λ-Kalkül und Typen

0. Einführung und Geschichte des λ-Kalküls

Thorsten Altenkirch
email: alti@informatik.uni-muenchen.de

4. November 1996

Slide 1

Entwicklung des ungetypten λ-Kalküls

- Nach 1930: Church entwickelt einen logischen Formalismus, der als Teilsystem den ungetypten λ-Kalkül enthält.
- Entwicklung der λ-Notation aus der Principia.

$$\hat{x}.fx$$

$$\hat{}x.x + y$$

$$\lambda x.x + y$$

- Reine $\lambda - Terme : x,\ MN,\ \lambda x.M$

  λ-I Um $\lambda x.M$ zu formen, muß x in M vorkommen.

  λ-K Keine Einschränkung

Slide 2

- $$(\lambda x.M)N = M[x := N]$$

  β-Reduktion, β-expansion.

- 1935: Kleene und Rosser zeigen, daß Church's logische Theorie inkonsistent ist.

- 1936: Church und Rosser zeigen eine schwache Form der Konsistenz für den reinen λ-Kalkül.

- Kombinatorische Logik (Schönfinkel,Curry,...)

  $$Kxy = x \qquad Sxyz = xz(yz)$$

  Curry zeigt Äquivalenz zum λ-Kalkül, wenn zusätzliche Axiome angenommen werden (Extensionalität).

- Modelle? Scott '72: D_∞, Domaintheorie...

Slide 3

λ-Definierbarkeit

- Church entwickelt eine Zahlendarstellung im reinen λ-I-Kalkül:

  $$\lambda fx.fx, \lambda fx.f(fx), \lambda fx.f(f(fx)), \ldots$$

- Welche Funktionen sind λ-definierbar? Church: successsor, addition, multiplikation.

- Kleene löst das Problem der Darstellung der Vorgängerfunktion.

- Church'sche These: λ-definierbar = intuitiv berechenbar

- Äquivalenz mit Gödels rekursiven Funktionen von Kleene.

- Äquivalenz mit Turing-Maschinenberechenbar von Turing.

Slide 4

Typen

- Einfache Typen (Church): o, $\sigma \to \tau$.
- **Curry** Typen sind Prädikate, die ungetypte Terme sortieren.

  Church Getypte Terme: Getypte Variablen x^σ, getypte Abstraktion $\lambda x : \sigma.M$. Eindeutigkeit der Typisierung.
- Starke Normalisierung (Tait 63).
  Älterer Beweis von Turing, publiziert von Gandy in 1980.
- Schwichtenberg (76): Nur die erweiterten Polynome sind getypt λ-definierbar (mit $\overline{\mathrm{Nat}} = (o \to o) \to (o \to o)$).

  $$x + y \qquad xy \qquad sg(x) = \begin{cases} 0 & \text{wenn } x = 0 \\ 1 & \text{sonst} \end{cases}$$

  Insbesondere nicht die Vorgängerfunktion!

Slide 5

- Modelle? Einfach! (Mengentheoretische Modelle).
- Lambek: Getypter λ-Kalkül = Cartesian Closed Categories (CCCs).
- Scott (1980): Wie hätte der λ-Kalkül entdeckt werden sollen? Als CCCs!
  Der getypte Kalkül ist fundamentaler als der ungetypte.

Slide 6

Lambda-Kalkül und Logik

- Curry-Howard-Isomorphismus

λ-Kalkül	Logik
Typen	Aussagen
λ-Terme	Beweise
Reduktion	Schnittelimination

- System T = getypter λ-Kalkül + Nat +

$$\begin{array}{rcl} \mathrm{R} & : & \sigma \to (\mathrm{Nat} \to \sigma \to \sigma) \to \mathrm{Nat} \to \sigma \\ \mathrm{R}MN0 & = & M \\ \mathrm{R}MN(\mathrm{s}L) & = & NL(\mathrm{R}MNL) \end{array}$$

  Primitive Rekursion höherer Ordnung.

- System T berechenbar = beweisbar total in Arithmetik.

Slide 7

- Girard: System F mit variablen Typen (72)

$$X \qquad \Pi X.\sigma$$

$$\lambda X.M \qquad M[\sigma]$$

- System F berechenbar = beweisbar in Logik 2.Ordnung.
- Starke Normalisierung für System F (nicht in Arithmetik beweisbar). Cut-Elimination für Logik höherer Ordnung (Takeuti's Conjecture).
- System F wird von Reynolds wiederentdeckt: Polymorpher λ-Kalkül, Formalisierung von parametrischen Polymorphismus in der Programmierung.

Slide 8

- Per Martin-Löf 72: Intuitionistische Typtheorie
  Logik, die Programme und Beweise identifiziert.
  Erste Version inkonsistent (Girard), dann predikative Typtheorie.
- Coquand und Huet 85: *Calculus of Constructions*
  Typtheorie + F^ω.

Slide 9

Anwendungen

- LISP (McCarthy '59)
  Zunächst dynamische Bindung (aus heutiger Sicht inkorrekt)
  SCHEME: statische Bindung
- Landin (66): The next 700 Programming Languages
- ML (Milner 78)
  Einfache Typen + let-polymorphismus + Rekursion
  Typinferenz.
- Analyse und Design von Programmiersprachen
  Objekt-Kalküle
  Lineare Typen um Garbage-Collection zu kontrollieren
- Design von Beweisassistenten und Theorembeweisern
  ALF, Coq, Isabelle, LEGO ...

Slide 10

Übersicht über die Vorlesung (geplant)

1. Der einfach getypte λ-Kalkül
2. Reduktion
3. Kombinatorische Logik
4. Logische Relationen
5. Kategorielle Modelle
6. System T
7. System F
8. Typinferenz

Slide 11

Organisatorisches

- Aktuelle Informationen siehe

  `http://thelonius.tcs.informatik.uni-muenchen.de/~alti/lambda.html`

- Bitte per email registrieren! Details auf der Webpage.
- Keine Übung diese Woche.
- Scheinbedingungen: Übungen + Rücksprache oder Klausur.
- Sprechstunden per email vereinbaren.