Einführung in die Logik - 07

Höherstufige Semantikformalismen: Typenlogik und λ-Kalkül

Stärken von PL-1 als semantischem Repräsentationsformalismus

[Lit.: Pinkal 2000]

- Näherung an angemessene Bedeutungsbeschreibung durch Vergleich von präzise angebbaren errechneten Wahrheitsbedingungen mit intuitiven Wahrheitsurteilen

  Nur Peter ist intelligent.

  $I(p) \wedge \forall x (x \neq p \rightarrow \neg I(x)))$

  Nicht nur Peter ist intelligent.

  $\neg(I(p) \wedge \forall x (x \neq p \rightarrow \neg I(x)))$

  - wahr, wenn niemand im Modell intelligent ist

    $I(p) \wedge \neg\forall x (x \neq p \rightarrow \neg I(x)))$

  - Präsupposition: *Peter ist intelligent*

Stärken von PL-1 als semantischem Repräsentationsformalismus

[Lit.: Pinkal 2000]

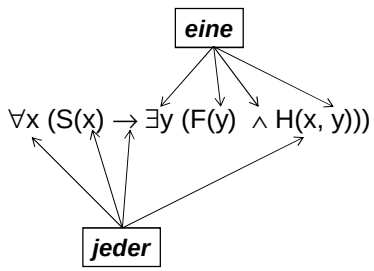
- PL-Deduktionskalkül
  - Inferenzverfahren zur Unterstützung der semantischen Auswertung
- denotationelle Semantik der PL
  - erlaubt Kontrolle des Deduktionsmechanismus bzgl. Korrektheit und Vollständigkeit

Schwächen von PL-1 als semantischem Repräsentationsformalismus

[Lit.: Pinkal 2000]

- keine durchgehend transparente Beziehung zwischen NL-Ausdrücken und Teilausdrücken der semantischen Repräsentation entsprechend lexikalischer Kategorie und syntaktischer Struktur

Jeder Student hat ***eine*** Frage.



Schwächen von PL-1 als semantischem Repräsentationsformalismus

- PL-1 ist als universeller Repräsentationsformalismus für NL-Semantik nicht hinreichend ausdrucksstark:

Dieser Wagen rostet	R(w)
Dieser Wagen rostet *schnell*.	~~R(w) ∧ S(w)~~
Maria ist eine Heidelberger Computerlinguistin.	CL(m) ∧ H(m)
Maria ist eine *begabte* Computerlinguistin (aber als Pianistin unbegabt).	~~CL(m) ∧ B(m)~~

Schwächen von PL-1 als semantischem Repräsentationsformalismus

- PL-1 ist als universeller Repräsentationsformalismus für NL-Semantik nicht hinreichend ausdrucksstark:

Maria ist eine *sehr begabte* Computerlinguistin.	**?????????**
Maria ist *glücklicherweise* an der Uni HD.	**?????????**
Maria arbeitet *gerne* an der Uni HD.	**?????????**
Maria war *gestern* an der Uni HD.	**?????????**

➢ **reichere Repräsentationssprache mit höherstufigen Prädikaten(*) erforderlich**

(*) auch: *Prädikate höherer Ordnung*

Höherstufige Formalismen: Typenlogik

- **reichhaltigere semantische Repräsentationen durch Ausdrücke der Typenlogik**

Typenlogik

– 2 Basistypen:

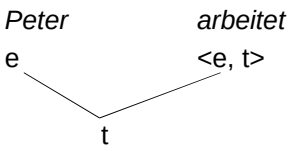
e	-	Individuen der Domäne ('entity')
t	-	Wahrheitswerte ('truth value')

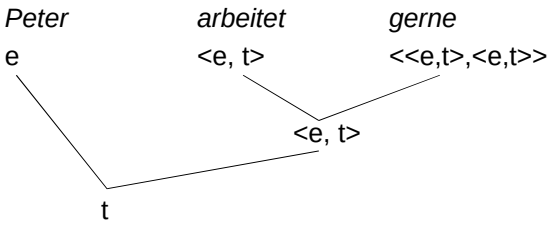
– komplexe Typen als Funktionen $\tau 1 \rightarrow \tau 2$ bzw. $<\tau 1, \tau 2>$

Peter	$\Rightarrow$	e
arbeitet	$\Rightarrow$	<e, t>
kennt	$\Rightarrow$	<e, <e, t>>
gerne	$\Rightarrow$	<<e, t>, <e, t>>

Höherstufige Formalismen: Typenlogik

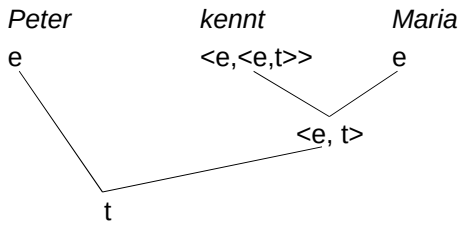
Komposition als funktionale Applikation:





Höherstufige Formalismen: Typenlogik

Komposition als funktionale Applikation:

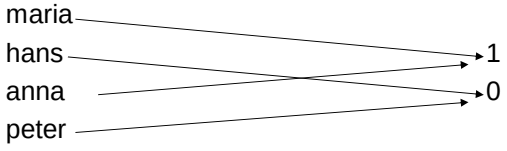


Höherstufige Formalismen: Typenlogik

semantische Typen als Funktionen: Beispiel Prädikate

Frau'(x)

D_e $\Rightarrow$ D_t

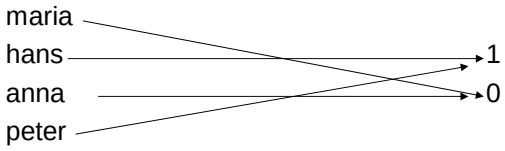


Höherstufige Formalismen: Typenlogik

semantische Typen als Funktionen: Beispiel Prädikate

Mann'(x)

D_e $\Rightarrow$ D_t

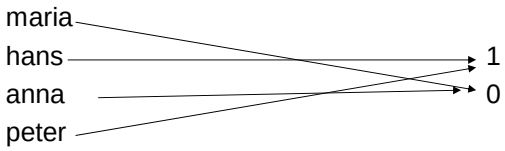


Höherstufige Formalismen: Typenlogik

semantische Typen als Funktionen: Beispiel Prädikate

Mann'(x)

D_e $\Rightarrow$ D_t

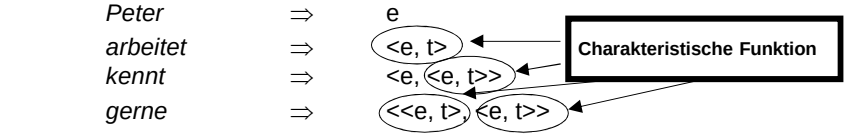


- charakteristische Funktion Mann'(x) ≈ Menge V(Mann')

Höherstufige Formalismen: Typenlogik

Typenlogik

- 2 Basistypen:
  - e - Individuen der Domäne ('entity')
  - t - Wahrheitswerte ('truth value')
- komplexe Typen als Funktionen $\tau 1 \rightarrow \tau 2$ bzw. $<\tau 1, \tau 2>$



Höherstufige Formalismen: Typenlogik

Lexikon und Syntax der Typenlogik

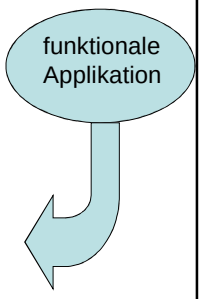
Die wfAs des Typs τ, $ME\tau$, sind wie folgt definiert:

- Jede Variable vom Typ τ ist Element von $ME\tau$
- Jede Konstante vom Typ τ ist Element von $ME\tau$
- Sind $\alpha \in ME(\tau,\sigma)$ und $\beta \in ME\tau$, so ist $\alpha(\beta) \in ME\sigma$
- Sind $\alpha \in ME\tau$ und $\beta \in ME\tau$, so ist $\alpha = \beta \in ME_t$
- Sind $\varphi \in ME_t$ und $\psi \in ME_t$, dann sind auch $\neg\varphi \in ME_t$, $\varphi \wedge \psi \in ME_t$, $\varphi \vee \psi \in ME_t$ und $\varphi \rightarrow \psi \in ME_t$
- Ist $\varphi \in ME_t$ und v eine Variable von beliebigem Typ, so sind auch $\forall v.\varphi \in ME_t$ und $\exists v.\varphi \in ME_t$

Die Ausdrücke aus ME_t heißen *Aussagen* oder *Formeln*.

