Typentheorie der Ambiguität

Christian Wurm (Düsseldorf)

Düsseldorf, 2023

Das Problem der Polysemie

Das Problem ist: es gibt Fälle von Ambiguität, wo wir dieses Kriterium der Konvexität (scheinbar?) verletzen. Diese Fälle fasst man normalerweise unter dem Namen **Polysemie** zusammen.

- Blatt
- Buch
- Abendessen
- Ehe
- ...

Heute betrachten wir dieses Problem etwas genauer.

Das Problem der Polysemie

Wir haben gesagt, Polysemie ist ein Grenzfall der Ambiguität:

Polysemie (denotationell)

Ein Wort hat eine polyseme Bedeutung, falls es zwei kategorial deutlich verschiedene Bedeutungen hat, die aber ebenso deutlich miteinander verknüpft sind.

Diese Definition ist üblich und denotationell. Man könnte ebensogut (und vielleicht besser) eine kombinatorische Definition geben:

Polysemie (kombinatorisch)

Ein Wort hat eine polyseme Bedeutung, falls es zwei deutlich verschiedene Bedeutungen mit denselben syntaktischen Eigenschaften hat, die aber in vielen/fast allen semantischen Kontexten nur eine Lesart zulassen.

Das Problem der Polysemie

Beide Definitionen sind problematisch:

- Die erst ist soz. in sich selbst widersprüchlich
- die zweite ist unzureichend, da es auch bei Polysemie viele Kontexte gibt, die nicht disambiguieren, und bei "normaler" Ambiguität viele Kontexte, die disambiguieren.

Dennoch sind beide interessant, denn bei der Darstellung polysemer Bedeutungen gibt einen breiten Konsens, dass es

1. Eine einheitliche Darstellung gibt
2. diese Darstellung aufkommen muss für die (semantisch) kombinatorische Disambiguierung.

Das Kriterium der Koprädikation

Charakteristisch für Polysemie ist, dass wir über beide Bedeutungen prädizieren können:

(1)
a. Peter lernte das Buch auswendig und verbrannte es.
b. Das Abendessen war lecker aber dauerte drei Stunden.
c. Meine Ehe wurde auf dem Standesamt Düsseldorf geschlossen, war unglücklich und dauerte drei Jahre.

Koprädikationen zwingen uns dazu, den Einträgen eine einheitliche Semantik zu geben.

Das Kriterium der Koprädikation

Dagegen:

(2) ?Eine Bank ist praktisch zum sitzen, aber sie gibt praktisch keine Zinsen mehr.

(2) ist seltsam, denn es wiederspricht der uniformen Nutzung. In den polysemen Beispielen ist das nicht der Fall:

- Es scheint dass die Mehrdeutigkeit durch die Kombinatorik bereits aufgehoben wird, bevor sie semantisch relevant wird (in der Konjunktion).
- Deswegen gibt es keine globale Ambiguität, und deswegen kein Problem mit uniformer Nutzung!

Typentheorie

Der "kanonische" Ansatz zur semantischen Kombinatorik ist sicherlich die **Typentheorie**.

Dementsprechend kann man auch von einem kanonischen Ansatz zur Polysemie sprechen: die sog. **Punkt-Typen** (dot-types) von Asher.

Die Regeln von Asher passen aber auf keine Folie, deswegen würde ich lieber einen eigenen Ansatz vorstellen, der einfacher ist.

Einfache Typentheorie: Die Typen

Die einfache Typentheorie ist relativ einfach, hat aber bereits ihre Tücken (wie alles einfache).

T ist die Menge der elementaren Typen, mit üblicherweise:

- e 'entity' – Gegenstände. e denotiert also alle Gegenstände!
- t 'truth value' – Wahrheitswerte. t denotiert also $\{0, 1\}$.

Es gilt also:

- e, t sind (atomare) Typen
- falls α, β Typen sind, dann ist auch $(\alpha \rightarrow \beta)$ ein Typ.
- Sonst ist nix ein Typ!

Bedeutung von Typen

Atomare Typen *denotieren* die Menge der Objekte, die diesen Typ haben (s.o.).

