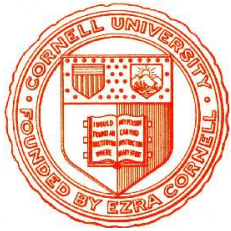


Automatisierte Logik und Programmierung

Einheit 7

Abhängige Datentypen



1. Motivation
2. Syntax, Typzugehörigkeit, Beweisregeln
3. Ausdruckskraft
4. Universen und Abhängigkeit

ERWEITERE AUSDRUCKSKRAFT DER TYPENTHEORIE

- **Einfache Typisierung von $\mathbb{B}$ ist zu restriktiv**

- Typschema für $\mathbb{B}$ ist $X \rightarrow X \rightarrow X$,
aber in **if b then s else $t \equiv b s t$** hängt X von s und t ab
- Einfache Typentheorie kann diese **Abhängigkeit** nicht beschreiben

- **Church Numerals sind Funktionen und Daten**

- Typschema für $\mathbb{N}$ ist $(X \rightarrow X) \rightarrow X \rightarrow X$, aber
in **$\mathbf{PRs}[base, h] \equiv \lambda n. n \ h \ base$** muß $n \in \mathbb{N} = (\mathbb{N} \rightarrow \mathbb{N}) \rightarrow \mathbb{N} \rightarrow \mathbb{N}$ sein
- Einfache Typentheorie kann diese **Doppelrolle von $\mathbb{N}$** nicht beschreiben

- **Funktionen wie $\lambda x.x$ sind polymorph**

- Das Typschema ‘für alle T gilt $\lambda x.x \in T \rightarrow T$ ’ ist nicht beschreibbar

- **Formuliere Abhängigkeit in Datentypen**

- Im Ausdruck $T[x]$ kann der Typ T vom Wert eines $x \in S$ abhängen
- Parameter x macht den Typ T polymorph

Zieldatentyp als Parameter in Formalisierung

• Neuformalisierung boolescher Operatoren

- $\mathbf{T} \equiv \lambda X. \lambda x. \lambda y. x$
- $\mathbf{F} \equiv \lambda X. \lambda x. \lambda y. y$
- $\mathbf{if\ } b \mathbf{\ then\ } s \mathbf{\ else\ } t : X \equiv b\ X\ s\ t$
- Führt zur Typisierung $\mathbb{B} \equiv \mathbf{X} : \mathbb{U} \rightarrow \mathbf{X} \rightarrow \mathbf{X} \rightarrow \mathbf{X}$
wobei $\mathbb{U}$ Repräsentant der Menge aller Datentypen
- Typisierung von s und t liefert Wert für den Parameter $\mathbf{X}$

• Church-Numerals mit Abhängigkeiten

- $\bar{n} \equiv \lambda X. \lambda f. \lambda x. f^n x$
- $\mathbf{PRs}[base, h] \equiv \lambda n. n(\mathbb{N})\ h\ base$
- Führt zu $\mathbb{N} \equiv \mathbf{X} : \mathbb{U} \rightarrow (\mathbf{X} \rightarrow \mathbf{X}) \rightarrow \mathbf{X} \rightarrow \mathbf{X}$
- Instantiierung von $\mathbf{X}$ mit $\mathbb{N}$ führt zu kontrollierter Selbstreferenz

Datentypen mit internen Querbezügen

● Verwendung in abstrakten Maschinenmodellen

- Endliche Automaten $(Q, \Sigma, q_0, \delta, F)$ mit $q_0 \in Q$, $\delta: Q \times \Sigma \rightarrow Q$, $F \subseteq Q$
 - Datentyp von q_0 hängt ab vom Wert der ersten beiden Komponenten

● Verwendung in der Programmierung

- Datentyp Record $[l_1:T_1; \dots; f_n:T_n]$
 - Datentyp T_i einer Komponente hängt vom Wert des Labels l_i ab
- Variant record `type date = Jan of 1..31 | Feb of 1..28 | ...`
 - Datentyp der zweiten Komponente hängt vom Wert der ersten ab
- Modellierung einer optionalen Eingabe abhängig von Testwerten
 - `λx. if x=0 then 4 else λinput.5*input`
 - Datentyp ist je nach Wert von **x** eine Zahl oder eine Funktion