Höherstufige Formalismen: Typenlogik

Lexikon und Syntax der Typenlogik



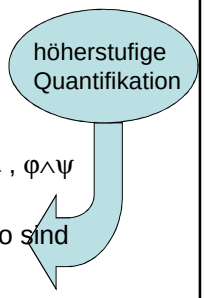
Die wfAs des Typs τ, $ME\tau$, sind wie folgt definiert:

- Jede Variable vom Typ τ ist Element von $ME\tau$
- Jede Konstante vom Typ τ ist Element von $ME\tau$
- Sind $\alpha \in ME(\tau,\sigma)$ und $\beta \in ME\tau$, so ist $\alpha(\beta) \in ME\sigma$
- Sind $\alpha \in ME\tau$ und $\beta \in ME\tau$, so ist $\alpha = \beta \in ME_t$
- Sind $\varphi \in ME_t$ und $\psi \in ME_t$, dann sind auch $\neg\varphi \in ME_t$, $\varphi\wedge\psi \in ME_t$, $\varphi\vee\psi \in ME_t$ und $\varphi\rightarrow\psi \in ME_t$
- Ist $\varphi \in ME_t$ und v eine Variable von beliebigem Typ, so sind auch $\forall v.\varphi \in ME_t$ und $\exists v.\varphi \in ME_t$

Die Ausdrücke aus ME_t heißen *Aussagen* oder *Formeln*.

Höherstufige Formalismen: Typenlogik

Lexikon und Syntax der Typenlogik



Die wfAs des Typs τ, $ME\tau$, sind wie folgt definiert:

- Jede Variable vom Typ τ ist Element von $ME\tau$
- Jede Konstante vom Typ τ ist Element von $ME\tau$
- Sind $\alpha \in ME(\tau,\sigma)$ und $\beta \in ME\tau$, so ist $\alpha(\beta) \in ME\sigma$
- Sind $\alpha \in ME\tau$ und $\beta \in ME\tau$, so ist $\alpha = \beta \in ME_t$
- Sind $\varphi \in ME_t$ und $\psi \in ME_t$, dann sind auch $\neg\varphi \in ME_t$, $\varphi\wedge\psi \in ME_t$, $\varphi\vee\psi \in ME_t$ und $\varphi\rightarrow\psi \in ME_t$
- Ist $\varphi \in ME_t$ und v eine Variable von beliebigem Typ, so sind auch $\forall v.\varphi \in ME_t$ und $\exists v.\varphi \in ME_t$

Die Ausdrücke aus ME_t heißen *Aussagen* oder *Formeln*.

Dr. Michael Herweg, Einführung in die Logik, Univ. Heidelberg

Höherstufige Formalismen: Typenlogik

Semantik der Typenlogik

Die Domäne D_τ des Typs τ ist definiert als:

$D_e = D$

$D_t = \{0, 1\}$

$D_{(\tau,\sigma)}$ *ist die Menge aller Funktionen von* D_τ *nach* D_σ

D.h., wg. Äquivalenz von Mengen mit charakteristischen Funktionen über ihren Elementen:

D_e : *Objekte des Typs e*
► *Denotate von Individuentermen*

$D_{<e,t>}$: *Mengen von Objekten des Typs e*
► *Denotate von Prädikaten*

usw.

Höherstufige Formalismen: Typenlogik

Ein **Modell für die Sprache der Typenlogik L_{TL}** ist
ein geordnetes Paar M = <D, V>, wobei

D ≠ ∅ (Domäne, Individuenbereich, Universum)
V (die Modellfunktion, Wertzuweisungsfunktion)
ist eine Abbildung, die jeder Konstanten vom Typ τ ein Element aus Dτ zuordnet.

Der semantische Wert (die **Interpretation**) eines typenlogischen Ausdrucks A in einem Modell M, $[|A|]^M$, ist definiert durch:

(a) $[|c|]^M$ = V(c) für alle Konstanten c

(b) $[|\alpha(\beta)|]^M = [|\alpha|]^M ([|\beta|]^M)$

- Rest wie PL
- Variablenbelegung g hier fortgelassen; s. PL

Höherstufige Semantikformalismen: Typenlogik und λ-Kalkül

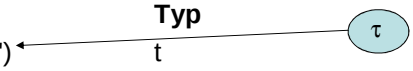
Der λ-Kalkül:

- **λ-Operator** als logische Konstante der TL
  - λ bindet Variablen v eines beliebigen Typs σ in wfAs A eines beliebigen Typs τ
  - der resultierende Ausdruck λvA ist (eine Funktion) vom Typ <σ, τ>
  - die Variablenposition v im Körper A des wfA λvA wird durch das Präfix λv als offene Argumentposition ausgezeichnet
  - durch funktionale Applikation auf einen wfA des Typs σ wird die Argumentstelle v in λvA wieder gebunden
  - lies λvA (intensional) als "ein v (zu sein), so dass A" oder (extensional) als "die Menge aller v, so dass A"

Höherstufige Semantikformalismen: Typenlogik und λ-Kalkül

F(a)	⇒	λxF(x)	λ-Abstraktion
λxF(x)(a)	⇒	F(a)	λ-Konversion, funktionale Applikation (*)

Beispiele:	**wfA**	**Typ**
(1)	Kennt'(hans', maria')	t

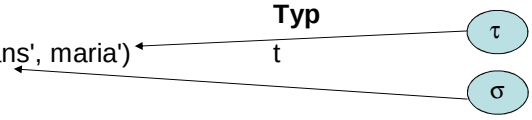


(*) *unter bestimmten Umständen auch: β-Reduktion*

Höherstufige Semantikformalismen: Typenlogik und λ-Kalkül

F(a)	⇒	λxF(x)	λ-Abstraktion
λxF(x)(a)	⇒	F(a)	λ-Konversion, funktionale Applikation

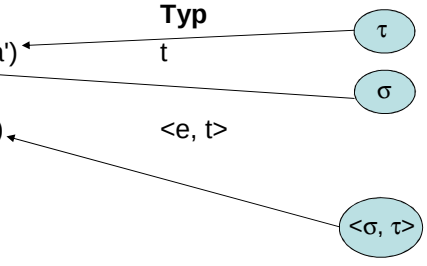
Beispiele:	**wfA**	**Typ**
(1)	Kennt'(hans', maria')	t



Höherstufige Semantikformalismen: Typenlogik und λ-Kalkül

F(a)	⇒	λxF(x)	λ-Abstraktion
λxF(x)(a)	⇒	F(a)	λ-Konversion, funktionale Applikation

Beispiele:	**wfA**	**Typ**
(1)	Kennt'(hans', maria')	t
(2)	λx Kennt'(x, maria')	<e, t>