$\alpha \rightarrow \beta$ denotiert die Menge aller **Funktionen** vom Typ α zum Typ β.
Sprich: $f : \alpha \rightarrow \beta$ bedeutet: f ist eine Funktion von Objekten in α zu Objekten in β.

- Konstanten (Namen?) haben den Typ e.
- Nomen (`Haus`, `Katze`, `Hund`) haben den Typ $e \rightarrow t$ (nämlich Mengen!)
- Ebenso intransitive Verben (`schlafen`,`sitzen`)
- Also bekommen transitive Verben den Typ $e \rightarrow (e \rightarrow t)$ (`tragen` etc.)

Semantische Typen sind also durchaus anders als syntaktische Typen!

Einfache Typentheorie: Die Typen

Es liegt auf der Hand das das nicht reicht für unsere Zwecke; wir brauchen zusätzliche Typen (subtypen von e):

- po – physikalische Objekte
- inf – Information
- ev – 'event', also Ereignisse
- bo – belebte Objekte

Also bekommt `tragen` den Typ $po \rightarrow (bo \rightarrow t)$

`lesen` hingegen nimmt sowohl physikalische Objekte als auch Information – wir kommen gleich dazu!

Typentheorie: Hierarchien

Die Typen stehen natürlich in einer Hierarchie:

- *e subsumiert po* und *bo*
- ...

Der Einfachheit halber nehmen wir an, es gibt einen Verband von Typen:

Verband

Ein Verband ist eine Struktur $(M, \leq, \wedge, \vee)$, so dass $(M, \leq)$ eine partielle Ordnung ist, und $x \leq y, z$ gdw. $x \leq y \wedge z$, $y, z \leq x$ gdw. $y \vee z \leq x$.

Wir haben also eine partielle Ordnung (Subsumption in unserem Fall), sowie eine **kleinste obere** und **größte untere** Schranke.

Typentheorie: Hierarchien

Verband der Type

Die atomaren Typen sind organisiert in einem Verband $(T, \sqsubseteq, \wedge, \vee, \perp)$, wobei

- $\sqsubseteq$ Subsumption ist (extensional: falls $e \sqsubseteq e'$, dann hat jedes Objekt mit Typ e auch implizit den Typ e'.
- $\perp$ ist $\sqsubseteq$ minimal. Man nutzt diese Konstante als Scheitern der Typisierung: kein Objekt hat $\perp$.

Wir haben also eine partielle Ordnung (Subsumption in unserem Fall), sowie eine **kleinste obere** und **größte untere** Schranke.

Normalerweise gibt es kein $\wedge$, im endlichen Falle ist das aber sowieso redundant (und Typen sind normalerweise endlich viele).

Typentheorie: Hierarchien

Wir brauchen den Typen $\vee$, denn üblicherweise haben wir Typen

- $(\alpha \vee \beta) \rightarrow \gamma$ – in unserem Fall `lesen`
- Wir müssen nun ausdrücken können dass es "egal" ist welchen Typ wir eingeben.

Normalerweise gibt es kein $\wedge$, im endlichen Falle ist das aber sowieso redundant (und Typen sind normalerweise endlich viele):

$$\alpha \wedge \beta = \bigvee \{\gamma : \gamma \sqsubseteq \alpha, \quad \gamma \sqsubseteq \beta\} \tag{1}$$

$\Longrightarrow \bigvee$ für endliche Mengen ist durch $\vee$ definiert!

Typentheorie: Punkt-Typen

Es gibt nun bei Asher noch einen weiteren Konstruktur: nämlich $\bullet$.

- Falls α, β Typen sind, dann ist auch $\alpha \bullet \beta$ ein Typ.
- Und wie gehabt: falls α, β Typen sind, dann ist auch $\alpha \rightarrow \beta$ ein Typ.

Man beachte: wir nutzen $\vee, \wedge$ nur als atomare Typen; wir applizieren sie also nicht auf $\bullet$- und $\rightarrow$-Typen.

Man beachte auch: die Menge der Typen ist selbst eine sehr interessante Logik (aber keine klassische!); für uns ist aber interessant, welche **Terme** wir **Typen zuweisen** können.

