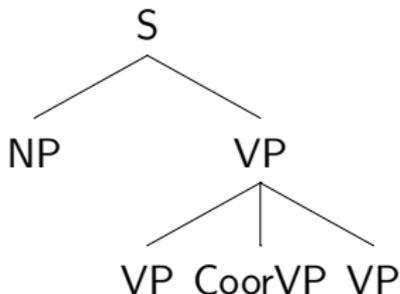
$$\lambda s. \phi \cup \lambda s. \psi = \lambda s. (\phi \vee \psi)$$

Satz-Koordination



VP-Koordination

- Koordination kann auch zwei VPn verknüpfen:
 - *Peter schläft und schnarcht.*
 - *Hans läuft oder steht.*
- syntaktische Struktur:



- Semantik: analog zu Satzoperatoren
Peter schläft und schnarcht \Leftrightarrow *Peter schläft und Peter schnarcht.*

VP-Koordination

- Regeln:

- $VP_1 \rightarrow VP_2, \text{CoorVP}, VP_3 ::$

- $\|VP_1\| = \|CoorVP\|(\|VP_2\|)(\|VP_3\|)$

- $CoorVP \rightarrow \text{und} :: \lambda P \lambda Q \lambda x \lambda s. P(x)(s) \wedge Q(x)(s)$

- $CoorVP \rightarrow \text{oder} :: \lambda P \lambda Q \lambda x \lambda s. P(x)(s) \vee Q(x)(s)$

VP-Koordination