Dr. Michael Herweg, Einführung in die Logik, Univ. Heidelberg

Höherstufige Semantikformalismen: Typenlogik und λ-Kalkül

F(a)	⇒	λxF(x)	λ-Abstraktion
λxF(x)(a)	⇒	F(a)	λ-Konversion, funktionale Applikation

Beispiele:	**wfA**	**Typ**
(1)	Kennt'(hans', maria')	t
(2)	λx Kennt'(x, maria')	<e, t>

Höherstufige Semantikformalismen: Typenlogik und λ-Kalkül

F(a)	⇒	λxF(x)	λ-Abstraktion
λxF(x)(a)	⇒	F(a)	λ-Konversion, funktionale Applikation

Beispiele:	**wfA**	**Typ**
(1)	Kennt'(hans', maria')	t
(3)	λy Kennt'(hans', y)	<e, t>

Dr. Michael Herweg, Einführung in die Logik, Univ. Heidelberg

Höherstufige Semantikformalismen: Typenlogik und λ-Kalkül

F(a)	⇒	λxF(x)	λ-Abstraktion
λxF(x)(a)	⇒	F(a)	λ-Konversion, funktionale Applikation

Beispiele:	**wfA**	**Typ**
(1)	Kennt'(hans', maria')	t
(4)	λy λx Kennt'(x, y)	<e, e, t> oder <e, <e, t>>

Höherstufige Semantikformalismen: Typenlogik und λ-Kalkül

F(a)	⇒	λxF(x)	λ-Abstraktion
λxF(x)(a)	⇒	F(a)	λ-Konversion, funktionale Applikation

Beispiele:	**wfA**	**Typ**
(1)	Kennt'(hans', maria')	t
(5)	λR R(hans', maria')	<<<e, <e, t>>, t>

Dr. Michael Herweg, Einführung in die Logik, Univ. Heidelberg

Höherstufige Semantikformalismen: Typenlogik und λ-Kalkül

F(a)	⇒	λxF(x)	λ-Abstraktion
λxF(x)(a)	⇒	F(a)	λ-Konversion, funktionale Applikation

Beispiele:	**wfA**	**Typ**
(1)	Kennt'(hans', maria')	t
(6)	λyλR R(hans', y)	<e, <<<e, <e, t>>, t>>

Höherstufige Semantikformalismen: Typenlogik und λ-Kalkül

F(a)	⇒	λxF(x)	λ-Abstraktion
λxF(x)(a)	⇒	F(a)	λ-Konversion, funktionale Applikation

Beispiele:	**wfA**	**Prädikatsextension**
(1)	Kennt'(hans', maria')	
(2)	λx Kennt'(x, maria')	*wer Maria kennt*

Dr. Michael Herweg, Einführung in die Logik, Univ. Heidelberg

Höherstufige Semantikformalismen: Typenlogik und λ-Kalkül

F(a)	⇒	λxF(x)	λ-Abstraktion
λxF(x)(a)	⇒	F(a)	λ-Konversion, funktionale Applikation

Beispiele:	**wfA**	**Prädikatsextension**
(1)	Kennt'(hans', maria')	
(3)	λy Kennt'(hans', y)	*wen Hans kennt*

Höherstufige Semantikformalismen: Typenlogik und λ-Kalkül

F(a)	⇒	λxF(x)	λ-Abstraktion
λxF(x)(a)	⇒	F(a)	λ-Konversion, funktionale Applikation

Beispiele:	**wfA**	**Prädikatsextension**
(1)	Kennt'(hans', maria')	
(4)	λy λx Kennt'(x, y)	*wer wen kennt*

Höherstufige Semantikformalismen: Typenlogik und λ-Kalkül

F(a)	⇒	λxF(x)	λ-Abstraktion
λxF(x)(a)	⇒	F(a)	λ-Konversion, funktionale Applikation

Beispiele:	**wfA**	**Prädikatsextension**
(1)	Kennt'(hans', maria')	
(5)	λR R(hans', maria')	*was zwischen Hans und Maria der Fall ist*

Höherstufige Semantikformalismen: Typenlogik und λ-Kalkül

F(a)	⇒	λxF(x)	λ-Abstraktion
λxF(x)(a)	⇒	F(a)	λ-Konversion, funktionale Applikation

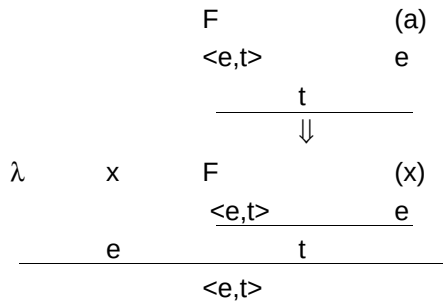
Beispiele:	**wfA**	**Prädikatsextension**
(1)	Kennt'(hans', maria')	
(6)	λyλR R(hans', y)	*was zwischen Hans und wem der Fall ist*

Dr. Michael Herweg, Einführung in die Logik, Univ. Heidelberg

Höherstufige Semantikformalismen: Typenlogik und λ-Kalkül

F(a)	⇒	λxF(x)	λ-Abstraktion
λxF(x)(a)	⇒	F(a)	λ-Konversion, funkt. Appl.

Typbestimmung von λ-Ausdrücken



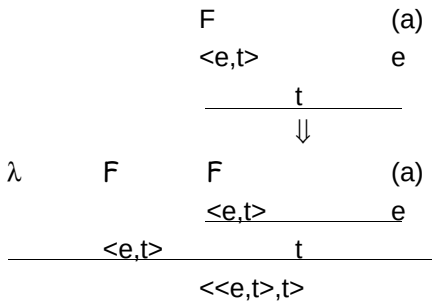
- **λxF(x)** : die Menge aller x mit Eigenschaft F

Dr. Michael Herweg, Einführung in die Logik, Univ. Heidelberg

Höherstufige Semantikformalismen: Typenlogik und λ-Kalkül

F(a)	⇒	λxF(x)	λ-Abstraktion
λxF(x)(a)	⇒	F(a)	λ-Konversion, funkt. Appl.

Typbestimmung von λ-Ausdrücken



- **λF F(a)** : die Menge aller Eigenschaften des Objekts a

Dr. Michael Herweg, Einführung in die Logik, Univ. Heidelberg

Höherstufige Semantikformalismen: Typenlogik und λ-Kalkül

F(a)	⇒	λxF(x)	λ-Abstraktion
λxF(x)(a)	⇒	F(a)	λ-Konversion, funktionale Applikation

Beispiele λ-Konversion

λx Kennt'(x, maria')(hans') ⇒ Kennt'(hans', maria')

Höherstufige Semantikformalismen: Typenlogik und λ-Kalkül

F(a)	⇒	λxF(x)	λ-Abstraktion
λxF(x)(a)	⇒	F(a)	λ-Konversion, funktionale Applikation

Beispiele λ-Konversion

λy Kennt'(hans', y)(maria') ⇒ Kennt'(hans', maria')

Dr. Michael Herweg, Einführung in die Logik, Univ. Heidelberg

Höherstufige Semantikformalismen: Typenlogik und λ-Kalkül

F(a)	⇒	λxF(x)	λ-Abstraktion
λxF(x)(a)	⇒	F(a)	λ-Konversion, funktionale Applikation

Beispiele λ-Konversion

λR R(hans', maria')(Kennt') ⇒ Kennt'(hans', maria')