Typen und Terme

Wir kommen nun zu den Termen der **einfachen Typentheorie**. Sei $Var = \{x_1, x_2, ...\}$ die übliche unendliche Menge an Variablen.

- Falls $x \in Var$, dann ist $x \in term(Var)$
- Falls $t_1, t_2 \in term(Var)$, dann ist $(t_1 t_2) \in term(Var)$
- Falls $t \in term(Var)$, $x \in Var$, dann ist $(\lambda x.t) \in term(Var)$
- sonst nix!

Die äußersten Klammern eines Terms werden wie immer weggelassen. So einfach!

Typen und Terme

- Falls $x \in Var$, dann ist $x \in term(Var)$
- Falls $t_1, t_2 \in term(Var)$, dann ist $t_1 t_2 \in term(Var)$
- Falls $t \in term(Var)$, $x \in Var$, dann ist $\lambda x.t \in term(Var)$

Wir haben also eine sehr einfache Sprache von Termen. Die erste zentrale Frage in der (elementaren) Typentheorie ist:

Zentrales Problem der Typentheorie 1

Welchem Term kann man einen (und welchen?) Typ zuweisen?

Typen (Verb)

Zentrales Problem der Typentheorie 1

Welchem Term kann man einen und welchen Typ zuweisen?

Das wird auch für uns das zentrale Problem bleiben; andere zentrale Probleme der Typentheorie sind linguistisch eher weniger relevant (erwähnen wir am Rand).

Wir sagen auch: ein Term wird getypt. Wie typen wir Terme?

Typen (Verb)

Wir nehmen die sog. schwache Typisierung. Hier muss man soz. **beweisen**, dass einem Term ein Typ zugewiesen werden kann. Wir haben also einen **Kalkül**. Es gibt ein Axiom:

$$\overline{\Gamma \vdash x : \alpha},$$

wobei $x \in \mathit{Var}$, $\alpha \in \mathit{Tp}$, $\Gamma(x) = \alpha$.

Γ nennt man ein Typenurteil; man kann sich das als partielle Funktion $\mathit{Var} \to \mathit{Tp}$ vorstellen.

Sprich: einer Variable kann jeder Typ zugewiesen werden.

Typen (Verb)

Wir haben nun für alle zwei Konstruktoren eine jeweils Regel:

Abstraktion:

$$\frac{\Gamma \vdash t : \beta \quad \Gamma \vdash x : \alpha}{\Gamma \vdash \lambda x.t : \alpha \to \beta}$$

Applikation:

$$\frac{\Gamma \vdash t_1 : \alpha \quad \Gamma \vdash t_2 : \alpha \to \beta}{\Gamma \vdash t_2 t_1 : \beta}$$

Das wars auch schon! Man spricht hier übrigens von **Abstraktion** (Funktion auf Argument) und **Applikation** (Wert zu Funktion).

Typen (Verb)

Z.B.:

- der Term $\lambda x.xx$ ist wohlgeformt, kann aber nicht getypt werden. Denn Γ müsste x zwei verschiedene Typen zuweisen!
- $\lambda x.x$ ist wohlgeformt und getypt: er denotiert die Identitätsfunktion.
- $\lambda y.yx$ ist wohlgeformt und getypt: er denotiert die Anhebung eines Objekts zur Funktion.

Die Idee hinter Typentheorie ist: Terme ohne Typ haben keine Bedeutung (siehe Russels Paradox).

Als nächstes betrachten wir Regeln, mit denen wir Terme **normalisieren**, also vereinfachen.

