

# Automatisierte Logik und Programmierung

## Teil II

### Konstruktive Typentheorie



# Automatisiert Logik und Programmierung

## Einheit 8



## Systematik des Aufbaus großer Kalküle



1. Anforderungen & Grundprinzipien
2. Syntaktische Struktur & Präsentation
3. Formale Semantikbeschreibung
4. Beschreibung des Inferenzsystems
5. Erweiterungen des Typsystems

## Vereinige Logik, Berechnung und Typstruktur

- **Den “einen richtigen Weg” gibt es nicht**
  - Der eleganteste Ansatz (abhängige Typen +  $\mathbb{U} \in \mathbb{U}$ ) ist inkonsistent
  - Alle anderen Theorien haben spezifische Vor- und Nachteile
    - Keine ist in jeder Hinsicht besser als die anderen
  - Vor- und Nachteile sind nach subjektiven Gesichtspunkten abzuwägen
    - Es hängt ab von den Zielen, die man mit der Theorie verfolgt
- **Nicht alle Ziele sind gleichzeitig erreichbar**
  - Mathematische Eleganz: einfacher Formalismus mit klarer Semantik
  - Hohe Ausdruckskraft: aller relevanten Konzepte formalisierbar
    - Die Mathematik ist nicht vollständig axiomatisierbar!
  - Implementierbarkeit: leichte Umsetzung als Beweiser-Software
  - Lesbarkeit: “natürliche” Syntax für Standardkonzepte
  - Automatisierbarkeit: computergestützte Beweisführung möglich

# COMPUTATIONAL TYPE THEORY (CTT)

- **Formalismus zum Schließen über Berechnung**

- Praktische Nutzbarkeit steht im Vordergrund
- Ausdruckskraft, Implementierbarkeit und Lesbarkeit sind wichtiger als mathematische Eleganz und vollständige Automatisierbarkeit

- **Basis: Martin-Löf's intuitionistische Typentheorie**

- Formalisierung wichtiger Konzepte aus Mathematik & Programmierung gekoppelt mit präziser Semantik und Beweiskalkül
- Signifikante Erweiterung von Martin-Löf's ursprünglicher Theorie
- Modifikation des Formalismus für Beweis- und Programmentwicklung

- **Implementiert im Nuprl Beweissystem**

- Formalisierung orientiert sich an Möglichkeiten eines Computersystems
- Implementierung ist direkte (korrekte!) Umsetzung des Beweiskalküls

# ENTWURFSENTSCHEIDUNGEN SIND ZU TREFFEN

## 1. Umfang des Formalismus: minimal oder nicht?

- Wieviele Konzepte sollten innerhalb der Theorie vordefiniert sein?
- Syntaktische oder semantische Behandlung des Typseins?
- Wird extensionale Gleichheit in den Kalkül integriert?
- Wird die Theorie geschlossen oder erweiterbar gestaltet?

## 2. Syntax: starr oder flexibel?

- Vorrang für leichte Lesbarkeit oder einfache Syntaxanalyse?

## 3. Semantik: modelltheoretisch oder direkt?

- Direkte Semantik liefert Konkurrenz zur Mengentheorie
- Was ist mit Normalisierbarkeit, Konfluenz, Typisierbarkeit?

## 4. Beweiskalkül überprüfend oder konstruierend?

- Kann Typzugehörigkeit und Gleichheit nur nachgewiesen werden oder kann man Terme (Programme) auch während des Beweises erzeugen

## Entscheidungen beeinflussen sich gegenseitig

- Manche Entscheidungen sind erst im Rückblick zu verstehen

- **Minimale Kalküle sind mathematisch elegant**

- + : Gut für Entwickler: Nachweis von Eigenschaften leichter
- + : Leicht zu erlernen: Regelsatz ist überschaubar
- : Mathematische Grundkonzepte müssen (korrekt!) simuliert werden
  - Großer Formalisierungsaufwand für (Erst-)Anwender
- : Einfache Schlüsse verlangen viele elementare Inferenzen (Effizienz?)

- **Umfangreiche Kalküle sind praktisch nutzbar** ✓

- + : Gut für Anwender: wichtige Grundkonzepte sind vordefiniert
  - Inferenzregeln arbeiten direkt auf Ebene dieser Konzepte
- : Viel zu lernen: großer Formalismus mit vielen Termen und Regeln
  - Vereinfachung möglich durch Namenssystematik und Taktiken
- : Aufwendig für Entwickler: in Beweisen sind viele Fälle zu prüfen
  - Vereinfachung möglich durch systematischen Aufbau und Minimierung der Einflüsse zwischen formalen Konzepten

# WELCHE KONZEPTE SOLLTEN VORDEFINIERT SEIN?

## ● Basiskonzepte

- Zahlen liegen fast allen Argumenten zugrunde
- Funktionen zentrales Berechnungskonzept
- Produkte elementarste Datenstruktur
- Abhängige Datentypen für genügende Ausdruckskraft
- Gleichheit wesentlich für Analyse von Werten
- Universen Repräsentation des Typseins
- Logik für Analyse der Struktur von Argumenten

## ● Programm- und Datenstrukturen

- Listen, Produkte, Aufzählungstypen, Textketten
- Vordefinierte Basisarithmetik statt minimaler Zahlendefinition
- Kontrollierte rekursive Programm- und Datenstrukturen
- Neuere Programmierkonstrukte wie Module, Objekte, Vererbung,...

## ● Mengenkonzepte

- Teilmengen, Vereinigung, Durchschnitt, Restklassen, ...

## ● Syntaktische Behandlung des Typseins

- + : Einfache Repräsentation des Typuniversums
- : Objekte und Typen müssen syntaktisch getrennt werden

## ● Semantische Behandlung des Typseins



- + : Natürliche Art der Unterscheidung von Objekten und Typen
- + : Keine Einschränkung für Verwendung von Symbolen durch Anwender
- : Kumulative Universenhierarchie  $\mathbb{U} = \mathbb{U}_1 \subseteq \mathbb{U}_2 \subseteq \mathbb{U}_3 \subseteq \dots$  erforderlich

## ● Intensionale Gleichheit

- + : Leicht zu testen, da rein syntaktisches Konzept (textliche Identität)
- : Semantische Gleichheit muß vom Benutzer definiert werden

## ● Extensionale Gleichheit



- + : Natürliche, wert-basierte Gleichheit in Theorie verankert
- : Extensionale Sicht bringt viele Unentscheidbarkeiten in die Theorie



## ● Geschlossene Formalisierung

- Theorie wird ein für alle Mal festgelegt
- + : Erlaubt Beweise mit Fallunterscheidung über alle möglichen Terme
- : Alle nicht explizit definierten Konzepte müssen simulierbar sein
  - Teile von Mathematik & Programmierung nicht ausdrückbar ?

## ● Erweiterbare Formalisierung



- Theorie kann jederzeit um neue Konzepte erweitert werden
- + : Neu entwickelte Konzepte können später integriert werden
  - Möglich bei systematischen Aufbau und Konzeptunabhängigkeit
- : Begrenzte Selbstreflektion: Schließen über alle Terme unmöglich
  - Derartige Beweise würden bei Erweiterungen ungültig

# LEITBEISPIEL: BASISKONZEPTE DER CTT

- **Variablensymbole**

- Ascii-Bezeichner, einheitlich für Objekt- und Typvariablen

- **Natürliche Zahlen**  $\mathbb{N}$

- Vorläufig als Minimalkonzept wie bei Peano-Axiomen
- Null  $0$ , Nachfolger  $s(i)$ , induktive Definition  $PR[f, n, x \mapsto g](i)$

- **Funktionsräume**  $x:S \rightarrow T$  und  $S \rightarrow T$

- Funktionsdefinition  $\lambda x. t$  und -anwendung  $f t$
- Typ des Bilds einer Funktion kann vom Wert der Eingabe abhängen

- **Produkträume**  $x:S \times T$  und  $S \times T$

- Paarbildung  $\langle a, b \rangle$  und uniforme Projektion  $\text{let } \langle x, y \rangle = p \text{ in } t$
- Typ der zweiten Komponente kann vom Wert der ersten abhängen

- **Gleichheit als Datentyp**  $s=t \in T$

- Notwendig für konstruierenden Beweiskalkül (siehe Folie 39)

- **Universen**  $\mathbb{U}_i$

- Repräsentation der Typ-Eigenschaft

## ENTWURFSENTSCHEIDUNGEN FÜR CTT (2): SYNTAX

- **Maschinenlesbarkeit ist notwendig**

- + : Leichte Erzeugung des abstrakten Syntaxbaums aus formalem Text
  - Syntaxbaum essentiell für Verarbeitung von Ausdrücken
- : Starre, maschinenlesbare Syntax ist unnatürlich

- **Flexibilität ist wichtig für Lesbarkeit**

- + : Verständlichkeit steigt durch Verwendung von Lehrbuchnotationen
  - Es gibt viele gleich gute Notationen für dasselbe Konzept
- : Zu viel Flexibilität macht Syntax sehr schwer zu verarbeiten

- **Versuche beides zu vereinigen**



- Vielfalt der Konzepte verlangt standardisierte Mechanismen für
  - Syntaktische Analyse vieler verschiedenartiger Ausdrücke
  - Erkennen freier und gebundener Vorkommen von Variablen
  - Substitution
- Mechanismen sollten Erweiterungen (auch konservative) unterstützen