Höherstufige Semantikformalismen: Typenlogik und λ-Kalkül

F(a)	⇒	λxF(x)	λ-Abstraktion
λxF(x)(a)	⇒	F(a)	λ-Konversion, funktionale Applikation

Beispiele λ-Konversion

λx λy λR R(x, y) (Kennt')(maria')(hans')

⇒ λy λR R(hans', y)) (Kennt')(maria')

⇒ λR R(hans', maria') (Kennt')

⇒ Kennt'(hans', maria')

Dr. Michael Herweg, Einführung in die Logik, Univ. Heidelberg

Höherstufige Semantikformalismen: Typenlogik und λ-Kalkül

Flexibilität bzgl. Funktor/Argument-Struktur durch Typanhebung ("Type Raising")

Funktor	Argument	Resultat
<e, t> Schläft'	e peter'	t Schläft'(peter')
<<e, t>, t> peter'	<e, t> Schläft'	t Schläft'(peter')
<<<e, t>, t>, t> Schläft'	<<e, t>, t> peter'	t Schläft'(peter')

Höherstufige Semantikformalismen: Typenlogik und λ-Kalkül

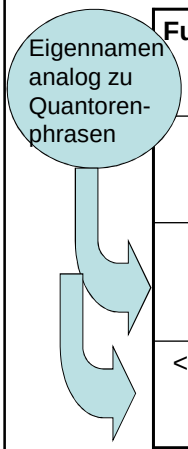
Flexibilität bzgl. Funktor/Argument-Struktur durch Typanhebung ("Type Raising")

Funktor	Argument	Resultat
<e, t> Schläft'	e peter'	t Schläft'(peter')
<<e, t>, t> peter'	<e, t> Schläft'	t Schläft'(peter')
<<<e, t>, t>, t> Schläft'	<<e, t>, t> peter'	t Schläft'(peter')

Dr. Michael Herweg, Einführung in die Logik, Univ. Heidelberg

Höherstufige Semantikformalismen: Typenlogik und λ-Kalkül

Flexibilität bzgl. Funktor/Argument-Struktur durch Typanhebung ("Type Raising")

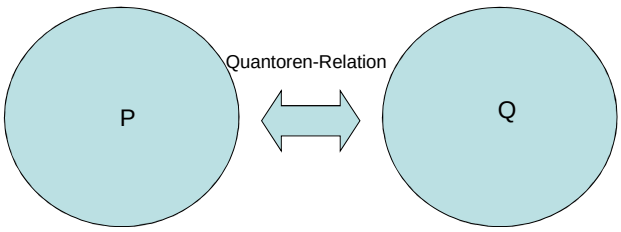


Funktor	Argument	Resultat
<e, t> Schläft'	e peter'	t Schläft'(peter')
<<e, t>, t> peter'	<e, t> Schläft'	t Schläft'(peter')
<<<e, t>, t>, t> Schläft'	<<e, t>, t> peter'	t Schläft'(peter')

Höherstufige Semantikformalismen: Typenlogik und λ-Kalkül

Quantoren + Determinatoren

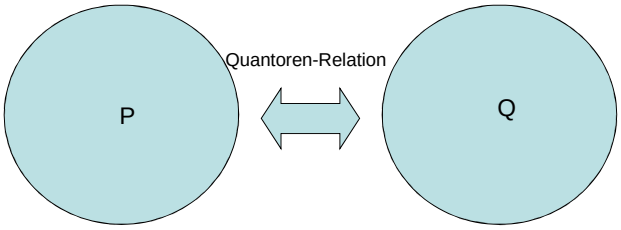
jede(r)	<<e, t>, <<e, t>, t>>	λP λQ ... (P(x) ... Q(x))
ein(e)	<<e, t>, <<e, t>, t>>	λP λQ ... (P(x) ... Q(x))
kein(e)	<<e, t>, <<e, t>, t>>	λP λQ ... (P(x) ... Q(x))
der/die/das	<<e, t>, <<e, t>, t>>	λP λQ ... (P(x) ... Q(x))



Höherstufige Semantikformalismen: Typenlogik und λ-Kalkül

Quantoren + Determinatoren

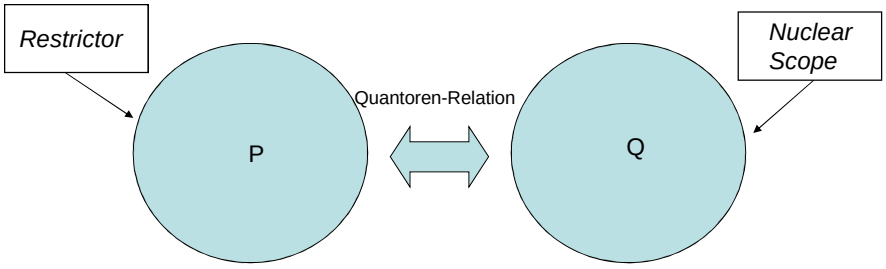
jede(r)	<<e, t>, <<e, t>, t>>	λP λQ **∀x** (P(x) → Q(x))
ein(e)	<<e, t>, <<e, t>, t>>	λP λQ **∃x** (P(x) ∧ Q(x))
kein(e)	<<e, t>, <<e, t>, t>>	λP λQ **¬∃x** (P(x) ∧ Q(x))
der/die/das	<<e, t>, <<e, t>, t>>	λP λQ **ιx** (P(x) ∧ Q(x)) bzw. λP λQ ∃x (∀y (P(y) ↔ y=x) ∧ Q(x))



Höherstufige Semantikformalismen: Typenlogik und λ-Kalkül

Quantoren + Determinatoren

jede(r)	<<e, t>, <<e, t>, t>>	λP λQ **∀x** (P(x) → Q(x))
ein(e)	<<e, t>, <<e, t>, t>>	λP λQ **∃x** (P(x) ∧ Q(x))
kein(e)	<<e, t>, <<e, t>, t>>	λP λQ **¬∃x** (P(x) ∧ Q(x))
der/die/das	<<e, t>, <<e, t>, t>>	λP λQ **ιx** (P(x) ∧ Q(x)) bzw. λP λQ ∃x (∀y (P(y) ↔ y=x) ∧ Q(x))



Dr. Michael Herweg, Einführung in die Logik, Univ. Heidelberg

Was spricht für eine einheitliche Typisierung von Quantorenphrasen (QP) und Nominalphrasen (NP) bzw. Determinatorenphrasen (DP) als <<e, t>, t>

- Alle Studierenden …
- Einige Studierende …
- Hans und Maria …
- Die Teilnehmerinnen und Teilnehmer der Logik-Vorlesung …
- Hans …
- Der Logik-Dozent …
- Ein Romanist …
- Kein Romanist …

… kennen/kennt das Theoretikum.

Aber (!): Es gibt auch gute – v.a. diskurssemantische - Gründe gegen eine Gleichbehandlung von QP und NP/DP.