Freie Variablen, Subsitutionen

Wir definieren $FV(t)$ induktiv wie üblich:

- $FV(x) = \{x\}$
- $FV(t_1 t_2) = FV(t_1) \cup FV(t_2)$
- $FV(\lambda x.t) = FV(t) - \{x\}$

Wichtig ist auch die **Substitution**: $t[t'/x]$: in t wird t' für x substituiert. Wir definieren das ebenfalls induktiv:

- $y[t'/x] = \begin{cases} t', \text{ falls } x = y \\ y \text{ andernfalls} \end{cases}$
- $(t_1 t_2)[t'/x] = t_1[t'/x]t_2[t'/x]$
- $(\lambda y.t)[t'/x] = \begin{cases} \lambda y.t, \text{ falls } x = y \\ \lambda y.t[t'/x] \text{ andernfalls} \end{cases}$

Freie Variablen, Subsitutionen

- $y[t'/x] = \begin{cases} t', \text{ falls } x = y \\ y \text{ andernfalls} \end{cases}$
- $(t_1 t_2)[t'/x] = t_1[t'/x]t_2[t'/x]$
- $(\lambda y.t)[t'/x] = \begin{cases} \lambda y.t, \text{ falls } x = y \\ \lambda y.t[t'/x] \text{ andernfalls} \end{cases}$

Substitutionen betreffen also nur freie Variablen!

Kleiner Ausblick (aber Hammer)

Geschlossene Terme

t ist geschlossen, falls $FV(t) = \emptyset$.
Man sagt ein Typ α ist **bewohnt**, falls es einen geschlossenen Term gibt, dem dieser Typ zugewiesen werden kann, also falls $\Gamma \vdash t : \alpha$ ableitbar ist.

Curry-Howard Isomorphismus

Ein Typ α ist bewohnt gdw. α ein Theorem intuitionistischer Logik ist.
Noch besser: der entsprechende Term kann transformiert werden in einen Beweis des Theorems (Natürliches Schließen).

Sprich: Typen sind Formeln, Terme sind Beweise!!

α, β, η-Konversion

Was uns auch interessiert ist die **Umformung** von Termen.
Hier gibt es drei Regeln: α, β, η.

Wichtig ist:

- Jede dieser Konversionen bewahrt die Gültigkeit des Typenurteils!

α, β, η-Konversion

α-Konversion:

$$\frac{\Gamma \vdash \lambda x.t : \gamma \quad \Gamma \vdash x : \beta \quad \Gamma \vdash y : \beta}{\Gamma \vdash \lambda y.t[y/x] : \gamma}\ \alpha$$

vorausgesetzt $y \notin FV(t)$

α-Konversion ist also "Variablenumbenennung", aber nur für gebundene Variablen!

Okkurenzen von x, die unabhängig gebunden sind, werden natürlich nicht ersetzt (siehe oben).

α, β, η-Konversion

β-Konversion (auch Reduktion):

$$\frac{\Gamma \vdash (\lambda x.t_1)t_2 : \alpha}{\Gamma \vdash t_1[t_2/x] : \alpha}\ \beta$$

vorausgesetzt alle freien Variablen von t_2 bleiben frei nach der Konversion.

β-Konversion ist also Applikation von Funktionen, Reduktion von Termen.
α-Konversion wird oft benötigt, um β-Konversion zu ermöglichen.

α, β, η-Konversion

η-Konversion ist eine Art Schmuddelkind der Typentheorie; wir nehmen sie Vollständigkeit halber auf:

$$\frac{\Gamma \vdash t : \alpha}{\Gamma \vdash (\lambda x.t)x : \alpha}\ \eta$$

η-Konversion sagt soviel wie Extensionalität; spielt aber keine so große Rolle hier.

α, β, η-Konversion

Wir schreiben $t \Longrightarrow_{\alpha\beta\eta} t'$ falls wir t' aus t mittels α ,β und η Konversion erhalten.

Church-Rosser Eigenschaft

Nimm an dass $t_1 \Longrightarrow_{\alpha\beta\eta} t_2$ und $t_1 \Longrightarrow_{\alpha\beta\eta} t_3$. Dann gibt es t_4 so dass $t_2 \Longrightarrow_{\alpha\beta\eta} t_4$ und $t_3 \Longrightarrow_{\alpha\beta\eta} t_4$.

Das bedeutet: alle getypten Terme konfluieren auf eine β-**Normalform**; die Reihenfolge der Schritte spielt hierbei keine Rolle!