FORMALISIERUNG VON ABHÄNGIGKEITEN

- **Abhängiges Produkt:** $x:S \times T[x]$
 - Elemente sind Paare (s, t)
 - Typ der zweiten Komponente t hängt vom Wert $s \in S$ der ersten ab
- **Abhängiger Funktionenraum:** $x:S \rightarrow T[x]$
 - Elemente sind Funktionen $\lambda x.t$
 - Typ $T[x]$ des Bilds t hängt vom Wert der Eingabe $s \in S$ ab
 - Einfachste Erweiterung des bisherigen Typsystems
- **Komplexere Abhängigkeiten ebenfalls möglich**
 - Durchschnitt $\cap x:S.T[x]$ / Vereinigung $\cup x:S.T[x]$ von Typfamilien
 - Elemente müssen zu allen/einem $T[x]$ für $x \in S$ gehören
 - Abhängiger Durchschnitt $x:S \cap T[x]$
 - Element s muß zu S und zu $T[s]$ gehören (Selbstreferenz!)
 - Gut für Formalisierung von Abstrakten Datentypen und Objekten

- **Erweiterte Syntax von Typ-Ausdrücken**
 - Typsymbole $T \in \mathcal{T}$
 - $x:S \rightarrow T[x]$ und (T) , wobei S und T Typen, $x \in \mathcal{V}$
 - $S \rightarrow T$ kürzt $x:S \rightarrow T[x]$ ab, wenn x in T nicht frei vorkommt
- **Objekt-Ausdrücke und Prioritäten wie zuvor**
- **Modifizierte Typzugehörigkeitsrelation**
 - $x \in T$ falls $x \in \mathcal{V}$ und T ein Typ
 - $\lambda y. t \in x:S \rightarrow T[x]$ falls $t[x/y] \in T[x]$ aus $x \in S$ folgt
 - $f t \in T$ falls $f \in S \rightarrow T$ und $t \in S$ für einen Typ S
 - $(t) \in T$, $t \in (T)$ falls $t \in T$
- **Typisierbarkeit und prinzipielles Typschema wie zuvor**

INFERENZREGELN FÜR ABHÄNGIGE FUNKTIONENRÄUME

Verfeinerte Beweisregeln des einfachen Kalküls

applyEq S	$\Gamma \vdash f\ t = g\ u \in T[t/x]$ $\Gamma \vdash f = g \in xS \rightarrow T$ $\Gamma \vdash t = u \in S$
lambdaEq $*$	$\Gamma \vdash \lambda x. t = \lambda y. u \in xS \rightarrow T$ $\Gamma, x':S \vdash t[x'/x] = u[x'/y] \in T[x'/x]$ $\Gamma \vdash S \in \mathbb{U}$
reduction	$\Gamma \vdash (\lambda x. u)\ s = t \in T$ $\Gamma \vdash u[s/x] = t \in T$
declaration i	$\Gamma, x:T, \Delta \vdash x \in T \quad \hat{=} \quad x=x \in T$
functionI	$\Gamma \vdash xS \rightarrow T \in \mathbb{U}$ $\Gamma \vdash S \in \mathbb{U}$ $\Gamma, x':S \vdash T[x'/x] \in \mathbb{U}$

**: Die Umbenennung $[x'/x]$ kann entfallen, wenn x nicht frei in Γ vorkommt*

Einfache Repräsentation vieler Basiskonzepte

- Polymorphe Formulierung Boolescher Ausdrücke

- $\mathbb{B} \equiv X:\mathbb{U} \rightarrow X \rightarrow X \rightarrow X$

- Korrekte Typisierung der Church Numerals

- $\mathbb{N} \equiv X:\mathbb{U} \rightarrow (X \rightarrow X) \rightarrow X \rightarrow X$

- Geschlossene Typisierung von Produkten

- $S \times T \equiv X:\mathbb{U} \rightarrow (S \rightarrow T \rightarrow X) \rightarrow X$