Semantikkonstruktion mit Ausdrücken des typisierten λ-Kalküls
$[_S\ [_{PN}$ Hans$]\ [_{VP}\ [_V$ kennt$]\ [_{PN}$ Maria$]\ ]\ ]$
vgl. Montague-Semantik (nach Richard Montague 1970)

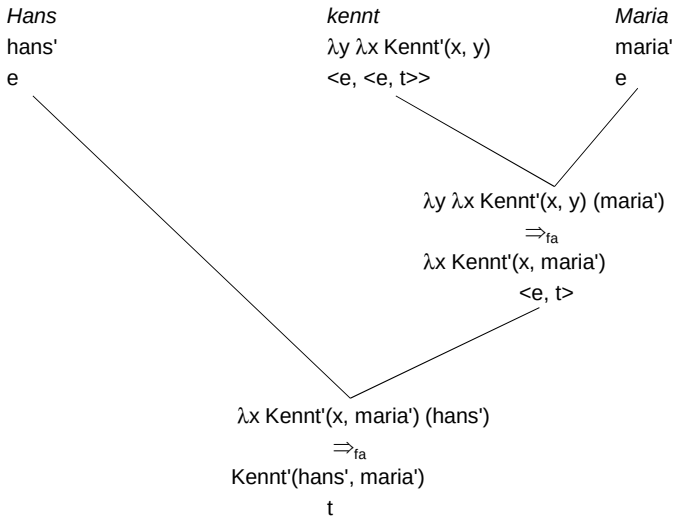
Lexem	synt. Kat.	sem. Typ	sem. Repr.
Hans	PN	e	hans'
kennt	V_{tr}	<e, <e, t>>	λy λx Kennt'(x, y)
Maria	PN	e	maria'

$[_V$ kennt$]$		λy λx Kennt'(x, y)
$[_{VP}\ [_V$ kennt$]\ [_{PN}$ Maria$]\ ]$		λy λx Kennt'(x, y) (maria')
	$\Rightarrow_{fa}$	λx Kennt'(x, maria')
$[_S\ [_{PN}$ Hans$]\ [_{VP}\ [_V$ kennt$]\ [_{PN}$ Maria$]\ ]\ ]$		λx Kennt'(x, maria') (hans')
	$\Rightarrow_{fa}$	Kennt'(hans', maria')

Semantikkonstruktion mit Ausdrücken des typisierten λ-Kalküls

[$_S$ [$_{PN}$ Hans] [$_{VP}$ [$_V$ kennt] [$_{PN}$ Maria]]]

vgl. Montague-Semantik (nach Richard Montague 1970)



Semantikkonstruktion mit Ausdrücken des typisierten λ-Kalküls

[$_S$ [$_{NP}$ Peter] [$_{VP}$ [$_V$ arbeitet] [$_{ADV}$ gerne]]]

vgl. Montague-Semantik (nach Richard Montague 1970)

Lexem	synt. Kat.		sem. Typ	sem. Repr.
arbeitet	V_{itr}		<e, t>	λx A(x)
gerne	ADV		<<*e, t*>, <e, t>>	λP λz (G(P(z)))
Peter	NP		e	p
		bzw.	<<e, t>, t>	λQ (Q(p))

[$_V$ arbeitet]		λx A(x)
[$_{VP}$ [$_V$ arbeitet] [$_{ADV}$ gerne]]		λP λz (G(P(z))) (λx A(x))
	⇒fa	λz (G(λx A(x)(z)))
	⇒fa	λz (G(A(z)))
[$_S$ [$_{NP}$ Peter]] [$_{VP}$ [$_V$ arbeitet] [$_{ADV}$ gerne]]]		
		λz (G(A(z))) (p)
	⇒fa	G(A(p))
bzw.		
[$_S$ [$_{NP}$ Peter]] [$_{VP}$ [$_V$ arbeitet] [$_{ADV}$ gerne]]]		
		λQ (Q(p)) λz (G(A(z)))
	⇒fa	λz (G(A(z))) (p)
	⇒fa	G(A(p))

Semantikkonstruktion mit Ausdrücken des typisierten λ-Kalküls
$[_S\ [_{NP}\ [_Q$ ein$]\ [_N$ Student$]\ ]\ [_{VP}\ [_V$ kennt$]\ [_{PN}$ Maria$]\ ]\ ]$
vgl. Montague-Semantik (nach Richard Montague 1970)

Lexem	synt. Kat.	sem. Typ	sem. Repr.
ein	Q	<<*e, t*>, <<e, t>, t>>	$\lambda P\ \lambda Q\ \exists z\ (P(z) \wedge Q(z))$
Student	N	<e, t>	$\lambda w\ (S(w) \wedge M(w))$
kennt	V_{tr}	<e, <e, t>>	$\lambda y\ \lambda x\ K(x, y)$
Maria	PN	e	m

$[_V$ kennt$]$		$\lambda y\ \lambda x\ K(x, y)$
$[_{VP}\ [_V$ kennt$]\ [_{PN}$ Maria$]\ ]$		$\lambda y\ \lambda x\ K(x, y)\ (m)$
	$\Rightarrow_{fa}$	$\lambda x\ K(x, m)$
$[_Q$ ein$]$		$\lambda P\ \lambda Q\ \exists z\ (P(z) \wedge Q(z))$
$[_{NP}\ [_Q$ ein$]\ [_N$ Student$]\ ]$		$\lambda P\ \lambda Q\ \exists z\ (P(z) \wedge Q(z))\ (\lambda w\ (S(w) \wedge M(w)))$
	$\Rightarrow_{fa}$	$\lambda Q\ \exists z\ (\lambda w(S(w) \wedge M(w))\ (z) \wedge Q(z))$
	$\Rightarrow_{fa}$	$\lambda Q\ \exists z\ (S(z) \wedge M(z) \wedge Q(z))$
$[_S\ [_{NP}\ [_Q$ ein$]\ [_N$ Student$]\ ]\ [_{VP}\ [_V$ kennt$]\ [_{PN}$ Maria$]\ ]\ ]$		$\lambda Q\ \exists z\ (S(z) \wedge M(z) \wedge Q(z))\ (\lambda x\ K(x, m))$
	$\Rightarrow_{fa}$	$\exists z\ (S(z) \wedge M(z) \wedge \lambda x\ K(x, m)\ (z))$
	$\Rightarrow_{fa}$	$\exists z\ (S(z) \wedge M(z) \wedge K(z, m))$

Dr. Michael Herweg, Einführung in die Logik, Univ. Heidelberg

Höherstufige Semantikformalismen: Typenlogik und λ-Kalkül

Stärken des Verfahrens:

- grammatische Einheiten sind als eigenständige logisch-semantische Konstrukte repräsentierbar
- gleichartige grammatische Elemente werden durch strukturgleiche logisch-semantische Repräsentationen dargestellt
- kombinatorische Eigenschaften von Wörtern und Phrasen sind direkt in deren logisch-semantischer Repräsentation kodiert

wichtige Eigenschaft des Verfahrens:

- Repräsentationen mehrdeutiger Ausdrücke werden durch alternative Semantikkonstruktionen erzeugt
  - Weiterentwicklung: unterspezifizierte Repräsentationen

Dr. Michael Herweg, Einführung in die Logik, Univ. Heidelberg

Recap Typentheorie & λ-Kalkül
Prof. Dr. Anette Frank
Formale Semantik, WS 2014/15

Literatur

- L.T.F. Gamut (1991): Logic, Language and Meaning, Vol II. University of Chicago Press. Chapter 4

Dr. Michael Herweg, Einführung in die Logik, Univ. Heidelberg

Grenzen der Logik erster Ordnung

(1) Franz ist ein Pianist
- Prädikat erster Ordnung: pianist(f*)

(2) Franz ist ein guter Pianist
- Eben nicht äquivalent zu: Franz ist gut und Franz ist Pianist
- Prädikat zweiter Ordnung: gut(pianist)(f*)

(3) Franz ist ein sehr guter Pianist
- Prädikat dritter Ordnung: sehr(gut)(pianist)(f*)

- Allgemeines linguistisches Phänomen: **Modifikation**

Dr. Michael Herweg, Einführung in die Logik, Univ. Heidelberg

Recap Typentheorie & λ-Kalkül aus:
Anette Frank, Formale Semantik, WS 2014/15

Grenzen der Logik erster Ordnung

(1) Peter ist blond
- blond(p*)