α, β, η-Konversion

Church-Rosser Eigenschaft

Das gilt nur für getypte Terme! Ungetypte Terme haben keine Konfluenz und Normalform; vgl. $(\lambda x.xx)(\lambda x.xx)$

Das bedeutet: ungetypte Terme haben keine β-**Normalform**!

Subtypen

Falls wir partiell geordnete Typen haben mit Subsumption, brauchen wir noch folgende Regel:

$$\frac{\Gamma \vdash t : \alpha \quad \alpha \sqsubseteq \beta}{\Gamma \vdash t : \beta}$$

Hiermit stellen wir sicher, dass wir immer den passenden Typen zur Hand haben!
In unserem Fall brauchen wir evtl. einen expliziten Kalkül für $\sqsubseteq$ (wegen $\bullet$!)

Typentheorie – Vokabular

Bislang haben wir noch kein Lexikon, nur Variablen und Lambdas.

- Als nächstes nehmen wir uns noch ein Vokabular V mit fixen Ausdrücken.
- Wir nehmen weiterhin an, dass V **getypt** ist, sprich: jeder Eintrag kommt mit einem fixen Typ.
- Typenzuweisungen haben dann die Form

$$V, \Gamma \vdash t : \alpha$$

Typentheorie – Vokabular

Z.B. bekommen wir folgende Einträge:

- (liebt:$e \rightarrow (bo \rightarrow t)$)
- (blau:$po \rightarrow t$)
- (Auto:$po \rightarrow t$)

Das ist aber noch nicht ein *linguistisches Lexikon* (dazu später), sondern nur ein *typentheoretisches*!

Z.B. (`blaues Auto` können wir hier noch nicht bilden!)

Typentheorie – Vokabular

Man kann das nuzen um Prädikatenlogik einzubetten, indem man annimmt:

- $(\wedge : t \to (t \to t))$,
- $(\neg : t \to t)$,
- $(\exists x. : t \to t)$
- $(P : e \to (e \to t))$
- logische Variable $x : e$
- ...

mitnimmt.

So funktioniert, grob gesagt, typentheoretische Semantik a la Montague.

Typentheorie – Punkttypen

Wir nutzen nun unsere "feineren" Typen: das ergibt folgendes Vokabular:

- (liebt: $e \rightarrow (bo \rightarrow t)$
- (kind:$e \rightarrow t$) (!)
- (film:$e \rightarrow t$) (!)

Jetzt kommen Polyseme:

- (Buch:$(po \bullet inf) \rightarrow t$)
- (Essen:$(ev \bullet po) \rightarrow t$)

Polysemie

Wie behandeln wir diese Terme?

- Terme unseres Vokabulars werden von der Typentheorie behandelt wie beliebige Terme, nur mit fixen Typen.
- Wir haben aber keine Typentheorie für $\bullet$!
- Darum kümmern wir uns jetzt!

Polysemie

Wie behandeln wir diese Terme? Drei Dinge:

1. Wir müssen ambige Terme einführen.
2. Wir müssen ambige Funktionen einführen.
3. Wir müssen disambiguieren.

Das machen wir jetzt!

Polysemie: Ambiguieren

Wir führen ambige Terme ein:

$$\frac{\Gamma \vdash t_1 : \alpha \quad \Gamma' \vdash t_2 : \beta}{\Gamma \bullet \Gamma' \vdash t_1 \| t_2 : \alpha \bullet \beta}$$

Hier ist definiert:

$$\Gamma \bullet \Gamma'(x) = \begin{cases} \Gamma(x) \bullet \Gamma'(x), \text{ falls } x \in \Gamma \cap \Gamma' \text{ und } \Gamma(x) \neq \Gamma'(x) \\ \Gamma \cup \Gamma'(x) \text{ andernfalls} \end{cases}$$

So führen wir ambige Variablen ein, wir **ambiguieren**. Wichtig ist hier, der Konnektor $\|$ (im Gegensatz zu $\wedge, \vee$) ist intrinsisch mit den Typen verknüpft. Das macht das Wesen der Polysemie aus, die ja nur halbsemantisch, halbsyntaktisch ist.