- Typisierung von Listen analog zu Zahlen

- $[] \equiv \lambda X. \lambda f. \lambda x. x$

- $t :: list \equiv \lambda X. \lambda f. \lambda x. f \ t \ (list(X) \ f \ x)$

- $list_ind[b, h] \equiv \lambda X. \lambda list. list(X) \ h \ b$

- $T \ list \equiv X:\mathbb{U} \rightarrow (T \rightarrow X \rightarrow X) \rightarrow X \rightarrow X$

DIE CURRY-HOWARD ISOMORPHIE SETZT SICH FORT

$\Gamma \vdash \lambda y. t \in x:S \rightarrow T$ BY **lambdaI**

$\Gamma, z:S \vdash t[z/y] \in T[z/x]$

$\Gamma \vdash S \in \mathbb{U}$

$\Gamma \vdash \forall x:S. A$ BY **allI**

$\Gamma, z:S \vdash A[z/x]$

$\Gamma \vdash f t \in T[t/x]$ BY **applyI** $x:S \rightarrow T$

$\Gamma \vdash f \in x:S \rightarrow T$

$\Gamma \vdash t \in S$

$\Gamma \vdash A[t/x]$ BY **InstUniv** $\forall x:S. A$

$\Gamma \vdash \forall x:S. A$

$\Gamma \vdash t \in S$

Analogien zwischen Typentheorie und Logik

Typ

Variable vom Typ S

Term vom Typ T (freie Variablen aus S_i)

Abhängiger Funktionenraum

Unabhängiger Funktionenraum

Formel

Annahme der Formel S

Beweis von T (Annahmen S_i)

All-Quantor

Implikation

Komplette Formalisierung der Logik möglich?

LOGIK IST DURCH ABHÄNGIGE DATENTYPEN CODIERBAR

● Konjunktion $A \wedge B$

- $A \wedge B$ gilt, wenn jede Aussage, die aus A und B folgt, gültig ist
- $A \wedge B \equiv \forall P:\mathbb{U}. (A \Rightarrow B \Rightarrow P) \Rightarrow P$

● Disjunktion $A \vee B$

- $A \vee B$ gilt, wenn jede Aussage gültig ist, die aus A und auch aus B folgt
- $A \vee B \equiv \forall P:\mathbb{U}. (A \Rightarrow P) \Rightarrow (B \Rightarrow P) \Rightarrow P$

● Negation $\neg A$ ist eine Abkürzung: $\neg A \equiv A \Rightarrow \text{ff}$

- Ein fundamentaler Widerspruch ist die Aussage, daß jede Aussage gilt
- $\text{ff} \equiv \forall P:\mathbb{U}. P$

● Existenzquantor $\exists x:S. A$

- $\exists x:S. A$ gilt, wenn jede Aussage gilt, die aus einem beliebigen $A[x]$ folgt
- $\exists x:S. A \equiv \forall P:\mathbb{U}. (\forall x:S. A \Rightarrow P) \Rightarrow P$

● Gleichheit $x=y \in A$

- $x=y$ gilt, wenn kein Prädikat x von y unterscheiden kann
- $x=y \in A \equiv \forall P:A \rightarrow \mathbb{U}. P(x) \Rightarrow P(y)$

ABHÄNGIGE TYPEN SIND EINE SINNVOLLE ERWEITERUNG

- **Zentrale Konzepte werden darstellbar**
 - Boolesche Ausdrücke, Natürliche Zahlen, Listen, Produkträume
 - Logischen Operatoren, Gleichheit
- **Informatik benötigt Abhängigkeitsstrukturen**
 - Endliche Automaten, (Variant) Records, optionale Argumente, ...
- **Unbegrenzte Selbstanwendung bleibt untypisierbar**
 - $\lambda x.x$ ist nicht typisierbar, da $T = x:T \rightarrow T$ keine Lösung hat
 - Parametrisierung wie bei $\mathbb{N}$ führt zu stark begrenzter Selbstanwendung
 - In $\lambda x.x(X)$ hätte x den Typ $T = X:\mathbb{U} \rightarrow X \rightarrow X$
 - Die Gleichung $T = T:\mathbb{U} \rightarrow T \rightarrow T$ enthält keine direkte Selbstreferenz
- **Es gibt einfache natürliche Modelle**
 - $x:S \rightarrow T[x]$ entspricht Abbildungen in eine Familie von Bildbereichen

Gibt es Probleme?