(2) Blond ist eine Haarfarbe
- haarfarbe(blond)

(3) Franz und Maria haben dieselbe Haarfarbe
- ∃G.haarfarbe(G) ∧ G(f*) ∧ G(m*)

- Quantifikation über Prädikate höherer Ordnung

Dr. Michael Herweg, Einführung in die Logik, Univ. Heidelberg

Recap Typentheorie & λ-Kalkül aus:
Anette Frank, Formale Semantik, WS 2014/15

Typtheorie

- Die Typen der nichtlogischen Ausdrücke, die von Logik erster Ordnung bereitgestellt werden, reichen nicht aus, um die semantische Funktion alle natürlichsprachlicher Ausdrücke zu beschreiben
- Typtheorie stellt eine **reichhaltigere Menge von Typen** zur Verfügung: Relationen und Funktionen höherer Ordnung
- Intuition: "Erweiterte Version von Prädikatenlogik"
  - Reichere Menge von Variablen, Konstanten, Funktionen
  - Strukturiert durch **Typsystem**

Dr. Michael Herweg, Einführung in die Logik, Univ. Heidelberg

Recap Typentheorie & λ-Kalkül aus:
Anette Frank, Formale Semantik, WS 2014/15

Typen für Ausdrücke [noch in PL1]

- Individuenkonstanten, Variablen (Eigennamen)
  - e
- Wahrheitswerte (Sätze: "es regnet")
  - t
- Einstellige Prädikate (schlafen, gehen, …)
  - ⟨e, t⟩
- Zweistellige Prädikate (ansehen, bewundern, …)
  - ⟨e, ⟨e, t⟩⟩
- Dreistellige Prädikate (geben, helfen (mit), …)
  - ⟨e, ⟨e, ⟨e, t⟩⟩⟩

Dr. Michael Herweg, Einführung in die Logik, Univ. Heidelberg

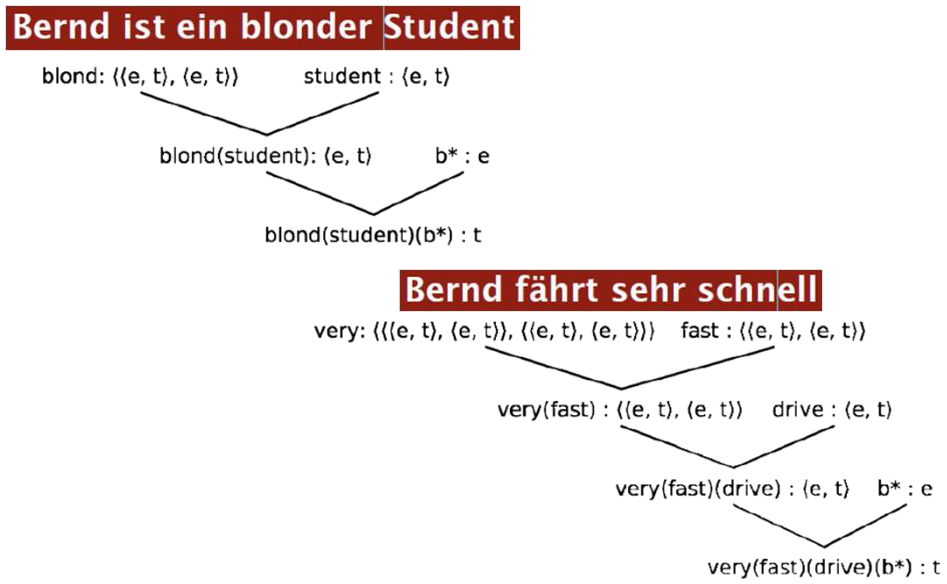
Recap Typentheorie & λ-Kalkül aus:
Anette Frank, Formale Semantik, WS 2014/15

Neue Typen in der Typtheorie

- Satzadverbien – gestern, leider:
  - Ich nehme einen Satz und liefere einen Satz zurück
  - ⟨t, t⟩
- Attributive Adjektive – verheiratet, arm:
  - Ich modifiziere eine Eigenschaft (=das Nomen)
  - ⟨⟨e,t⟩, ⟨e,t⟩⟩
- Gradmodifikatoren – sehr, relativ:
  - Ich modifiziere ein Adjektiv
  - ⟨⟨⟨e,t⟩,⟨e,t⟩⟩, ⟨⟨e,t⟩,⟨e,t⟩⟩⟩

Dr. Michael Herweg, Einführung in die Logik, Univ. Heidelberg

Recap Typentheorie & λ-Kalkül aus:
Anette Frank, Formale Semantik, WS 2014/15



Dr. Michael Herweg, Einführung in die Logik, Univ. Heidelberg

Recap Typentheorie & λ-Kalkül aus:
Anette Frank, Formale Semantik, WS 2014/15

Semantik der Typentheorie

Typen und ihre Denotate

Typ	Denotation
e	Individuum (entity)
t	Wahrheitswert (WW)
⟨e,t⟩	Funktion von Individuen nach WW, d.h. (charakteristische Funktion einer) Menge von Individuen
⟨t,t⟩	Funktion von WW nach WW
⟨e,e⟩	Funktion von Individuen nach Individuen
⟨⟨e,t⟩, ⟨e,t⟩⟩	Funktion von Mengen von Individuen nach Mengen von Individuen
⟨e, ⟨e,t⟩⟩	Funktion von Individuen nach Mengen von Individuen
…	…

Dr. Michael Herweg, Einführung in die Logik, Univ. Heidelberg

Recap Typentheorie & λ-Kalkül aus:
Anette Frank, Formale Semantik, WS 2014/15

Interpretation von NPs, 1. Versuch

- Wir weisen quantifizierten NPs eine **atomare semantische Interpretation** zu, die zur korrekten globalen Analyse für den gesamten Satz führt – analog zu Eigennamen
- Bill works
  - Bill ↦ b* : e
  - Bill works ↦ work'(b*) : t
- Every student works
  - Every student ↦ every-student' : e
  - Every student works ↦ work'(every-student') : t
- **Nicht zufriedenstellend**

Dr. Michael Herweg, Einführung in die Logik, Univ. Heidelberg

Recap Typentheorie & λ-Kalkül aus:
Anette Frank, Formale Semantik, WS 2014/15

Interpretation von NPs, 2. Versuch

- **Idee:** NP denotiert eine **Menge von Individuen** für die bestimmte **Eigenschaften** gelten – ggf. die des Satzprädikats
- Every student works
  - Every student ↦ every-student' : ⟨⟨**e, t**⟩, **t**⟩
  - Every student works ↦ every-student'(work') : t
- every-student' ist ein Prädikat zweiter Ordnung
  - Denotiert die Menge der Eigenschaften *aller* Studenten
  - Formal: Für alle $A \in D_{\langle e, t \rangle}$ (d.h. $A \subseteq U_M$)
    - $[\![\text{every-student'}]\!]^{M,g}(A) = 1$ gdw $[\![\text{student'}]\!]^{M,g} \subseteq A$
- a-student': Eigenschaften eines Students
  most-students': Eigenschaften der meisten Studenten, …