Polysemie: Ambiguieren

$$\frac{\Gamma \vdash t_1 : \alpha \quad \Gamma' \vdash t_2 : \beta}{\Gamma \bullet \Gamma' \vdash t_1 \| t_2 : \alpha \bullet \beta}$$

Man beachte:

- Eine Variable x kann in der Prämisse verschiedene Typen haben –
- In der Konsequenz bekommt sie dann einen $\bullet$-Typen
- Es gibt hier sonst keine Typenkontrolle!
- Aber aus der Definition folgt (u.a.): $\Gamma \bullet \Gamma = \Gamma$!

Ambiges Abstrahieren

Als nächstes **abstrahieren** mit ambigen Variablen. Das ist geradeaus:

$$\frac{\Gamma \bullet \Gamma' \vdash t : \gamma \quad \Gamma \vdash x : \alpha \quad \Gamma' \vdash x : \beta}{\Gamma \bullet \Gamma' \vdash \lambda x.t : (\alpha \bullet \beta) \to \gamma}$$

Wir erlauben also, über ambige Variablen zu abstrahieren, vorausgesetzt wir haben vorher ambig typisiert! Beachte:

- $\Gamma \bullet \Gamma = \Gamma$!
- Wir können also beliebig "leere" Ambiguität einführen (Idempotenz) nach Schema $(\alpha \bullet \alpha) \to \beta$!

Applizieren – Disambiguieren

Applikation funktioniert nun wie folgt:

$$\frac{\Gamma \vdash \lambda x.t : (\alpha \bullet \beta) \to \gamma \quad \Gamma \vdash t' : \alpha}{\Gamma \vdash (\lambda x.t)t' : \gamma}$$

und

$$\frac{\Gamma \vdash \lambda x.t : (\alpha \bullet \beta) \to \gamma \quad \Gamma \vdash t' : \beta}{\Gamma \vdash (\lambda x.t)t' : \gamma}$$

- Das führt nun evtl. zu einem Mismatch von Typen auf der anderen Seite – das nutzen wir zur Disambiguierung!
- Bevor wir disambiguieren, brauchen wir aber erstmal β-Reduktion!

β-Konversion – Ambig!

β-Reduktion erfolgt wie immer, mit einem Zusatz:

- $y[t'/x] = \begin{cases} t', \text{ falls } x = y \\ y \text{ andernfalls} \end{cases}$
- $(t_1 t_2)[t'/x] = t_1[t'/x]t_2[t'/x]$
- $(\lambda y.t)[t'/x] = \begin{cases} \lambda y.t, \text{ falls } x = y \\ \lambda y.t[t'/x] \text{ andernfalls} \end{cases}$

$\Longrightarrow$ $(t_1 \| t_2)[t'/x] = (t_1[t'/x]) \| (t_2[t'/x])$

Also nichts neues hier!

Disambiguieren

Hier kommt der entscheidende Schritt: wir suchen eine Regel, die erlaubt, bei Typen-Fehler zu disambiguieren.

$$\frac{\Gamma \bullet \Gamma' \vdash t[t_1 \| t_2] : \gamma \quad \Gamma' \vdash t[t_2] : \bot}{\Gamma \vdash t[t_1] : \gamma}$$

Parallel dazu:

$$\frac{\Gamma \bullet \Gamma' \vdash t[t_1 \| t_2] : \gamma \quad \Gamma \vdash t[t_1] : \bot}{\Gamma \vdash [t_2] : \gamma}$$

Wir können also disambiguieren, wenn wir einen ambigen Term typen können – aber nicht die Teilterme!

Polysemie: Applizieren

Man sieht dass dieser Ansatz sehr allgemein ist:

- wir können auch leicht Szenarien konstruieren, in denen nicht disambiguiert wird;
- wir können ebenfalls Szenarien entwerfen, in denen typentheoretische Disambiguierung nicht möglich ist!

Curry-Howard

Ich habe mich bei dieser Formulierung am Curry-Howard Isomorphismus orientiert:

- Jede geschlossene Formel soll ein Theorem der (intuitionistischen) Logik der Ambiguität sein.
- Das scheint zu passen – Beweis steht noch aus.
- Ebenso wäre es schön, die Church-Rosser Eigenschaft zu behalten.
- Das scheint aber eher schwierig zu sein.