## SYNTAXDEFINITION OHNE SYSTEMATISIERUNG

- $x$  Ausdruck, falls  $x \in \mathcal{V}$
- $(t)$  Ausdruck, falls  $t$  Ausdruck
- $\mathbb{N}$  und  $0$  sind Ausdrücke
- $\mathbf{s}(t)$  Ausdruck, falls  $t$  Ausdruck
- $\mathbf{PR}[f, n, x \mapsto g](t)$  Ausdruck, falls  $x, n \in \mathcal{V}$  und  $f, g$ , und  $t$  Ausdrücke
- $x : S \rightarrow T$  und  $S \rightarrow T$  Ausdrücke, falls  $x \in \mathcal{V}$ ,  $S$  und  $T$  Ausdrücke
- $\lambda x. t$  Ausdruck, falls  $x \in \mathcal{V}$  und  $t$  Ausdruck
- $f t$  Ausdruck, falls  $t$  und  $f$  Ausdrücke
- $x : S \times T$  und  $S \times T$  Ausdrücke, falls  $x \in \mathcal{V}$ ,  $S$  und  $T$  Ausdrücke
- $\langle a, b \rangle$  Ausdruck, falls  $a$  und  $b$  Ausdrücke
- **let**  $\langle x, y \rangle = p$  **in**  $t$  Ausdruck, falls  $x, y \in \mathcal{V}$  und  $p$  und  $t$  Ausdrücke
- $s = t \in T$  Ausdruck, falls  $s, t$  und  $T$  Ausdrücke
- $\mathbb{U}_i$  Ausdruck für alle  $i \geq 0$

**Syntaxanalyse wird schnell unüberschaubar**

# EIN EINHEITLICHES PRINZIP DER SYNTAXDEFINITION

## Trenne Abstraktion und Darstellung von Termen

- **Strikte maschinenlesbare interne Struktur**
  - Abstrakte Einheitssyntax für Terme – leicht zu verarbeiten
  - Einheitliche Mechanismen für Variablenvorkommen und Substitution
  - Individuelle Konzepte sind Terme dieser Syntax  $\mapsto$  Tabelleneinträge
- **Flexible externe Präsentation**
  - Notation (Präsentation) wird getrennt von Konzept (formaler Term)
  - Display Formen unterstützen vielfältige Notationen für denselben Term
    - Ermöglicht (nahezu) natürliche externe Syntax
    - Erlaubt Notationswechsel ohne Veränderung des eigentlichen Terms
- **Prinzip unterstützt Erweiterungen der Theorie**
  - Ergänze Tabelle der Terme und definiere Display Formen
  - Gleiche Methodik für echte und konservative Erweiterungen

# UNIFORME INTERNE SYNTAX – WAS IST NÖTIG?

## ● Name

- Terme müssen eindeutig identifizierbar sein
- $\mathbb{N}$  ist Präsentationsform eines Terms mit Namen **nat**

## ● Parameter in Namen

- Namen können parametrisiert werden
- $x$  ist eine Variable mit Namen  $x$  – Termname ist **var** $\{x:v\}$

## ● Teilterme

- Die meisten Terme sind aus anderen Termen zusammengesetzt
- $f\ t$  ist Abkürzung für einen Ausdruck der Form **apply** $(f;t)$

## ● Gebundene Variablen in Teiltermen

- Teilterme können Variablen enthalten, die gebunden werden
- In  $\lambda x.t$  wird  $x$  in  $t$  gebunden – Ausdruck ist **lam** $(x.t)$
- In  $x:S \rightarrow T$  wird  $x$  in  $T$  (nicht in  $S$ ) gebunden  
Ausdruck ist **function** $(S; x.T)$

**Allgemeine Form:**  $opid\{p_1 \cdot \dots p_k\}(x_1^1 \dots x_{m_1}^1.t_1; \dots x_1^n \dots x_{m_n}^n.t_n)$

## TERME DER CTT: PRÄZISE DEFINITION

Die **Abstraktionsform** eines **Terms** hat die Gestalt

$$\textit{opid}\{p_1:F_1, \dots, p_k:F_k\} (x_1^1, \dots, x_{m_1}^1 \cdot t_1; \dots, x_1^n, \dots, x_{m_n}^n \cdot t_n)$$

wobei

- $\textit{opid}\{p_1:F_1, \dots, p_k:F_k\}$  **Operator**:
  - $\textit{opid}$ : **Operatorname**  $\rightarrow$  *Operatorentabelle*  
Operatorentabelle muß Operator var enthalten
  - $p_j:F_j$ : **Parameter**
    - $F_j$ : **Parametertyp** (var, nat, tok, str, level)
    - $p_j$ : **Parameterwert**
- $x_1^i, \dots, x_{m_i}^i \cdot t_i$  **gebundener Term**, mit Term  $t_i$  und Variablen  $x_k^j$   
( $m_1, \dots, m_n$ ) ist die **Arität** (Anzahlen der Variablen aller Teilterme)

**Abstraktionsform und Darstellungsform werden  
getrennt behandelt, aber simultan betrachtet**

# AUSZUG DER OPERATORENTABELLE

| <i>Operator und Termstruktur</i>     | <i>Standard-Darstellungsform</i> |
|--------------------------------------|----------------------------------|
| <b>var</b> { <i>x</i> : <i>v</i> }() | <i>x</i>                         |
| <b>Nat</b> { }()                     | $\mathbb{N}$                     |
| <b>zero</b> { }()                    | 0                                |
| <b>suc</b> { }(t)                    | s(t)                             |
| <b>p_rec</b> { }(t; f; n, x.g)       | PR[f, n, x ↦ g](t)               |
| <b>function</b> { }(S; x.T)          | <i>x</i> :S → T                  |
| <b>lambda</b> { }(x.t)               | λ <i>x</i> .t                    |
| <b>apply</b> { }(f; t)               | f t                              |
| <b>product</b> { }(S; x.T)           | <i>x</i> :S × T                  |
| <b>pair</b> { }(s, t)                | ⟨s, t⟩                           |
| <b>spread</b> { }(e; x, y.u)         | let ⟨x, y⟩ = e in u              |
| <b>Axiom</b> { }()                   | A <sub>x</sub>                   |
| <b>equal</b> { }(s; t; T)            | s = t ∈ T                        |
| <b>universe</b> { <i>j</i> :1 }()    | U <sub>j</sub>                   |



# DISPLAY FORMEN

## ● Beschreibe Notation eines abstrakten Terms

**EdAlias** lam:  $\lambda\langle x:\text{var}\rangle.\langle t:\text{term}:E\rangle == \text{lambda}\{\}(\langle x\rangle.\langle t\rangle)$

**#Hd** A::  $\lambda\langle x:\text{var}\rangle,\langle \#: \text{term}:E\rangle == \text{lambda}\{\}(\langle x\rangle.\langle \#\rangle)$

**#Tl** A::  $\langle x:\text{var}\rangle.\langle t:\text{term}:E\rangle == \text{lambda}\{\}(\langle x\rangle.\langle t\rangle)$

**#Tl** A::  $\langle x:\text{var}\rangle,\langle \#: \text{term}:E\rangle == \text{lambda}\{\}(\langle x\rangle.\langle \#\rangle)$

- Linke Seite beschreibt, wie rechte Seite darzustellen ist
- Beliebige Anordnung von Parametern (spitze Klammern) in Freitext

## ● Sonderformen erhöhen Flexibilität

- Weglassen von Parametern links möglich (implizite Information)
- Iteration kann anders geschrieben werden:  $\lambda x, y. t$  statt  $\lambda x. \lambda y. t$
- **Alias** für Aufruf im Editor kann vom Operatornamen abweichen
- Unterstützung für **Layout** großer Terme, automatische Klammerung ...

**EdAlias** primrec:

$h(\langle t:\text{int}\rangle) \{->\} \text{where } h(0) = \langle f:\text{base}\rangle$

$\{\backslash\backslash\} \text{and } h(n+1) = \langle g:\text{step}\rangle(\langle n:\text{var}\rangle, h(n)/\langle x:\text{var}\rangle)$

$== \text{p\_rec}\{\}(\langle t\rangle; \langle f\rangle; \langle n\rangle, \langle x\rangle.\langle g\rangle)$

# ERWEITERUNGEN DER SYNTAX

- **Abstraktionen für konservative Erweiterungen**

- $\text{add}\{\}(.i;.j) \equiv \text{PR}[i, n, \text{add-in} \mapsto \mathbf{s}(\text{add-in})](j)$
- $\text{mul}\{\}(.i;.j) \equiv \text{PR}[0, n, \text{mul-in} \mapsto \text{add}\{\}(. \text{mul-in}; j)](j)$
- Linke Seite beschreibt Erweiterung der Operorentabelle
- Rechte Seite beschreibt Semantik durch existierende Terme

- **Systemeintrag für echte Theorieerweiterungen**

- Term wird direkt in Operorentabelle vermerkt
- Semantik und Inferenzregeln sind explizit zu beschreiben

- **Displayformen sind in beiden Fällen anzugeben**

- **EdAlias**  $\text{add}: \langle i:\text{int} \rangle + \langle j:\text{int} \rangle == \text{add}\{\}(. \langle i \rangle; . \langle j \rangle)$
- **EdAlias**  $\text{mul}: \langle i:\text{int} \rangle * \langle j:\text{int} \rangle == \text{mul}\{\}(. \langle i \rangle; . \langle j \rangle)$

- **Beispiel für Präsentation von Termen**

- Interne Form  $\text{lambda}\{\}(x.\text{lambda}\{\}(y.\text{lambda}\{\}(z.\text{add}\{\}(. \text{var}\{x:v\}()); . \text{mul}\{\}(. \text{var}\{y:v\}(); . \text{var}\{z:v\}())))))$
- Externe Präsentation  $\lambda x, y, z. x + y * z$