Dr. Michael Herweg, Einführung in die Logik, Univ. Heidelberg

Recap Typentheorie & λ-Kalkül aus:
Anette Frank, Formale Semantik, WS 2014/15

Interpretation von Artikeln

- Artikel bezeichnen Funktionen von Eigenschaften ("student") zu Mengen von Eigenschaften
- Every student works
  - every ↦ every' : ⟨⟨e,t⟩, ⟨⟨e, t⟩, t⟩⟩
  - student ↦ student' : ⟨e,t⟩
  - every student ↦ every'(student') : ⟨⟨e, t⟩, t⟩
- Alternative Interpretation:
  - Funktion von einem Prädikat 1. Ordnung ("student") zu einer Menge von einem Prädikat 1. Ordnung ("Eigenschaften von Studenten")

Dr. Michael Herweg, Einführung in die Logik, Univ. Heidelberg

Recap Typentheorie & λ-Kalkül aus:
Anette Frank, Formale Semantik, WS 2014/15

Interpretation von Artikeln

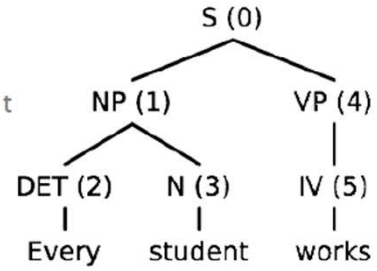
- Für alle A, B ⊆ U_M und Q vom Typ : ⟨⟨e,t⟩, ⟨⟨e, t⟩, t⟩⟩
  - $[\![every']\!]^{M,g}(A)$ =
    die Funktion f ∈ $D_{\langle\langle e,t\rangle,t\rangle}$ sodass für alle B ∈ $D_{\langle e,t\rangle}$ f(B) = 1 gdw A ⊆ B
  - $[\![every']\!]^{M,g}(A)(B)$ = 1 gdw A ⊆ B
- Andere Artikel:
  - $[\![every']\!]^{M,g}(A)(B)$ = 1 iff A ⊆ B — Every student sleeps.
  - $[\![a']\!]^{M,g}(A)(B)$ = 1 iff A ∩ B ≠ ∅ — A student sleeps.
  - $[\![some']\!]^{M,g}(A)(B)$ = 1 iff A ∩ B ≠ ∅ — Some student sleeps.
  - $[\![no']\!]^{M,g}(A)(B)$ = 1 iff A ∩ B = ∅ — No student sleeps.

Dr. Michael Herweg, Einführung in die Logik, Univ. Heidelberg

Recap Typentheorie & λ-Kalkül aus:
Anette Frank, Formale Semantik, WS 2014/15

Every student works

- (2) ↦ every' : ⟨⟨e, t⟩, ⟨⟨e, t⟩, t⟩
- (3) ↦ student' : ⟨e, t⟩
- (1) ↦ every'(student') : ⟨⟨e, t⟩, t⟩
- (5) ↦ work' : ⟨e, t⟩
- (4) ↦ work' : ⟨e, t⟩
- (0) ↦ every'(student')(work') : t



Dr. Michael Herweg, Einführung in die Logik, Univ. Heidelberg

Recap Typentheorie & λ-Kalkül aus:
Anette Frank, Formale Semantik, WS 2014/15

Ein neuer Operator: λ

- λ–Abstraktion ist eine Operation, die einen Ausdruck nimmt und eine bestimmte Argumentposition "öffnet"
  - Explizite Schreibweise für Funktion (Menge)
  - λx (fahren'(x))
    - Denotation: Die Menge der Individuen, die fahren
      $[\![\lambda x \text{ (fahren'(x))}]\!]^{M,g} = [\![\text{ fahren' }]\!]^{M,g}$
  - λx (fahren'(x) ∧ telefonieren'(x))
    - Denotation: Die Menge der Individuen, die fahren und telefonieren
    - Lässt sich nicht äquivalent durch "atomares" Prädikat ausdrücken!

Dr. Michael Herweg, Einführung in die Logik, Univ. Heidelberg

Recap Typentheorie & λ-Kalkül aus:
Anette Frank, Formale Semantik, WS 2014/15

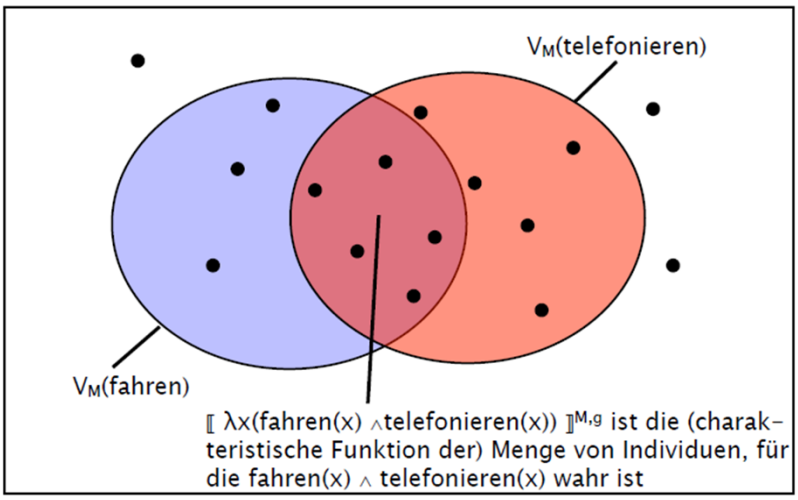
Typtheorie mit dem λ-Operator

- Wir fügen einen neuen Fall zur Definition wohlgeformter Ausdrücke hinzu:
- **Sei α in WE_τ und v eine Variable vom Typ σ. Dann ist λvα ein wohlgeformter Ausdruck vom Typ ⟨σ, τ⟩.**
- Der Skopus des λ-Operators ist der **kleinste** wohlgeformte Ausdruck zu seiner Rechten. Weiterer Skopus muss durch Klammerung angezeigt werden.
- Die "Punktnotation" **λx.** … wird häufig verwenden, um anzuzeigen, dass der λ-Operator den weitestmöglichen Skopus nimmt.

Dr. Michael Herweg, Einführung in die Logik, Univ. Heidelberg

Recap Typentheorie & λ-Kalkül aus:
Anette Frank, Formale Semantik, WS 2014/15

λx(fahren(x) ∧ telefonieren(x))



Dr. Michael Herweg, Einführung in die Logik, Univ. Heidelberg

Recap Typentheorie & λ-Kalkül aus:
Anette Frank, Formale Semantik, WS 2014/15

λ-Abstraktion: Semantik

- Wenn $\alpha \in WE_\tau$, $v \in VAR_\sigma$, dann ist $[\![\lambda v \alpha]\!]^{M,g}$ die Funktion $f : D_\sigma \rightarrow D_\tau$ sodass für alle $a \in D_\sigma$, $f(a) = [\![\alpha]\!]^{M,g[v/a]}$
- **Korrolar**: Wenn der λ-Ausdruck auf ein passendes Argument angewendet wird, können wir die Interpretation vereinfachen: $[\![\lambda v \alpha(\beta)]\!]^{M,g} = [\![\alpha]\!]^{M,g[v/[\![\beta]\!]M,g]}$
  - Alle Vorkommen der λ-abstrahierten Variable in α werden durch die Interpretation von β substituiert.
  - Wohlgetypt gdw es gibt τ sodass $v \in WE_\tau$ und $\beta \in WE_\tau$
  - $[\![\lambda x(\text{fahren}(x) \wedge \text{telefonieren}(x))(b^*)]\!]^{M,g} = 1$
    - iff $[\![\lambda x(\text{fahren}(x) \wedge \text{tel}(x))]\!]^{M,g}([\![b^*]\!]^{M,g}) = 1$
    - iff $[\![\text{fahren}(x) \wedge \text{tel}(x)]\!]^{M,g[x/[\![b^*]\!]M,g} = 1$
    - iff $[\![\text{fahren}(x)]\!]^{M,g[x/[\![b^*]\!]M,g}$ and $[\![\text{tel}(x)]\!]^{M,g[x/[\![b^*]\!]M,g} = 1$
    - iff $V_M(\text{fahren})(V_M(b^*)) = 1$ and $V_M(\text{tel})(V_M(b^*)) = 1$