Polysemie: Beispiel

(3) Hans trägt ein Buch.

(4) Hans liest ein Buch.

(5) Hans liest und verbrennt ein Buch.

Polysemie: Die ganze Komplexität

Wir haben die Dinge bislang sehr vereinfacht:

- Wir haben angenommen, dass bei der Applikation die Typen *matchen* müssen. Also:

$$\frac{\alpha \to \beta \quad \alpha}{\beta}$$

- Bei einer komplexen Typenhierarchie ist das aber unangemessen; es gilt:

$$\frac{\alpha \to \beta \quad \gamma \quad \gamma \sqsubseteq \alpha}{\beta}$$

Wir brauchen also einen Kalkül der Typen um das Problem angemessen zu behandeln. Wir lassen das an dieser Stelle aus – das wird kompliziert.

Ein Montague Lexikon

Zur Montague-Grammatik gibt es auch eine Syntax (Kategorialgrammatik); die lassen wir aber an dieser Stelle aus.

Wir nehmen also an, dass ein Dämon dafür sorgt dass Worte richtig miteinander kombiniert werden.

Folgender Beispielsatz:

(6) a. Ein Hund schläft.
b. $(\exists x.)((hund(x) \wedge schlaft(x))) : t$

Ein Montague Lexikon

Ein Eintrag im Montague Lexikon hat die (vereinfachte) Form (w, t, α), wobei

1. w das Wort ist
2. t der Term
3. α der Typ

- hund:$\lambda x.hund(x) : e \rightarrow t$
- schläft:$\lambda x.schlaft(x) : e \rightarrow t$
- ein:$\lambda P.\lambda Q.(\exists x.)(P(x) \wedge Q(x)) : (e \rightarrow t) \rightarrow ((e \rightarrow t) \rightarrow t)$

Ein Montague Lexikon

- hund:$\lambda x.hund(x) : e \rightarrow t$
- schläft:$\lambda x.schlaft(x) : e \rightarrow t$
- ein:$\lambda P.\lambda Q.(\exists x.)(Px) \wedge (Qx) : (e \rightarrow t) \rightarrow ((e \rightarrow t) \rightarrow t)$

Wir bekommen also für den Term

(7) `(ein hund)schläft`

folgende Interpretation:

$$\frac{((\lambda P.\lambda Q.(\exists x.)(Px) \wedge (Qx))\ \lambda x.hund(x))\ \lambda x.schlaft(x) : t}{(\exists x.)((hund(x)) \wedge (schlaft(x))) : t}\ \beta$$

Bedeutung

Ein Term mit Typ t sollte wiederum wahr oder falsch sein. Die allgemeinen Sätze zur Typentheorie besagen nun:

- Jeder Term vom Typ t, der aus einem Montague Lexikon gebaut wird, ist eine prädikatenlogische Formel;
- und kann als solche in einem Modell interpretiert werden.
- Das ist eine Anwendung von Church-Rosser, denn die Prädikatenlogische Formel ist nur die β-Normalform (da sieht man wie wichtig das ist)!

Ein Montague Lexikon

(8) a. Hans sieht einen Hund.
b. $\exists x.((hund(x)) \wedge (sieht(h,x)) : t$

- hund:$\lambda x.hund(x) : e \rightarrow t$
- sieht:$\lambda x.\lambda y.sieht(y,x) : e \rightarrow (e \rightarrow t)$
- hans:$h : e$
- einen:$\lambda Q.\lambda P.\lambda y.(\exists x).(Qx) \wedge ((Px)y)$
$: (e \rightarrow t) \rightarrow ((e \rightarrow (e \rightarrow t)) \rightarrow (e \rightarrow t)$

NB: `einen` ist Akkusativ, das ist hier entscheidend!