# VORKOMMEN VON VARIABLEN (OHNE SYSTEMATIK)

- $x$ : die Variable  $x$  kommt frei vor;  $y \neq x$  kommt nicht vor.
- $(t)$ : freie Vorkommen von  $x$  in  $t$  bleiben frei, gebundene Vorkommen bleiben gebunden.
- $\mathbb{N}, 0$ :  $x$  kommt nicht vor.
- $\mathbf{s}(t)$ : freie Vorkommen von  $x$  in  $t$  bleiben frei, gebundene Vorkommen bleiben gebunden.
- $\mathbf{PR}[f, n, x \mapsto g](t)$ : freie Vorkommen von  $x$  und  $n$  in  $g$  werden gebunden;  
freie Vorkommen von  $y \neq x/n$  in  $g$  bleiben frei, gebundene bleiben gebunden.
- $x:S \rightarrow T$ : freie Vorkommen von  $x$  in  $T$  werden gebunden;  
freie Vorkommen von  $y \neq x$  in  $T$  bleiben frei, gebundene Vorkommen bleiben gebunden.
- $\lambda x.t$ : freie Vorkommen von  $x$  in  $t$  werden gebunden;  
freie Vorkommen von  $y \neq x$  in  $t$  bleiben frei, gebundene Vorkommen bleiben gebunden.
- $f t$ : freie Vorkommen von  $x$  in  $f$  und  $t$  bleiben frei, gebundene Vorkommen bleiben gebunden.
- $x:S \times T$ : freie Vorkommen von  $x$  in  $T$  werden gebunden;  
freie Vorkommen von  $y \neq x$  in  $T$  bleiben frei, gebundene Vorkommen bleiben gebunden.
- $\langle a, b \rangle$ : freie Vorkommen von  $x$  in  $a$  und  $b$  bleiben frei, gebundene bleiben gebunden.
- $\mathbf{let} \langle x, y \rangle = p \mathbf{in} t$ : freie Vorkommen von  $x/y$  in  $t$  werden gebunden; freie Vorkommen von  $z \neq x/y$  in  $t$  und jeder Variablen  $z'$  in  $p$  bleiben frei, gebundene bleiben gebunden.
- $s=t \in T$ : freie Vorkommen von  $x$  bleiben frei, gebundene Vorkommen bleiben gebunden.
- $\mathbb{U}_i$ :  $x$  kommt nicht vor.

Viele Definitionen für nahezu dasselbe – geht es nicht einfacher?

# VORKOMMEN VON VARIABLEN: EINHEITLICHE DEFINITION

## Definition nur auf Basis der uniformen Syntax

- **var** $\{x : v\}$ : die Variable  $x \in \mathcal{V}$  kommt frei vor;  
 $y \neq x$  kommt nicht vor.
- **op** $(b_1; ..; b_n)$ : freie Vorkommen von  $x$  in  $b_i$  bleiben frei,  
gebundene Vorkommen bleiben gebunden.
- **$x_1, .., x_n$** . **t**: freie und gebundene Vorkommen der  $x_i$  in  $t$   
werden gebunden;  
freie Vorkommen von  $y \neq x_i$  bleiben frei,  
gebundene Vorkommen bleiben gebunden

**$t[x_1, .., x_n]$** :  $x_1, .., x_n$  kommen frei in  $t$  vor

**t geschlossen**:  $t$  enthält keine freien Variablen

## SUBSTITUTION $a[t/x]$ : EINHEITLICHE DEFINITION

### Endliche Abbildung $\sigma$ von Variablen in Terme

- $\sigma = [t_1, \dots, t_n / x_1, \dots, x_n] \hat{=} \sigma(x_1) = t_1, \dots, \sigma(x_n) = t_n$
- $a\sigma$ : Anwendung von  $\sigma$  auf den Ausdruck  $a$

$$\begin{aligned}\text{var}\{x:v\}() [t/x] &= t \\ \text{var}\{x:v\}() [t/y] &= \text{var}\{x:v\}() && (y \neq x) \\ (op(b_1; \dots; b_n)) [t/x] &= op(b_1 [t/x]; \dots; b_n [t/x]) \\ (x_1, \dots, x_n . u) [t/x_i] &= x_1, \dots, x_n . u \\ (x_1, \dots, x_n . u) [t/y] &= x_1, \dots, x_n . u [t/y] && * \\ (x_1, \dots, x_j, \dots, x_n . u) [t/y] &= (x_1, \dots, z, \dots, x_n . u [z/x_j]) [t/y] && **\end{aligned}$$

\*:  $y$  verschieden von allen  $x_i$ ,  $y$  nicht frei in  $u$  oder keines der  $x_i$  frei in  $t$

\*\* :  $y$  verschieden von allen  $x_i$ ,  $y$  frei in  $u$ ,  $x_j$  frei in  $t$ ,  $z$  neue Variable

# ENTWURFSENTSCHEIDUNGEN FÜR CTT (3): SEMANTIK

- **Einbettung in Mengentheorie ist Standard**
  - + : Abbildung auf etablierte Theorie
  - + : Gewohnte Technik mit Interpretationen und Modellen
  - : Mengentheorie ist hochgradig nichtkonstruktiv
    - Keine Unterscheidung verschiedener Beschreibungen eines Objektes
- **Direkte Semantik ist intuitiver** ✓
  - + : **Selbsttragend** – keine Abstützung auf andere Theorien erforderlich
  - + : Semantik kann **konstruktiv (beweisbasiert)** formuliert werden
    - Eigenschaften von Werten werden direkt als Urteile formuliert
  - : Ist es sinnvoll, eine **Konkurrenz zur Mengentheorie** aufzubauen?
- **Typentheorie ist eine Grundlagentheorie**
  - Abstützung der Semantik “in sich selbst” ist intuitiv plausibel
  - Nur die Widerspruchsfreiheit (Konsistenz) kann nachgewiesen werden

# FORMULIERUNG EINER DIREKTEN SEMANTIK FÜR CTT

## ● **Ausdrücke haben Bedeutung**

- Sie beschreiben Objekte, Funktionen, Datentypen, Aussagen, ...
  - Ausdrücke wie  $0$ ,  $\langle 0, s(0) \rangle$ ,  $\lambda x.x$  haben intuitive Bedeutung
  - Andere müssen erst “ausgerechnet” werden
- Bedeutung muß mathematisch präzisiert werden
- Konsistenz der formalen Semantik muß sichergestellt werden

## ● **Bestimme den Wert von Ausdrücken**

- Unterteile Ausdrücke in kanonische und nichtkanonische Terme
- Reduziere Ausdrücke auf kanonische Terme (Werte)

## ● **Beschreibe Eigenschaften von Werten als Urteile**

- Urteile legen fest, wann bestimmte Grundwahrheiten gelten
- Hypothetische Urteile erklären, was aus bestimmten Annahmen folgt

## ● **Entwerfe systematische Beschreibung**

- Beschreibe Reduktion durch Grundalgorithmus und Tabelleneinträge
- Beschreibe Urteile durch Iteration von Reduktion und Tabellensuche

# STARKE NORMALISIERBARKEIT IST NICHT MÖGLICH

- **Ein leerer Datentyp kann formuliert werden**

- **Void**  $\equiv \mathbf{x} : \mathbb{U}_1 \rightarrow (\mathbf{x} = \mathbf{x} \rightarrow \mathbf{x} \in \mathbb{U}_1)$  kann keine Elemente haben

- **Nicht normalisierende Terme werden typisierbar**

*Für jeden Typ  $T$  liegt  $\lambda z.(\lambda x. x x)(\lambda x. x x)$  in  $\mathbf{Void} \rightarrow T$*

- Sei  $z$  eine Eingabe aus  $\mathbf{Void} = \mathbf{x} : \mathbb{U}_1 \rightarrow (\mathbf{x} = \mathbf{x} \rightarrow \mathbf{x} \in \mathbb{U}_1)$

- Dann ist  $z(T)$  ein Beweis für (Element von)  $T = T \rightarrow T \in \mathbb{U}_1$

- Also kann  $x$  gleichzeitig zu  $T \rightarrow T$  und zu  $T$  gehören

- Damit  $x x \in T$  und  $\lambda x. x x \in T \rightarrow T = T$  also  $(\lambda x. x x)(\lambda x. x x) \in T$

- **Begriff der Normalisierung muß geändert werden**

- $\lambda z.(\lambda x. x x)(\lambda x. x x)$  hat keine Normalform im bisherigen Sinn

- Weder starke noch schwache Normalisierung gilt

- $\lambda z.(\lambda x. x x)(\lambda x. x x)$  stellt dennoch eine “echte Funktion” dar

- Für jedes  $z \in \mathbf{Void}$  bestimmt sie einen Wert aus  $T$

- Da es kein  $z \in \mathbf{Void}$  gibt, muß  $(\lambda x. x x)(\lambda x. x x)$  nie berechnet werden

- $\lambda z.(\lambda x. x x)(\lambda x. x x)$  terminiert, wenn man nur ‘bei Bedarf’ reduziert



# AUSWERTUNG VON AUSDRÜCKEN

- **Extensionale Gleichheit macht SN unmöglich**

- Extensionale Gleichheit ist für Anwendungen sehr wichtig
- Starke und schwache Normalisierbarkeit sind nicht wirklich erforderlich
- Es reicht, Ausdrücke nur “bei Bedarf” auszuwerten
  - Ein Funktionskörper wird bei Anwendung der Funktion ausgerechnet
  - Komponenten eines Paares werden erst bei einem Zugriff bestimmt

- **Der Begriff des Wertes wird abgeschwächt**

- Die äußere Form entscheidet, ob ein Term als Wert angesehen wird
- Innerhalb des Terms können durchaus noch Redizes vorkommen