● Abhängigkeiten benötigen “höhere Typen”

- In $x:S \rightarrow T$ kommt x frei in T vor
 - x muß durch jeden Wert $s \in S$ ersetzt werden können
 - $T[s/x]$ ist ein Typ für beliebige $s \in S$, gehört also zum Universum $\mathbb{U}$
- Formale Beschreibung benötigt Funktion $\hat{T}$ mit $\hat{T}(s) \xrightarrow{\beta} T[s/x]$
- Was ist der Typ der Funktion $\hat{T}$?
 - $\hat{T}$ liefert für jedes $s \in S$ einen Typ, also $\hat{T} \in S \rightarrow \mathbb{U}$
 - $S \rightarrow \mathbb{U}$ muß ein Typ sein

● $\mathbb{U}$ bekommt eine Doppelrolle

- $\mathbb{U}$ ist ein Typ
- $\mathbb{U}$ ist Repräsentant des Universums aller Typen

Wie formuliert man die Doppelrolle von $\mathbb{U}$?

FORMALISIERUNG DER ROLLE DES UNIVERSUMS $\mathbb{U}$

Elegantester Ansatz: definiere $\mathbb{U} \in \mathbb{U}$ (Martin-Löf, 1971)

- **Einfacher, extrem ausdrucksstarker Kalkül**
- **Kein Problem mit Russel's Paradox**
 - Die Menge $\{S \mid S \notin S\}$ ist wegen der Typstruktur nicht formulierbar
- **Theorie scheitert aber an Girard's Paradox**
 - Eine Variante des **Burali-Forti Paradox** für Ordinalzahlen läßt sich in einer Theorie formalisieren, die abhängige Datentypen und das Axiom $\mathbb{U} \in \mathbb{U}$ enthält (Girard, 1972)



Jede Theorie mit $x:S \rightarrow T$ und $\mathbb{U} \in \mathbb{U}$ ist inkonsistent

GIRARD'S PARADOX

● Grundlage: Variante des Burali-Forti Paradox

- Betrachte die Menge aller Wohlordnungen $(M, <_M)$ auf Mengen $M \in \mathbb{U}$
 - $<_M$ transitiv, ohne unendlich absteigende Kette $x_0 >_M x_1 >_M x_2 \dots$
- Definiere eine Ordnung auf dieser Menge $(M, <_M) <_{\mathbb{U}} (M', <_{M'})$
 $\equiv \exists f: M \rightarrow M'. f \text{ ordnungsisomorph} \wedge \exists y: M'. \forall x: M. f(x) <_{M'} y$
- Dann ist $(\mathbb{U}, <_{\mathbb{U}})$ eine Wohlordnung und $(M, <_M) <_{\mathbb{U}} (\mathbb{U}, <_{\mathbb{U}})$
gilt für jede Wohlordnung $(M, <_M)$ mit $M \in \mathbb{U}$
d.h. $(\mathbb{U}, <_{\mathbb{U}})$ ist ein maximales Element dieser Ordnung
- Wegen $\mathbb{U} \in \mathbb{U}$ folgt dann auch $(\mathbb{U}, <_{\mathbb{U}}) <_{\mathbb{U}} (\mathbb{U}, <_{\mathbb{U}})$
- Damit gibt es in $<_{\mathbb{U}}$ eine unendlich absteigende Kette

● Paradox formulierbar, falls Theorie $\mathbb{U} \in \mathbb{U}$ zuläßt

- Alle Ordnungskonzepte sind mit Higher-Order Quantoren definierbar
- $x: S \rightarrow T$ reicht zur Formalisierung der Quantoren

ALTERNATIVE REPRÄSENTATIONEN DES UNIVERSUMS

● Theorie benötigt Universum und Abhängigkeiten

- Universum $\mathbb{U}$ ist notwendig zur Formulierung des “Typseins”
- Abhängige Datentypen sind notwendig für die Ausdruckskraft
- Formulierung der Abhängigkeit benötigt Typen mit freien Variablen
- Der elegantest Weg ist versperrt: $\mathbb{U}$ darf selbst kein Typ sein

