Semantik und Pragmatik

13. Mai 2008

Gerhard Jäger

Lambda-Notation

Lambda-Notation mit Angabe des Definitionsbereichs

- Funktionen haben Definitionsbereich:

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

- 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:
  λ Variable $\in$ Definitionsbereich.(Beschreibung des Wertes der Variable)

Lambda-Notation

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

Lambda-Notation

Variablen-Konventionen

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

Lambda-Notation

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.

Lambda-Notation

Funktionen mit Funktionen als Argumente

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

Lambda-Notation

Funktionen mit Funktionen als Argumente
Weitere Beispiele:

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

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

Lambda-Notation

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.

Lambda-Notation

Funktionen mit Funktionen als Wert

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

ist eine Abkürzung für

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

Lambda-Notation

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:

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

- wichtig ist nur welche Variablenvorkommen gleichnamig sind und welche ungleichnamig

Lambda-Notation

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

Lambda-Notation

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, $M = \{x | x \textit{ ist ein Mensch}\}$
  - dann: $\chi_M = \lambda x.x \textit{ ist ein Mensch}$

Mengen können generell als Lambda-Terme dargestellt werden.

Lambda-Notation

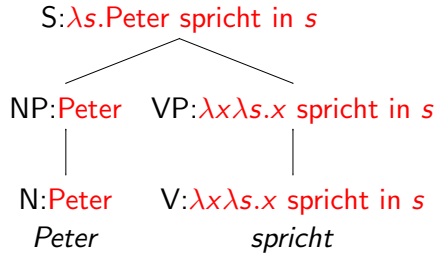
Darstellung von Bedeutungen in Lambda-Notation

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

Kompositionalität

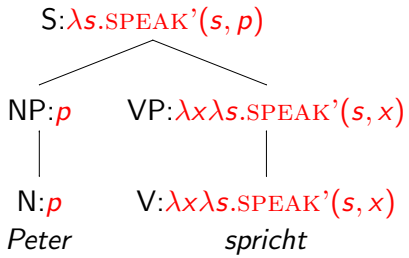
- Satzbedeutung = lexikalische Bedeutungen + Syntax
- Beispiel:
  Peter spricht.
  - Satzbedeutung: $\lambda s.$Peter spricht in s
  - lexikalische Bedeutungen:
    - $\|Peter\|$ = Peter
    - $\|spricht\| = \lambda x \lambda s.x$ spricht in s
  - Syntax: $[_S\ [_{NP}\ [_N$ *Peter* $]\ ]\ [_{VP}\ [_V$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

Kompositionalität



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



Kompositionalität

- 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)

Kompositionalität

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

Kompositionalität

Schönfinkelisierung

- Bedeutung transitiver Verben: **zweistellige Relation**
- z.B.: *lieben* $\rightsquigarrow \{\langle x, y\rangle | \textsc{love'}(x, y)\}$[1]
- Darstellung als charakteristische Funktion:

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

- Lambda-Konversion:

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

[1] Wir ignorieren für den Moment die Situationsabhängigkeit.

Kompositionalität

Schönfinkelisierung

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

$$\|\textit{liebt Hans}\| = \{x | \textsc{love'}(x, h)\} \approx \lambda x.\textsc{love'}(x, h)$$

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

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

Kompositionalität

Schönfinkelisierung

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

$$\lambda y \lambda x.\textsc{love'}(x, y)$$

- generelle Technik:

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

- auch auf mehrstellige Relationen anwendbar:

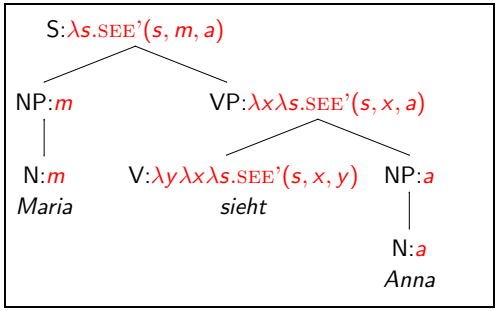
$$\{\langle x_1, \cdots, x_n\rangle | S(x_1, \cdots, x_n)\} \rightsquigarrow \lambda x_n. \cdots .\lambda x_1.S(x_1, \cdots, x_n)$$

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

Transitive Verben

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

Transitive Verben



Regeln:

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

Boolsche Operatoren

Die kompositionale Analyse der Boolschen 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:

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

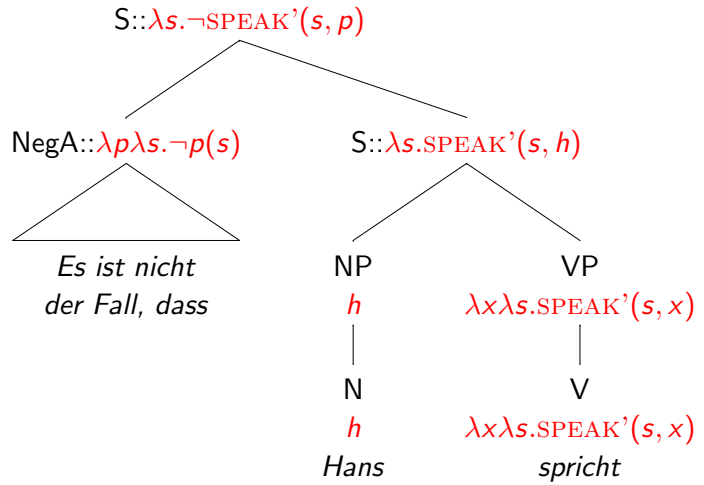
Boolsche Operatoren

Negation

- Neue Regeln:
  - $S_1 \rightarrow NegA, S_2 :: \|S_1\| = \|NegA\|(\|S_2\|)\|$
  - $S_1 \rightarrow S_2, NegI :: \|S_1\| = \|NegI\|(\|S_2\|)\|$
  - $NegA \rightarrow$ *Es ist nicht der Fall, dass* $:: \|NegA\| = \lambda p \lambda s.\neg p(s)$
  - $NegI \rightarrow$ *nicht* $:: \|NegI\| = \lambda p \lambda s.\neg p(s)$

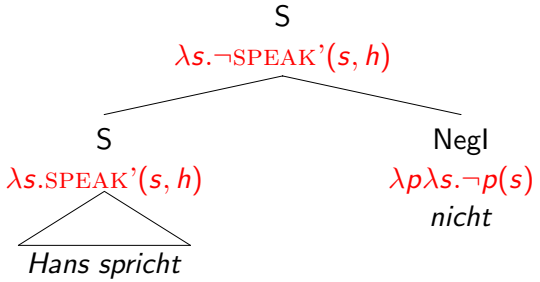
Boolsche Operatoren

Negation



Boolsche Operatoren

Negation



Boolsche Operatoren

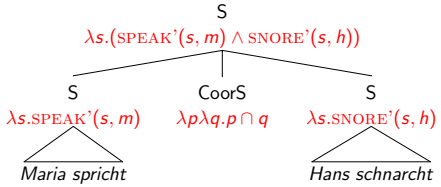
Satz-Koordination

- Regeln:
  - $S_1 \rightarrow S_2, CoorS, S_3 :: \|S_1\| = \|CoorS\|(\|S_2\|)(\|S_3\|)$
  - $CoorS \rightarrow$ *und* $:: \lambda p \lambda q.p \cap q$
  - $CoorS \rightarrow$ *oder* $:: \lambda p \lambda q.p \cup q$
- Merke:

$$\begin{aligned} \lambda s.\phi \cap \lambda s.\psi &= \lambda s.(\phi \wedge \psi) \\ \lambda s.\phi \cup \lambda s.\psi &= \lambda s.(\phi \vee \psi) \end{aligned}$$

Boolsche Operatoren

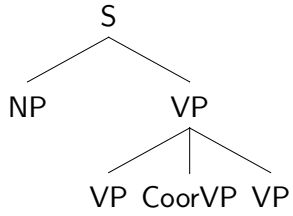
Satz-Koordination



Boolsche Operatoren

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.*

Boolsche Operatoren

VP-Koordination

- Regeln:
  - $VP_1 \rightarrow VP_2, CoorVP, VP_3$ ::
    $\|VP_1\| = \|CoorVP\|(\|VP_2\|)(\|VP_3\|)$
  - $CoorVP \rightarrow$ *und* :: $\lambda P \lambda Q \lambda x \lambda s.P(x)(s) \wedge Q(x)(s)$
  - $CoorVP \rightarrow$ *oder* :: $\lambda P \lambda Q \lambda x \lambda s.P(x)(s) \vee Q(x)(s)$

Boolsche Operatoren

VP-Koordination