- **Auswertung folgt einer “lässigen” Strategie**

- Die Auswertung endet, wenn die äußere Form einem Wert entspricht
- Reduzierbare innere Teilterme werden nur bei Bedarf ausgewertet

**Die Semantik von CTT basiert auf Lazy Evaluation**

# AUSWERTUNG VON AUSDRÜCKEN

## ● Definiere **kanonische Terme**

- $opid\{P_1, \dots, P_n\}(b_1, \dots, b_n)$  **kanonisch**, falls  $opid$  in  $\rightarrow$  *Operatorentabelle* als kanonisch gekennzeichnet (**var** muß kanonisch sein)

## ● Definiere **Hauptargumente**

- Spezielle Teilterme eines Terms  $t$  werden in der  $\rightarrow$  *Operatorentabelle* als **Hauptargumente** (*prinzipielle Argumente*) gekennzeichnet

## ● Definiere **Reduzierbarkeit**

- **Redex**: spezieller nichtkanonischer Term  $t$ , mit allen Hauptargumenten in kanonischer Form,  $t$  eingetragen in  $\rightarrow$  *Redex-Kontrakta Tabelle*
- **Kontraktum von  $t$** : zugehöriger Eintrag in Redex-Kontrakta Tabelle
- **$t$  reduzierbar auf  $u$**  ( $t \xrightarrow{\beta} u$ ), falls  $u$  Kontraktum von  $t$

## ● Definiere **Wert von Termen**

- $t$  ist ein **Wert**, wenn  $t$  geschlossener kanonischer Term
- **$u$  ist Wert eines geschlossenen Terms  $t$**  ( $t \xrightarrow{l} u$ ), falls  $u$  Ergebnis der lässigen Auswertung auf  $t$  ist  $\rightarrow$  **EVAL**

# OPERATORENTABELLE MIT KENNZEICHNUNG (AUSZUG)

| <i>Operator und Termstruktur</i>                                           | <i>Standard-Darstellungsform</i>                                                                 | <i>kanonisch?</i> |
|----------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------|-------------------|
| <b>var</b> { <i>x</i> : <i>v</i> }()                                       | <i>x</i>                                                                                         | ja                |
| <b>Nat</b> {}                                                              | $\mathbb{N}$                                                                                     | ja                |
| <b>zero</b> {}                                                             | 0                                                                                                | ja                |
| <b>suc</b> {}( <i>t</i> )                                                  | <i>s</i> ( <i>t</i> )                                                                            | ja                |
| <b>p_rec</b> {}( $\boxed{t}$ ; <i>f</i> ; <i>n</i> , <i>x</i> . <i>g</i> ) | $\text{PR}[\mathbf{f}, \mathbf{n}, \mathbf{x} \mapsto \mathbf{g}](\boxed{t})$                    | nein              |
| <b>function</b> {}( <i>S</i> ; <i>x</i> . <i>T</i> )                       | $\mathbf{x} : \mathbf{S} \rightarrow \mathbf{T}$                                                 | ja                |
| <b>lambda</b> {}( <i>x</i> . <i>t</i> )                                    | $\lambda \mathbf{x} . \mathbf{t}$                                                                | ja                |
| <b>apply</b> {}( $\boxed{f}$ ; <i>t</i> )                                  | $\boxed{\mathbf{f}} \mathbf{t}$                                                                  | nein              |
| <b>product</b> {}( <i>S</i> ; <i>x</i> . <i>T</i> )                        | $\mathbf{x} : \mathbf{S} \times \mathbf{T}$                                                      | ja                |
| <b>pair</b> {}( <i>s</i> , <i>t</i> )                                      | $\langle \mathbf{s}, \mathbf{t} \rangle$                                                         | ja                |
| <b>spread</b> {}( $\boxed{e}$ ; <i>x</i> , <i>y</i> . <i>u</i> )           | $\text{let } \langle \mathbf{x}, \mathbf{y} \rangle = \boxed{\mathbf{e}} \text{ in } \mathbf{u}$ | nein              |
| <b>equal</b> {}( <i>s</i> ; <i>t</i> ; <i>T</i> )                          | $\mathbf{s} = \mathbf{t} \in \mathbf{T}$                                                         | ja                |
| <b>Axiom</b> {}                                                            | Ax                                                                                               | ja                |
| <b>universe</b> { <i>j</i> :1}()                                           | $\mathbb{U}_{\mathbf{j}}$                                                                        | ja                |

*Hauptargumente nichtkanonischer Terme sind umrandet*

# REDEX–KONTRAKTA TABELLE (AUSZUG)

| Redex                                                                        |                       | Kontraktum                                         |
|------------------------------------------------------------------------------|-----------------------|----------------------------------------------------|
| $\mathbf{apply}\{\}(\mathbf{lambda}\{\}(x.u);t)$                             | $\xrightarrow{\beta}$ | $u[t/x]$                                           |
| $\mathbf{spread}\{\}(\mathbf{pair}\{\}(s,t);x,y.u)$                          | $\xrightarrow{\beta}$ | $u[s,t/x,y]$                                       |
| $\mathbf{p\_rec}\{\}(\mathbf{zero}\{\}();f;n,x.g)$                           | $\xrightarrow{\beta}$ | $f$                                                |
| $\mathbf{p\_rec}\{\}(\mathbf{suc}\{\}(t);f;n,x.g)$                           | $\xrightarrow{\beta}$ | $g[t, \mathbf{p\_rec}\{\}(t;f;n,x.g) / n, x]$      |
| $\vdots$                                                                     | $\vdots$              | $\vdots$                                           |
| $(\lambda x.u) \ t$                                                          | $\xrightarrow{\beta}$ | $u[t/x]$                                           |
| $\mathbf{let} \ \langle x,y \rangle = \langle s,t \rangle \ \mathbf{in} \ u$ | $\xrightarrow{\beta}$ | $u[s,t/x,y]$                                       |
| $\mathbf{PR}[f, \ n, x \mapsto g](0)$                                        | $\xrightarrow{\beta}$ | $f$                                                |
| $\mathbf{PR}[f, \ n, x \mapsto g](\mathbf{s}(t))$                            | $\xrightarrow{\beta}$ | $g[t, \mathbf{PR}[f, \ n, x \mapsto g](t) / n, x]$ |
| $\vdots$                                                                     | $\vdots$              | $\vdots$                                           |

## ALGORITHMUS EVAL( $t$ ) FÜR LÄSSIGE AUSWERTUNG

- Falls  $t$  kanonisch  $\rightarrow$  Ausgabe  $t$
- Falls  $t$  nichtkanonisch:
  - Bestimme Hauptargumente  $t_1, ..t_n$  von  $t$
  - Berechne  $s_1 := \text{EVAL}(t_1), \dots, s_n := \text{EVAL}(t_n)$
  - Bestimme  $s := t[s_1, ..s_n / t_1, ..t_n]$  \*
  - Falls  $s$  ein Redex ist:
    - Bestimme das zugehörige Kontraktum  $u$
    - Berechne  $v := \text{EVAL}(u) \rightarrow$  Ausgabe  $v$
  - Falls  $s$  kein Redex ist:
    - Stoppe ohne Ergebnis:  $t$  besitzt keinen Wert

### Auswertungsbeispiel

–  $\text{EVAL}((\lambda x. \langle (\lambda y. s(y)) x, 0 \rangle) 0)$

$= \langle (\lambda y. s(y)) 0, 0 \rangle$

Endergebnis hat inneren Redex

*\*:  $t[s_1, ..t_n / s_1, ..t_n]$  ist hier Ersetzung von Termen (Syntaxbäumen) durch Terme*

## Beschreibe semantische Eigenschaften durch Urteile

### Vier Klassen von Eigenschaften sind grundlegend

#### ● Typzugehörigkeit

geschrieben als  $t \in T$

- Ausdruck  $t$  beschreibt ein Element des durch  $T$  bezeichneten Typs
- Alternative Lesart:  $t$  hat die durch  $T$  beschriebene Eigenschaft

#### ● Elementgleichheit

geschrieben als  $s = t \in T$

- Ausdrücke  $s$  und  $t$  beschreiben dasselbe Element des durch  $T$  bezeichneten Typs

#### ● Typ-Sein

geschrieben als  $T \text{ Typ}$

- Ausdruck  $T$  bezeichnet einen Typ (Menge)

#### ● Typgleichheit

geschrieben als  $S = T$

- Ausdrücke  $S$  und  $T$  bezeichnen den gleichen Typ

### Semantische Gleichheit ist mehr als Identität der Werte

- Präzisierung der Urteile muß tiefer gehen als lässige Auswertung

## Iteration von Reduktion und Tabellensuche

- **Typzugehörigkeit**

- $t \in T$ , falls  $t = t \in T$

- **Elementgleichheit**

- $s = t \in T$ , falls es kanonische Terme  $s'$ ,  $t'$  und  $T'$  gibt mit  $s \xrightarrow{l} s'$ ,  
 $t \xrightarrow{l} t'$ ,  $T = T'$  und  $s' = t' \in T'$  folgt aus  $\rightarrow$  *Elementsemantiktabelle*

- **Typ-Sein**

- $T \text{ Typ}$ , falls  $T = T$

- **Typgleichheit**

- $S = T$ , falls es kanonische Terme  $S'$  und  $T'$  gibt mit  $S \xrightarrow{l} S'$ ,  
 $T \xrightarrow{l} T'$  und  $S' = T'$  folgt aus  $\rightarrow$  *Typsemantiktabelle*

