Typentheorie und Lambdakalkül

Christian Wurm (Düsseldorf)

Düsseldorf, 23.11.2021

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 $(\alpha \to \beta)$ ein Typ.
- Sonst ist nix ein Typ!

Bedeutung von Typen

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

$\alpha \to \beta$ denotiert die Menge aller **Funktionen** vom Typ α zum Typ β.
Sprich: $f : \alpha \to \beta$ 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 $e \to t$ (nämlich Mengen!)
- Ebenso intransitive Verben (`schlafen`,`sitzen`)
- Also bekommen transitive Verben den Typ $e \to (e \to t)$ (`tragen` etc.)

Semantische Typen sind also durchaus anders als syntaktische Typen!

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

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:

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

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

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.

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:

(1) 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:$\lambda x.\mathit{hund}(x) : e \to t$
- schläft:$\lambda x.\mathit{schlaft}(x) : e \to t$
- ein:$\lambda P.\lambda Q.(\exists x.)(P(x) \wedge Q(x)) : (e \to t) \to ((e \to t) \to t)$

Ein Montague Lexikon

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

Wir bekommen also für den Term

(2) `(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 umgekehrt!);
- 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

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

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

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

Montague Grammar mit 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 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 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 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!

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