● Syntaktische Alternative

System F, Coq

- **Trennung**: Variablen in Typausdrücken sind keine Objektvariablen
 - Funktionen $\hat{T} \in S \rightarrow \mathbb{U}$ werden für $x:S \rightarrow T[x]$ nicht benötigt
- Alles andere kann i.w. beibehalten werden

● Semantische Alternative

Martin-Löf (1978), Nuprl

- **Unterschied** von Objekt- und Typausdrücken liegt in der Bedeutung
 - $\mathbb{U}$ und $S \rightarrow \mathbb{U}$ sind keine “einfache” Typen wie $\mathbb{N}$ oder $S \rightarrow T$
- **Universum gliedert sich in Stufen** $\mathbb{U} \in \mathbb{U}_2 \in \mathbb{U}_3 \dots$
 - Nur die Gesamtheit aller Stufen beschreibt das “Typsein”

Keine der Alternativen ist perfekt

SYNTAKTISCHE ABTRENNUNG DES UNIVERSUMS

- **Spezielle Syntax vermeidet $\mathbb{U}$ in Typausdrücken**

- Strikte syntaktische Trennung von Objekt- und Typausdrücken
- In $x:S \rightarrow T[x]$ ist x eine Spezies Variable und $T[x]$ ein Typ mit Parameter aus S

- **Girard's Paradox wird elegant umgangen**

- Typisierte Objekte können selbst keine Typen enthalten
- $\mathbb{U}$ ist selbst kein Typ — $\mathbb{U} \in \mathbb{U}$ ist nicht formulierbar
- Unklar ob andere Paradoxien formulierbar sind (unwahrscheinlich)

- **Theorie kann minimal bleiben**

- Nur abhängiger Funktionenraum (“ Π -Type”) erforderlich
- Andere Konzepte wie bisher repräsentierbar (unnatürlich?)

- **Beweisformalismus hat relativ komplexe Syntax**

- Formulierung von Coq verbessert System F signifikant

SEMANTISCHE AUFTRENNUNG DES UNIVERSUMS

- **Semantisch der klarste Ansatz**

- Abstraktes ‘Typsein’ ist syntaktisch nicht mit **einem** Konzept erfaßbar
 - Ein Universum von Typen ist selbst kein einfacher Typ
- Das Universum einfacher Typen ist Element eines höheren Universums

- **Typhierarchie** $\mathbb{U} \in \mathbb{U}_2 \in \mathbb{U}_3 \dots$

- Typsein ist nur durch **Gesamtheit aller Universen** repräsentierbar
- Vermeide Komplikationen mit Mischtypen wie $S \rightarrow \mathbb{U}$ durch
“kumulative” Universen $\mathbb{U} \subseteq \mathbb{U}_2 \subseteq \mathbb{U}_3 \dots$ –

- **Theorie wird sehr umfangreich**

- **Viele Konzepte nicht** in voller Allgemeinheit **simulierbar**
 - Definitionen mit Universumsparemeter wären nicht sinnvoll
- Nichtsimulierbare **Konzepte müssen explizit formalisiert werden**
 - Aufwendig für Theorieentwickler, sehr nützlich für Anwender
- Formulierung von **Nuprl** erweitert Martin-Löf’s Theorie erheblich

FORMALE KONSTRUKTIVE THEORIEN – WO HIN?

- **Theorie abhängiger Typen ist ausdrucksstark**

Umfaßt **Prädikatenlogik** (Schließen über logische **Struktur**)

λ -Kalkül (Schließen über **Wert** von Ausdrücken)

Typanalyse (Schließen über **Eigenschaften** von Ausdrücken)

- **Minimaler Kalkül elegant aber unnatürlich**

- $\mathbb{B}, \mathbb{N}, \times, \wedge, \vee, \neg, \exists, =, \dots$ sind fundamentale mathematische Konzepte
- Anwendungen benötigen Formalisierungen dieser Konzepte
- Eine Simulation erschwert formales Schließen in der Praxis

- **Formalisiere wichtige Konzepte explizit**

- **Konstruktive (intuitionistische) Typentheorie** präzisiert syntaktische Repräsentation, Bedeutung und Inferenzregeln für **Standardkonzepte** aus Mathematik & Programmierung
- Umfang des Formalismus erfordert **Systematik für Aufbau des Kalküls**