Semantiktabellen erklären Bedeutung konkreter kanonischer Ausdrücke

# TYPSEMANTIKTABELLE (AUSZUG)

|                                                     |                                                                                                      |
|-----------------------------------------------------|------------------------------------------------------------------------------------------------------|
| $\mathbb{N} = \mathbb{N}$                           | gilt immer                                                                                           |
| $x_1:S_1 \rightarrow T_1 = x_2:S_2 \rightarrow T_2$ | falls $S_1=S_2$ und $T_1[s_1/x_1]=T_2[s_2/x_2]$<br>für alle Terme $s_1, s_2$ mit $s_1=s_2 \in S_1$ . |
| $T = S_2 \rightarrow T_2$                           | falls $T = x_2:S_2 \rightarrow T_2$ für ein beliebiges $x_2 \in \mathcal{V}$ .                       |
| $S_1 \rightarrow T_1 = T$                           | falls $x_1:S_1 \rightarrow T_1 = T$ für ein beliebiges $x_1 \in \mathcal{V}$ .                       |
| $x_1:S_1 \times T_1 = x_2:S_2 \times T_2$           | falls $S_1=S_2$ und $T_1[s_1/x_1]=T_2[s_2/x_2]$<br>für alle Terme $s_1, s_2$ mit $s_1=s_2 \in S_1$ . |
| $T = S_2 \times T_2$                                | falls $T = x_2:S_2 \times T_2$ für ein beliebiges $x_2 \in \mathcal{V}$ .                            |
| $S_1 \times T_1 = T$                                | falls $x_1:S_1 \times T_1 = T$ für ein beliebiges $x_1 \in \mathcal{V}$ .                            |
| $s_1 = t_1 \in T_1 = s_2 = t_2 \in T_2$             | falls $T_1=T_2$ und $s_1=s_2 \in T_1$ und $t_1=t_2 \in T_1$                                          |
| $\mathbb{U}_{j_1} = \mathbb{U}_{j_2}$               | falls $j_1=j_2$ (als natürliche Zahl)                                                                |



# ELEMENTSEMANTIKTABELLE (AUSZUG)

|                                                                        |                                                                                                                                              |
|------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------|
| $\vdots$                                                               | $\vdots$                                                                                                                                     |
| $0 = 0 \in \mathbb{N}$                                                 | gilt immer                                                                                                                                   |
| $\mathbf{s}(t_1) = \mathbf{s}(t_2) \in \mathbb{N}$                     | falls $t_1 = t_2 \in \mathbb{N}$                                                                                                             |
| $\lambda x_1. t_1 = \lambda x_2. t_2 \in x:S \rightarrow T$            | falls $x:S \rightarrow T$ Typ und $t_1[s_1/x_1] = t_2[s_2/x_2] \in T[s_1/x]$<br>für alle Terme $s_1, s_2$ mit $s_1 = s_2 \in S$ .            |
| $\langle s_1, t_1 \rangle = \langle s_2, t_2 \rangle \in x:S \times T$ | falls $x:S \times T$ Typ und $s_1 = s_2 \in S$ und $t_1 = t_2 \in T[s_1/x]$ .                                                                |
| $\mathbf{Ax} = \mathbf{Ax} \in s = t \in T$                            | falls $s = t \in T$                                                                                                                          |
| $\vdots$                                                               | $\vdots$                                                                                                                                     |
| $\mathbb{N} = \mathbb{N} \in \mathbb{U}_j$                             | gilt immer                                                                                                                                   |
| $x_1:S_1 \rightarrow T_1 = x_2:S_2 \rightarrow T_2 \in \mathbb{U}_j$   | falls $S_1 = S_2 \in \mathbb{U}_j$ und $T_1[s_1/x_1] = T_2[s_2/x_2] \in \mathbb{U}_j$<br>für alle Terme $s_1, s_2$ mit $s_1 = s_2 \in S_1$ . |
| $x_1:S_1 \times T_1 = x_2:S_2 \times T_2 \in \mathbb{U}_j$             | falls $S_1 = S_2 \in \mathbb{U}_j$ und $T_1[s_1/x_1] = T_2[s_2/x_2] \in \mathbb{U}_j$<br>für alle Terme $s_1, s_2$ mit $s_1 = s_2 \in S_1$ . |
| $T = S_2 \rightarrow T_2 \in \mathbb{U}_j$                             | falls $T = x_2:S_2 \rightarrow T_2 \in \mathbb{U}_j$ für ein beliebiges $x_2 \in \mathcal{V}$ .                                              |
| $S_1 \rightarrow T_1 = T \in \mathbb{U}_j$                             | falls $x_1:S_1 \rightarrow T_1 = T \in \mathbb{U}_j$ für ein beliebiges $x_1 \in \mathcal{V}$ .                                              |

# SEMANTISCHE UNTERSUCHUNG VON EIGENSCHAFTEN

**Zeige**  $(\lambda x. \langle (\lambda y. \mathbf{s}(y)) x, 0 \rangle) 0$

$= \text{let } \langle x, y \rangle = \langle 0, \mathbf{s}(0) \rangle \text{ in } \langle y, x * y \rangle \in \mathbb{N} \times \mathbb{N}$

## ● Auswertung der beteiligten Terme

$$\begin{array}{ll} - (\lambda x. \langle (\lambda y. \mathbf{s}(y)) x, 0 \rangle) 0 & \xrightarrow{l} \langle (\lambda y. \mathbf{s}(y)) 0, 0 \rangle \\ - \text{let } \langle x, y \rangle = \langle 0, \mathbf{s}(0) \rangle \text{ in } \langle y, x * y \rangle & \xrightarrow{l} \langle \mathbf{s}(0), 0 * \mathbf{s}(0) \rangle \\ - \mathbb{N} \times \mathbb{N} & \xrightarrow{l} \mathbb{N} \times \mathbb{N} \end{array}$$

## ● Semantiktabelle ergibt drei Bedingungen

1.  $(\lambda y. \mathbf{s}(y)) 0 = \mathbf{s}(0) \in \mathbb{N}$

Nach Auswertung  $\mathbf{s}(0) = \mathbf{s}(0) \in \mathbb{N}$

Bedingung laut Tabelle  $0 = 0 \in \mathbb{N}$  erfüllt, laut Tabelle

2.  $0 = 0 * \mathbf{s}(0) \in \mathbb{N}$

Nach Auswertung  $0 = 0 \in \mathbb{N}$  erfüllt, laut Tabelle

3.  $\mathbb{N} \times \mathbb{N}$  Typ

Bedingung laut Tabelle  $\mathbb{N}$  Typ (zweimal) erfüllt, laut Tabelle

# SEMANTISCHE UNTERSUCHUNG VON EIGENSCHAFTEN II

**Zeige**  $\lambda f . \lambda x . f x \in (S \rightarrow T) \rightarrow S \rightarrow T$  für beliebige  $S, T$

## Analysiere Bedingungen der Semantiktabelle

1.  $\lambda f . \lambda x . f x$  ist kanonisch und es gilt

$$\lambda f . \lambda x . f x \in (S \rightarrow T) \rightarrow S \rightarrow T$$

falls  $\lambda x . g x \in S \rightarrow T$  für alle Terme  $g \in S \rightarrow T$

2.  $\lambda x . g x$  ist kanonisch und es gilt

$$\lambda x . g x \in S \rightarrow T \text{ falls } g s \in T \text{ für alle Terme } g \in S \rightarrow T, s \in S$$

3.  $g s$  ist nicht kanonisch und kann nicht reduziert werden

aber  $g \in S \rightarrow T$  gilt nur, wenn  $g$  zu einem kanonischen  $\lambda y . t$  reduziert  
und  $t[s/y] \in T$  für alle Terme  $s \in S$  gilt

4. Es folgt  $g s \xrightarrow{l} (\lambda y . t) s \xrightarrow{l} t[s/y] \in T$  für alle Terme  $s \in S$  ✓

# SEMANTISCHE UNTERSUCHUNG VON EIGENSCHAFTEN III

**Zeige**  $\lambda x. x x \notin S \rightarrow T$  für jeden (kanonischen) Typ  $S, T$

## Analysiere Bedingungen der Semantiktabelle

1.  $\lambda x. x x$  ist kanonisch und es gilt

$\lambda x. x x \in S \rightarrow T$  falls  $s s \in T$  für alle Terme  $s \in S$

2.  $s s$  ist nicht kanonisch und  $s$  kann nur zu  $\lambda y. t$  reduziert werden, wenn

$S = Y \rightarrow Z$  für Typen  $Y, Z$  ist und  $t[s/y] \in Z$  für alle  $s \in Y$  gilt

In dem Fall folgt  $s s \xrightarrow{l} (\lambda y. t) s \xrightarrow{l} t[s/y] \in Z$  falls  $s \in Y$

3. Damit muß  $Z = T$  und  $S = Y$ , also  $S = S \rightarrow T$  sein

Für die Behauptung  $S = S \rightarrow T$  gibt es **keine Unterstützung**

4. Das Urteil  $s s \in T$  kann nicht gefällt werden

Die Sprechweise “*Das Urteil ist nicht gültig*” sollte vermieden werden

5.  $\lambda x. x x \in S \rightarrow T$  **gilt nicht**



## GENERISCHE EIGENSCHAFTEN VON URTEILEN

- **Typgleichheit**  $S=T$  ist transitiv und symmetrisch
  - aber nicht reflexiv ( $S$  und  $T$  müssen Typausdrücke sein)
- $S=T$  gilt genau dann, wenn es ein  $S'$  gibt mit  $S \xrightarrow{l} S'$  und  $S'=T$
- **Elementgleichheit**  $s=t \in T$  ist transitiv und symmetrisch in  $s$  und  $t$ 
  - aber nicht reflexiv ( $s$  und  $t$  müssen Elemente von  $T$  sein)
- $s=t \in T$  gilt genau dann, wenn es ein  $s'$  gibt mit  $s \xrightarrow{l} s'$  und  $s'=t \in T$
- Wenn  $s=t \in T$  gilt dann ist  $T$  ein Typ ( $T=T$ )
- Wenn  $s=t \in T$  und  $S=T$  gilt, dann auch  $s=t \in S$