Dr. Michael Herweg, Einführung in die Logik, Univ. Heidelberg

Recap Typentheorie & λ-Kalkül aus:
Anette Frank, Formale Semantik, WS 2014/15

λ-Operator: Semantik (II)

- Wenn $\alpha \in WE_\tau$, $v \in VAR_\sigma$, dann ist $[\![\lambda v \alpha]\!]^{M,g}$ die Funktion $f : D_\sigma \rightarrow D_\tau$ so dass für alle $a \in D_\sigma$, $f(a) = [\![\alpha]\!]^{M,g[v/a]}$

  Dh. $f \in D_{\langle \sigma, \tau \rangle}$ und der Typ $\langle \sigma, \tau \rangle$ von $\lambda v \alpha$ ist gleich dem Typ von α
- **λ-Konversion**: $[[\lambda v \alpha(\beta)]]^{M,g} = [[\alpha]]^{M,g[v/[[\beta]]M,g]}$
- Durch die Modifikation der Variablenbelegung wird der Wert des Arguments des λ-Ausdrucks in α ersetzt und wird zum Wert aller durch den λ-Operator gebundenen Variablen.
- Dies ist äquivalent zur **Substitution** der λ-gebundenen Variablen v in $\lambda v \alpha(\beta)$ **durch das Argument** β, und nachfolgende **Interpretation** des resultierenden Ausdrucks.
- **β-Reduktion**: $[[\lambda v \alpha(\beta)]]^{M,g} = [[\ [\beta/v]\alpha\ ]]^{M,g}$
  ($[\beta/v]\alpha$: ersetze v durch β in α)

Dr. Michael Herweg, Einführung in die Logik, Univ. Heidelberg

Recap Typentheorie & λ-Kalkül aus:
Anette Frank, Formale Semantik, WS 2014/15

λ-Konversion und β-Reduktion

Sind λvα(β) und [β/v]α immer äquivalent?

- λx (drink(x)∧ drive(x))(john) $\Rightarrow_\beta$ drink(john) ∧ drive(john)
  λx (drink(x)∧ drive(x))(y) $\Rightarrow_\beta$ drink(y) ∧ drive(y)
  λx (∀y know(x)(y))(john) $\Rightarrow_\beta$ ∀y know(john)(y)
  λx (∀y know(x)(y))(y) $\neq_\beta$ ∀y know(y)(y)
- **Bedingung:**
  λvα(v') und [v'/v]α sind äquivalent, falls *v' in α für v frei* ist.
- Eine Variable *v' ist frei für v in einem Ausdruck α* gdw. sich *kein freies Vorkommen von v in α* im Skopus eines Quantors ∃v', ∀v' oder λv' befindet.

Dr. Michael Herweg, Einführung in die Logik, Univ. Heidelberg

Recap Typentheorie & λ-Kalkül aus:
Anette Frank, Formale Semantik, WS 2014/15

β-Reduktion

- λx (∀y know(x)(y))(john)
- *john ist frei für x in ∀y know(x)(y)*:
  es gibt kein freies Vorkommen von x das sich im Skopus eines Quantors über x befindet

  λx (∀y know(x)(y))(john) = [john/x] ∀y know(x)(y)

  $\Rightarrow_\beta$ ∀y know(john)(y)
- λx (∀y know(x)(y))(y)

  y ist nicht frei für x in ∀y know(x)(y)

  => Umbenennung von Variablen so dass y frei für x

  λx (∀z know(x)(z))(y)

Dr. Michael Herweg, Einführung in die Logik, Univ. Heidelberg

Recap Typentheorie & λ-Kalkül aus:
Anette Frank, Formale Semantik, WS 2014/15

Konversionsregeln des λ-Kalküls

- **β-Reduktion / λ-Konversion:**

  $\lambda v\alpha(\beta) \Leftrightarrow [\beta/v]\alpha$

  wenn all freien Variablen in β für v in α frei sind.

- **α-Konversion:**

  Umbenennung der abstrahierten Variablen v nach v'

  $\lambda v\alpha \Leftrightarrow \lambda v'[v'/v]\alpha$

  wenn v' in α frei ist für v.

- **η-Reduktion**

  $\lambda v\alpha(v) \Leftrightarrow \alpha$

Dr. Michael Herweg, Einführung in die Logik, Univ. Heidelberg

Recap Typentheorie & λ-Kalkül aus:
Anette Frank, Formale Semantik, WS 2014/15

Lambda-Terme und Lexikoneinträge

- $\text{schläft}_{\langle e,t\rangle} \leftrightarrow \lambda x_e.\ \text{schläft}_{\langle e,t\rangle}(x)$
- $\text{gefährlich}_{\langle\langle e,t\rangle,\ t\rangle} \leftrightarrow \lambda P_{\langle e,t\rangle}.\ \text{gefährlich}_{\langle\langle e,t\rangle,\ t\rangle}(P)$
- $\text{bill}_{\langle\langle e,t\rangle,\ t\rangle} \leftrightarrow \lambda P_{\langle e,t\rangle}.\ P_{\langle\langle e,t\rangle,\ t\rangle}(\text{bill})$

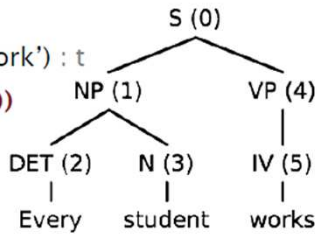
- Unsere bisherige Verwendung von "nackten" Prädikatskonstanten kann also als eta-reduzierte Variante von Lambda-Termen verstanden werden!

Dr. Michael Herweg, Einführung in die Logik, Univ. Heidelberg

Recap Typentheorie & λ-Kalkül aus:
Anette Frank, Formale Semantik, WS 2014/15

Every student works

- (2) ↦ λPλQ∀x(P(x) → Q(x)) : ⟨⟨e, t⟩, ⟨⟨e, t⟩, t⟩⟩
- (3) ↦ student' : ⟨e, t⟩
- (1) ↦ λPλQ∀x(P(x) → Q(x))(student'): ⟨⟨e, t⟩, t⟩
  ⇒β λQ∀x(student'(x) → Q(x))
- (5) = (6) ↦ work' : ⟨e, t⟩
- (0) ↦ λQ∀x(student'(x) → Q(x))(work') : t
  ⇒β **∀x(student'(x) → work'(x))**



Dr. Michael Herweg, Einführung in die Logik, Univ. Heidelberg

Recap Typentheorie & λ-Kalkül aus:
Anette Frank, Formale Semantik, WS 2014/15

Uniforme Repräsentation für NPs

Alle NPs sind vom Typ ⟨⟨e,t⟩,t⟩:
Ausdrücke, die Mengen von Eigenschaften denotieren.

- [[John]]: die Menge der Eigenschaften P, so dass John die Eigenschaft P hat (= Menge der Eigensch. P die John hat)

  λP(P(john))

- [[every student]]: die Menge der Eigenschaften P, so dass jeder Student die Eigenschaft P hat

  λP(∀x(student(x) → P(x)))

- „a student": λP(∃x(student(x) ∧ P(x)))
- „John and Mary": λP(P(john) ∧ P(mary))

Dr. Michael Herweg, Einführung in die Logik, Univ. Heidelberg