Montague Grammar mit Polysemen

- buch:$\lambda x.(buch_{inf}\, x) \| (buch_{po}\, x) : (inf \bullet po) \to t$
- trägt:$\lambda x.\lambda y.tragt(y, x) : po \to (e \to t)$
- hans:$h : bo$
- ein:$\lambda Q.\lambda P.\lambda y.(\exists x).(Qx) \wedge ((Px)y)$
  $: (e \to t) \to ((e \to (e \to t)) \to (e \to t)$

Wir haben nun feinere Typen an dieser Stelle.

Montague Grammar mit Polysemen

- buch:$\lambda x.(buch_{inf}\, x) \| (buch_{po}\, x) : (inf \bullet po) \to t$
- versteht:$\lambda x.\lambda y.versteht(y, x) : inf \to (e \to t)$
- hans:$h : bo$
- ein:$\lambda Q.\lambda P.\lambda y.(\exists x).(Qx) \wedge ((Px)y)$
  $: (e \to t) \to ((e \to (e \to t)) \to (e \to t)$

Wir haben nun feinere Typen an dieser Stelle.

Montague Grammar mit Polysemen: Konjunktion

Die Eigenschaft von Konjunktion ist, dass sie alle Typen verknüpft:

$$\text{und}:??:\alpha \to (\alpha \to \alpha)$$

Wie sieht der zugehörige Term aus? Hier müssen wir tatsächlich unterscheiden, welchen Typ wir zuordnen.
Subjekt-NP haben die Form:

$$\text{NP1} - \lambda P.t : (e \to t) \to t$$
$$\text{NP2} - \lambda P'.t : (e \to t) \to t$$

Wichtig ist: wir müssen dafür sorgen, dass $P = P'$.

Montague Grammar mit Polysemen: Konjunktion

Wir machen das wie folgt:

- S-NP1 – $\lambda P.t : (e \to t) \to t$
- S-NP2 – $\lambda P'.t : (e \to t) \to t$
- subjekt-NP-und:$\lambda \mathbf{P}.\lambda \mathbf{Q}.\lambda P.(\mathbf{P}P \wedge \mathbf{Q}P)$:
$$((e \to t) \to t)) \to ((e \to t) \to t)) \to ((e \to t) \to t)))$$

Wir können hier sogar schwache Typen nutzen, d.h. der Typ ist nicht festgelegt.

Also: das ist Konjunktion **für alle** Typen $\beta \to \alpha$, wobei α atomar.

Montague Grammar mit Polysemen: Konjunktion

Objekt NP haben (s.o.) die Form

- O-NP1 $\lambda P.\lambda y.t : (e \to t) \to (e \to t)$
- O-NP2 $\lambda P'.\lambda y'.t : (e \to t) \to (e \to t)$
- Objekt-NP-und:$\lambda \mathbf{P}.\lambda \mathbf{Q}.\lambda P.\lambda y.((\mathbf{P}P)y \wedge (\mathbf{Q}P)y)$:
  $((e \to t) \to (e \to t)) \to ((e \to t) \to (e \to t))) \to ((e \to t) \to (e \to t))))$

Wir können hier schwache Typen nutzen, d.h. der Typ ist nicht festgelegt.

Also: das ist Konjunktion **für alle** Typen $\beta \to \alpha$, wobei $\alpha = \alpha_1 \to \alpha_2$.

Montague Grammar mit Polysemen: Konjunktion

VP haben (s.o.) die Form

- VP1 $\lambda x.t : e \to t$
- VP2 $\lambda x'.t : e \to t$
- VP-und:$\lambda \mathbf{P}.\lambda \mathbf{Q}.\lambda y.(\mathbf{P}y \wedge \mathbf{Q}y)$:
$$(e \to (e \to t)) \to ((e \to (e \to t)) \to (e \to (e \to t)))$$

Denn VPs haben bereits ihr Objekt!

Montague Grammar mit Polysemen: Konjunktion

Reduziere:

(9) Hans versteht und besitzt ein Buch.

(10) Eine Bank ist praktisch und zum sitzen und gibt Zinsen.

Literatur

- Hindley: Basic simple type theory (Typentheorie kompakt)
- Hindley, Seldin: Typed Lambda Calculus (ausführlich)
- Nicolas Asher: Dot-Types, Polysemy etc.
- Dowty, Wall, Peters: Introduction to Montague Semantics