Eigenschaften folgen unmittelbar aus Definition + Tabellen

# EIGENSCHAFTEN VON CTT TERMEN

## ● Normalisierbarkeit

- CTT Terme sind i.a. weder stark noch schwach normalisierbar
- Auswertung auf Basis von Lazy Evaluation ist hinreichend

## ● Konfluenz

- Nicht erforderlich, da Auswertungsstrategie festgelegt
- Für stark normalisierende Terme gilt dennoch Konfluenz

## ● Typisierbarkeit

- Semantik legt eindeutig fest, wann  $t \in T$  gilt
- Typisierbare Terme können nichttypisierbare Teilterme enthalten
  - Semantik für  $\lambda x.t \in \text{Void} \rightarrow T$  stellt keine “echten” Bedingungen an  $t$
  - Ausdrücke mit **Y**-Kombinator sind typisierbar, wenn sie terminieren
- Typzugehörigkeit und Typsein ist i.a. nicht entscheidbar
  - Objekt- und Typausdrücke können nichtterminierende Terme sein
  - Halteproblem wird Teil des Typisierbarkeitsproblems
- Typzugehörigkeit und Typsein muß im Beweiskalkül geprüft werden
  - unvermeidbare Konsequenz von extensionaler Gleichheit

## Fundamentales Konzept für semantische Argumente

- **Urteil unter Annahmen:**  $x_1:T_1, \dots, x_n:T_n \models J$ 
  - Das Urteil  $J$  wird gefällt unter der Annahme, daß die  $x_i$  Variablen vom Typ  $T_i$  sind
  - Nicht identisch mit “ $J$  gilt, wenn  $x_i$  Variablen vom Typ  $T_i$  sind”
  - Die Gültigkeit der Annahmen wird vorausgesetzt
- **Konzept beinhaltet**
  - **Funktionalität:**  $J[t_i/x_i]$  gilt für alle kanonischen Elemente  $t_i \in T_i$
  - **Extensionalität:** falls  $t_i = t'_i \in T_i$ , dann gilt  $J[t_i/x_i]$  genau dann, wenn  $J[t'_i/x_i]$  gilt ( $t_i$  und  $t'_i$  auch nichtkanonisch)
- **Formale Definition aufwendig**

- **Syntaktische Simulation logischer Schlüsse**
  - Formaler Nachweis, daß bestimmte Urteile gefällt werden können
  - Inferenzregeln ersetzen semantische Argumente auf Basis der Tabellen
- **Beweise operieren zielorientiert**
  - Oberstes Beweisziel ist die zu beweisende Behauptung (“*Urteil J ‘gilt’*”)
  - Inferenzregeln beweisen ein Ziel durch *Verfeinerung* in Teilziele, deren Gültigkeit hinreichend für die Gültigkeit des Ziels ist
  - Sinnvoller für computergestützte Beweisführung als synthetisches Vorgehen
- **Beweisziele werden als Sequenzen beschrieben**
  - Syntaktisches Gegenstück zu hypothetischen Urteilen  
Schreibweise  $x_1:T_1, \dots x_n:T_n \vdash C$
  - Urteile werden durch Sequenzen ohne Hypothesen beschrieben  
Oberstes Beweisziel hat die Form  $\vdash C$
  - Sequenzen stellen nur Behauptungen dar  
Erst der Beweis zeigt, daß das entsprechende Urteil gefällt werden kann



- **Nur eine syntaktische Klasse für Konklusionen**
  - Vermeidet Aufblähung des syntaktischen Apparates
  - Benötigt einheitliche Repräsentation der vier Urteilklassen
- **Universen repräsentieren Typ-Eigenschaft**
  - Urteile  $T \text{ Typ}$  und  $S = T$  werden repräsentiert als  $T \in \mathbb{U}_i$  und  $S = T \in \mathbb{U}_i$
  - Kumulative Universenhierarchie  $\mathbb{U} = \mathbb{U}_1 \subseteq \mathbb{U}_2 \subseteq \mathbb{U}_3 \subseteq \dots$  erforderlich
- **Gleichheit wird dargestellt als Datentyp  $s = t \in T$** 
  - Ermöglicht Verwendung von Gleichheiten in Hypothesen
    - z.B.  $h_1 : x = 4 \in \mathbb{N}, h_2 : f(4) = 18 \in \mathbb{N} \vdash f(x) = 18 \in \mathbb{N}$
  - Elemente des Typs repräsentieren Beweise der Gleichheit
    - $\mapsto$  **Propositionen werden als Datentypen beschrieben**
- **Grundform der Konklusion ist Typzugehörigkeit**
  - Urteile  $t \in T$  und  $s = t \in T$  repräsentiert als  $t \in T$  und  $\Lambda x \in s = t \in T$

## Prüfender oder konstruierender Kalkül?

- **Überprüfung (Verifikation) von Typzugehörigkeit**

Typ  $T$  und Term  $t$  sind vorgegeben, Beweis prüft  $t \in T$

+: Geringere Beweislast, da viel Information vorgegeben

–: Wegen des Prinzips “Propositionen sind Datentypen” muß beim Beweis logischer Aussagen ein Beweisterm im Voraus angegeben werden

- **Konstruktion (Synthese) von Elementen**

Typ  $T$  ist vorgegeben, Term  $t$  mit  $t \in T$  wird im Beweis konstruiert

+: Ermöglicht Erzeugung korrekter Programme durch Beweisführung

Beweis für  $x:\mathbb{N} \rightarrow \{y:\mathbb{N} \mid y^2 \leq x < (y+1)^2\}$  enthält Algorithmus für  $\sqrt{x}$

–: Reine Verifikation nicht möglich

- **Implizite Darstellung von Elementen**



– Ersetze  $t \in T$  durch  $\mathbf{T} \text{ [ext } t]$  ‘ $T$  hat ein (noch unbekanntes) Element  $t$ ’

– Synthese: Extrakt-Term  $t$  kann nach Beweisführung extrahiert werden

– Explizite Verifikation möglich durch Beweis von  $t = t \in T \text{ [ext } Ax]$

## ● **Verifikation:** Überprüfung expliziter Urteile

- *Typ-Sein* “ $T \text{ Typ}$ ”:  $\text{zeige} \vdash T = T \in \mathbb{U}_j \text{ [ext } p]$   
(Zeige Existenz eines Terms  $p$  und eines Levels  $j$  mit  $p \in T = T \in \mathbb{U}_j$ )
- *Typgleichheit* “ $S = T$ ”:  $\text{zeige} \vdash S = T \in \mathbb{U}_j \text{ [ext } p]$
- *Typzugehörigkeit* “ $t \in T$ ”:  $\text{zeige} \vdash t = t \in T \text{ [ext } p]$
- *Elementgleichheit* “ $s = t \in T$ ”:  $\text{zeige} \vdash s = t \in T \text{ [ext } p]$

## ● **Synthese:** Konstruktion von Termen

- *Konstruktion von Datentypen:*  $\text{zeige} \vdash \mathbb{U}_j \text{ [ext } T]$
- *Konstruktion von Elementen eines Typs:*  $\text{zeige} \vdash T \text{ [ext } t]$   
(Zeige Existenz eines Terms  $t$  mit  $t \in T$ )

# BEWEISKALKÜL: PRÄZISIERUNG DER KONZEPTE (I)

- **Sequenz:**  $x_1:T_1, \dots, x_n:T_n \vdash C$

“Unter der Annahme, daß  $x_i$  Variablen vom Typ  $T_i$  sind, gilt  $C$ ”

- $x_i:T_i$  **Deklaration** “ $x_i$  ist Variable vom Typ  $T_i$ ” ( $x_i$  Variable,  $T_i$  Term)

- $\Gamma \equiv x_1:T_1, \dots, x_n:T_n$  **Hypothesenliste**

- $C$  **Konklusion** ( $C$  Term)

- **Geschlossene Sequenz**  $\Gamma \vdash C$

- Jede freie Variable  $x$  in  $C$  oder  $T_i$  zuvor durch  $x:T_j$  deklariert

- **Reine Sequenz**  $\Gamma \vdash C$

- Geschlossene Sequenz, in der jede Variable nur einmal deklariert ist

- **Gültige Sequenz**  $\Gamma \vdash C$

- Es gibt einen Term  $t$ , so daß  $\Gamma \models t \in C$  hypothetisches Urteil

- Notation  $\Gamma \vdash C \text{ [ext } t]$ , falls  $t$  bekannt

**Definition direkt umsetzbar in Implementierung**

# IMPLEMENTIERUNG VON SEQUENZEN

Struktur:  $x_1:T_1, \dots, x_n:T_n \vdash C$

|                           |                 |
|---------------------------|-----------------|
| $x_i$                     | Variable,       |
| $T_i, C$                  | Term            |
| $x_i:T_i$                 | Deklaration     |
| $x_1:T_1, \dots, x_n:T_n$ | Hypothesenliste |
| $C$                       | Konklusion      |

```
abstype declaration = var # term # bool
  with mk_declaration v t b = abs_declaration(v,t,b)
  and dest_declaration d = rep_declaration d
;;
lettype sequent = declaration list # term;;
  let mk_sequent vtb_list term = map mk_declaration term
  and dest_sequent seq = map dest_declaration (fst seq), snd seq
;;
```

In Nuprl werden Sequenzen mit Beweisen gleichgesetzt

## ● Verfeinerung des Beweisziels

$\Gamma \vdash \lambda x_1. t_1 = \lambda x_2. t_2 \in x:S \rightarrow T$

$\lambda x_1. t_1 = \lambda x_2. t_2 \in x:S \rightarrow T$

by **lambdaEquality**  $j$

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

falls  $x:S \rightarrow T$  Typ und

$\Gamma, x':S \vdash t_1[x'/x_1] = t_2[x'/x_2] \in T[x'/x]$

$t_1[s_1/x_1] = t_2[s_2/x_2] \in T[s_1/x]$

für alle Terme  $s_1, s_2$  mit  $s_1 = s_2 \in S$

- Regel spiegelt Tabelleneintrag in vereinfachter Form wieder
  - $x:S \rightarrow T$  Typ falls  $S$  Typ und  $T[s_1/x]$  Typ für alle  $s_1 \in S$
  - Letzteres ist auch Bedingung für  $t_1[s_1/x_1] = t_2[s_2/x_2] \in T[s_1/x]$
- $S$  Typ ist unvollständig repräsentiert durch Wahl des Universums

## ● Aufbau von Evidenz

$x:S \rightarrow T$  [ext  $\lambda x'. t_j$

by **lambdaFormation**  $j$

$\Gamma, x':S \vdash T[x'/x]$  [ext  $t_j$

$\Gamma \vdash S \in \mathbb{U}_j$  [Ax]

- Evidenz für Teilziele wird zu Evidenz für Hauptziel zusammengesetzt
- Evidenz für Verifikationen hat keine konstruktive Bedeutung

# BEWEISKALKÜL: PRÄZISIERUNG DER KONZEPTE (II)

- **Dekomposition**

- Abbildung von Sequenzen in endliche Listen von Sequenzen
- Verfeinerung eines Beweisziels in Teilziele

- **Validierung**

- Abbildung von endlichen Listen von Termen und Sequenzen in Terme
- Konstruktion von Evidenz für Beweisziel aus Evidenzen für Teilziele

- **Inferenzregel**

- Paar  $r = (\text{dec}, \text{val})$  bestehend aus Dekomposition + Validierung
- Darstellung als schematisches Objekt mit Meta-Variablen

- **Korrektheit einer Inferenzregel  $r = (\text{dec}, \text{val})$**

Für jede reine Sequenz  $Z = \Gamma \vdash C$  gilt

- Ist  $\text{dec}(Z) = [Z_1, \dots, Z_n]$ , dann sind alle Teilziele  $Z_i = \Gamma_i \vdash C_i$  rein
- Die Gültigkeit von  $Z$  folgt aus der Gültigkeit aller  $Z_i$
- Gilt  $\Gamma_i \vdash C \llbracket \text{ext } t_i \rrbracket$  für alle  $Z_i$ , so folgt  $\Gamma \vdash C \llbracket \text{ext } t \rrbracket$ ,  
für  $t = \text{val}([(Z_1, t_1), \dots, (Z_n, t_n)])$

# BEWEISKALKÜL: PRÄZISIERUNG DER KONZEPTE (III)

## Beweise sind markierte Inferenzbäume

- **Beweis  $\pi$  mit Wurzel  $Z$**  ( $Z = \Gamma \vdash C$  Sequenz)
  1.  $\pi = Z$  (unvollständiger Beweis) oder
  2.  $\pi = (Z, r, [\pi_1, \dots, \pi_n])$  und
    - $r = (\text{dec}, \text{val})$  korrekte Inferenzregel,
    - $\pi_1, \dots, \pi_n$  Beweise, deren Wurzeln die Teilziele von  $\text{dec}(Z)$  sind $\pi$  **vollständig**, falls  $\pi$  ohne unvollständigen Teilbeweise
- **Extrakt-Term des vollständigen Beweises  $\pi$** 
  - $\text{EXT}(Z, (\text{dec}, \text{val}), [\pi_1, \dots, \pi_n]) = \text{val}([(Z_1, \text{EXT}(\pi_1)), \dots, (Z_n, \text{EXT}(\pi_n))])$   
( $Z_i$  Wurzel von  $\pi_i$ )
- **Initialsequenz**
  - Geschlossene Sequenz der Form  $\vdash C$
- **Theorem**
  - Vollständiger Beweis mit einer Initialsequenz als Wurzel



# IMPLEMENTIERUNG VON REGELN UND BEWEISEN

Inferenzregel:  $r = (\text{dec}, \text{val})$

$\text{dec}$  Dekomposition: Abbildung von Sequenzen in Listen von Sequenzen

$\text{val}$  Validierung: Abbildung von Listen von Termen und Sequenzen in Terme

Beweis mit Wurzel  $Z$ : Sequenz  $Z$  oder Struktur  $\pi = (Z, r, [\pi_1, \dots, \pi_n])$

$Z$  Sequenz

$r$  Inferenzregel

$\pi_1, \dots, \pi_n$  Beweise, deren Wurzeln die Teilziele von  $\text{dec}(Z)$  sind

---

```
abstype rule      = .....
absrectype proof  = sequent # rule # proof list
  with make_proof_node decs t = abs_proof((decs,t),  $\diamond$ , [])
  and refine r p    = let children = deduce_children r p
                      and validation = deduce_validation r p
                      in children, validation

  and hypotheses p = fst (fst (rep_proof p))
  and conclusion p = snd (fst (rep_proof p))
  and refinement p = fst (snd (rep_proof p))
  and children    p = snd (snd (rep_proof p))
;;
lettype validation = proof list -> proof;;
lettype tactic     = proof -> (proof list # validation);;
```

# SYSTEMATIK FÜR AUFBAU DES REGELSYSTEMS

## Vier Klassen von Regeln für jeden Datentyp

### ● Typ-Formationsregeln:

- Wann sind zwei Typen gleich  $(typeEquality) \quad \Gamma \vdash S = T \in \mathbb{U}_j$
- Wie ist ein Typ zusammengesetzt?  $(typeFormation) \quad \Gamma \vdash \mathbb{U}_j \text{ [ext } T]$

### ● Kanonische Regeln:

- Wann sind kanonische Elemente gleich?  $(elementEquality) \quad \Gamma \vdash s = t \in T$
- Wie sind Elemente zusammengesetzt?  $(elementFormation) \quad \Gamma \vdash T \text{ [ext } t]$

### ● Nichtkanonische Regeln:

- Wann gehört ein Ausdruck zu einem Typ?  $(nichtkanonischEquality) \quad \Gamma \vdash s = t \in T$
- Wie kann eine Variable benutzt werden?  $(typElimination) \quad \Gamma, x:S, \Delta \vdash T \text{ [ext } t]$

### ● Berechnungsregeln:

- Reduktion von Redizes in Gleichheit  $(nichtkanonischReduce*) \quad \Gamma \vdash redex = t \in T$

### ● Spezielle Regeln

# BEISPIEL: REGELN FÜR FUNKTIONENRAUM

$\Gamma \vdash \mathbb{U}_j \text{ [ext } x:S \rightarrow T]$   
**by** dependent\_functionFormation  $x \ S$

$\Gamma \vdash S \in \mathbb{U}_j \text{ [Ax]}$   
 $\Gamma, x:S \vdash \mathbb{U}_j \text{ [ext } T]$

$\Gamma \vdash x_1:S_1 \rightarrow T_1 = x_2:S_2 \rightarrow T_2 \in \mathbb{U}_j \text{ [Ax]}$   
**by** functionEquality  $x$

$\Gamma \vdash S_1 = S_2 \in \mathbb{U}_j \text{ [Ax]}$   
 $\Gamma, x:S_1 \vdash T_1[x/x_1] = T_2[x/x_2] \in \mathbb{U}_j \text{ [Ax]}$

$\Gamma \vdash \lambda x_1.t_1 = \lambda x_2.t_2 \in x:S \rightarrow T \text{ [Ax]}$   
**by** lambdaEquality  $j \ x'$

$\Gamma, x':S \vdash t_1[x'/x_1] = t_2[x'/x_2] \in T[x'/x] \text{ [Ax]}$   
 $\Gamma \vdash S \in \mathbb{U}_j \text{ [Ax]}$

$\Gamma \vdash f_1 t_1 = f_2 t_2 \in T[t_1/x] \text{ [Ax]}$   
**by** applyEquality  $x:S \rightarrow T$

$\Gamma \vdash f_1 = f_2 \in x:S \rightarrow T \text{ [Ax]}$   
 $\Gamma \vdash t_1 = t_2 \in S \text{ [Ax]}$

$\Gamma, f:x:S \rightarrow T, \Delta \vdash C \text{ [ext } t[fs, Ax/y, z]]$   
**by** dependent\_functionElimination  $i \ s \ y \ z$

$\Gamma, f:x:S \rightarrow T, \Delta \vdash s \in S \text{ [Ax]}$   
 $\Gamma, f:x:S \rightarrow T, y:T[s/x], z:y=fs \in T[s/x], \Delta \vdash C \text{ [ext } t]$

$\Gamma \vdash (\lambda x.t) s = t_2 \in T \text{ [Ax]}$   
**by** applyReduce

$\Gamma \vdash t[s/x] = t_2 \in T \text{ [Ax]}$

$\Gamma \vdash \mathbb{U}_j \text{ [ext } S \rightarrow T]$   
**by** independent\_functionFormation

$\Gamma \vdash \mathbb{U}_j \text{ [ext } S]$   
 $\Gamma \vdash \mathbb{U}_j \text{ [ext } T]$

$\Gamma \vdash S_1 \rightarrow T_1 = S_2 \rightarrow T_2 \in \mathbb{U}_j \text{ [Ax]}$   
**by** independent\_functionEquality

$\Gamma \vdash S_1 = S_2 \in \mathbb{U}_j \text{ [Ax]}$   
 $\Gamma \vdash T_1 = T_2 \in \mathbb{U}_j \text{ [Ax]}$

$\Gamma \vdash x:S \rightarrow T \text{ [ext } \lambda x'.t]$   
**by** lambdaFormation  $j \ x'$

$\Gamma, x':S \vdash T[x'/x] \text{ [ext } t]$   
 $\Gamma \vdash S \in \mathbb{U}_j \text{ [Ax]}$

$\Gamma, f:S \rightarrow T, \Delta \vdash C \text{ [ext } t[fs, /y]]$   
**by** independent\_functionElimination  $i \ y$

$\Gamma, f:S \rightarrow T, \Delta \vdash S \text{ [ext } s]$   
 $\Gamma, f:S \rightarrow T, y:T, \Delta \vdash C \text{ [ext } t]$

$\Gamma \vdash f_1 = f_2 \in x:S \rightarrow T \text{ [ext } t]$   
**by** functionExtensionality  $j \ x_1:S_1 \rightarrow T_1 \ x_2:S_2 \rightarrow T_2 \ x'$

$\Gamma, x':S \vdash f_1 x' = f_2 x' \in T[x'/x] \text{ [ext } t]$   
 $\Gamma \vdash S \in \mathbb{U}_j \text{ [Ax]}$

$\Gamma \vdash f_1 \in x_1:S_1 \rightarrow T_1 \text{ [Ax]}$   
 $\Gamma \vdash f_2 \in x_2:S_2 \rightarrow T_2 \text{ [Ax]}$

# INFERENZREGELN BRAUCHEN STEUERUNGSPARAMETER

- **Position**  $i$  einer zu verwendenden Hypothese

$$\Gamma, x:T, \Delta \vdash T \text{ [ext } x] \\ \text{by hypothesis } i$$

- **Name**  $x$  einer neu zu erzeugenden Variablen

$$\Gamma \vdash x_1:S_1 \rightarrow T_1 = x_2:S_2 \rightarrow T_2 \in \mathbb{U}_j \text{ [Ax]} \\ \text{by functionEquality } x \\ \Gamma \vdash S_1 = S_2 \in \mathbb{U}_j \text{ [Ax]} \\ \Gamma, x:S_1 \vdash T_1[x/x_1] = T_2[x/x_2] \in \mathbb{U}_j \text{ [Ax]}$$

- **Level**  $j$  des Universums,  
das in einem Unterziel benötigt wird

$$\Gamma \vdash \lambda x_1. t_1 = \lambda x_2. t_2 \in x:S \rightarrow T \text{ [Ax]} \\ \text{by lambdaEquality } j \ x' \\ \Gamma, x':S \vdash t_1[x'/x_1] = t_2[x'/x_2] \in T[x'/x] \text{ [Ax]} \\ \Gamma \vdash S \in \mathbb{U}_j \text{ [Ax]}$$

# STEUERUNGSPARAMETER FÜR INFERENZREGELN (II)

- **Term**  $s$ , der für eine Variable einzusetzen ist

$$\begin{array}{l} \Gamma, f: x:S \rightarrow T, \Delta \vdash C \quad [\text{ext } t[fs, Ax/y, z]] \\ \text{by dependent\_functionElimination } i \ s \ y \ z \\ \Gamma, f: x:S \rightarrow T, \Delta \vdash s \in S \quad [Ax] \\ \Gamma, f: x:S \rightarrow T, y:T[s/x], z: y=f \ s \in T[s/x], \Delta \vdash C \quad [\text{ext } t] \end{array}$$

- **Typ**  $T$  eines Teilterms des Beweiszieles, der in einem Unterziel isoliert auftritt

$$\begin{array}{l} \Gamma \vdash f_1 t_1 = f_2 t_2 \in T[t_1/x] \quad [Ax] \\ \text{by applyEquality } x:S \rightarrow T \\ \Gamma \vdash f_1 = f_2 \in x:S \rightarrow T \quad [Ax] \\ \Gamma \vdash t_1 = t_2 \in S \quad [Ax] \end{array}$$

- **Abhängigkeit** eines (instantiierten) Terms  $C$  von einer Variablen  $x$

$$\begin{array}{l} \Gamma \vdash C[s/x] \quad [\text{ext } u] \\ \text{by substitution } j \ s=t \in T \ x \ C \\ \Gamma \vdash s=t \in T \quad [Ax] \\ \Gamma \vdash C[t/x] \quad [\text{ext } u] \\ \Gamma, x:T \vdash C \in \mathbb{U}_j \quad [Ax] \end{array}$$

## VEREINFACHUNGEN DES REGELSYSTEMS

- **Vollständiges Regelsystem wird sehr umfangreich**
  - Durchschnittlich 9 Regeln pro Datentyp
  - Namensvielfalt trotz Systematik zu groß und schwer zu merken
- **Viele Regeln sind “strukturell ähnlich”**
  - Formationsregeln zerlegen Top-Level Terme in Komponenten
  - Gleichheitsregeln zerlegen Terme in einer Gleichheit in Komponenten
  - Berechnungsregeln reduzieren ein Redex in einer Gleichung
  - Gleichartige Regeln sollten unter einem Namen zusammengefaßt werden
- **Taktiken verhindern Namensflut**
  - **D**  $i$ : Top-Level Dekomposition einer Hypothese oder der Konklusion
  - **EqD**  $i$ : Dekomposition der Elemente einer Gleichheit  $s=t \in T$
  - **EqTypeD**  $i$ : Dekomposition des Typs einer Gleichheit
  - **MemD**  $i$ : Dekomposition einer Typzugehörigkeit  $t \in T$
  - **Reduce**  $i$ : Top-Level (!) Reduktion

Taktiken bestimmen passende Inferenzregel aus dem Kontext.  
Parameter für neue Variablen, Universen, Abhängigkeiten werden z.T. automatisch bestimmt

## TAKTIKEN KÖNNEN KOMBINIERT WERDEN

$t_1$  THEN  $t_2$ : “Wende  $t_2$  auf alle von  $t_1$  erzeugten Teilziele an”

$t$  THENL  $[t_1; t_2; ..t_n]$ : “Wende  $t_i$  auf das  $i$ -te von  $t$  erzeugte Teilziel an”

$t_1$  ORELSE  $t_2$ : “Wende  $t_1$  an. Falls dies fehlschlägt, wende  $t_2$  an”.

Repeat  $t$ : “Wiederhole die Taktik  $t$  bis sie fehlschlägt”

Complete  $t$ : “Wende  $t$  nur an, wenn der Beweis vollständig wird”

Progress  $t$ : “Wende  $t$  nur an, wenn ein Fortschritt erzielt wird”

Try  $t$ : “Wende  $t$  an; falls dies fehlschlägt, lasse das Ziel unverändert”

*Mehr dazu beim Thema Automatisierung*

## ZUSAMMENFASSUNG: ENTWURFSENTSCHEIDUNGEN

- CTT wird als **erweiterbare Theorie** formalisiert
- CTT ist eine **Grundagentheorie** mit direkter Semantik
- **Abstraktion und Darstellungsform** von Termen wird getrennt
- Die Semantik von CTT basiert auf **Lazy Evaluation**
- Die Semantik von CTT wird durch **Urteile** beschrieben
- Eine **Universenhierarchie** repräsentiert die Typeigenschaft
- Gleichheit in CTT ist **extensional**
- Logische **Aussagen** werden durch **Typen** repräsentiert
- Alle Urteile werden durch Konklusionen  $T \text{ [ext } t]$  dargestellt
- Beweise werden analytisch mit **Verfeinerungsregeln** geführt



# KONSEQUENZEN DER ENTWURFSENTSCHEIDUNGEN

- **Typ- und Regelsystem ist sehr umfangreich**
  - + : Wichtige Anwenderkonzepte sind bereits vordefiniert
  - + : Vereinheitlichung der Regelnamen durch Elementartaktiken
- **Wohlgeformtheit von Termen ist unentscheidbar**
  - In der Formulierung kann ein unentscheidbares Problem enthalten sein
  - : Wohlgeformtheit und Typzugehörigkeit müssen explizit bewiesen werden
  - + : Die meisten dieser Ziele sind mit einfachen Taktiken zu beweisen
- **Nichtnormalisierende Terme können typisierbar sein**
  - + : Kontrollierte Rekursion und Y-Kombinator können verwendet werden
- **Beweisterme können extrahiert werden**
  - + : Aus Beweisen zu Programmeigenschaften sind Algorithmen extrahierbar

# LEITLINIEN FÜR ERWEITERUNGEN DES TYPSYSTEMS

- **Beschreibe abstrakte Syntax und Darstellungsform**
  - Trage Abstraktionsform in **Operatorentabelle** ein
  - Erzeuge Display-Objekt für Präsentationsform des Terms
- **Trenne kanonische und nichtkanonische Terme**
  - Typen haben üblicherweise ein kanonisches/nichtkanonisches Element
  - Erweitere **Redex-Kontrakta Tabelle** entsprechend
- **Beschreibe Semantik kanonischer Terme**
  - Vermeide Interaktionen mit Termen anderer Typen
  - Ergänze **Semantiktabellen** und prüfe Konsistenz der Ergänzung
- **Ergänze Inferenzsystem**
  - Erzeuge **Regelobjekte**, welche die Semantik widerspiegeln
- **Alternativ: konservative Erweiterung**
  - Definiere Abstraktion + Semantik durch formale Definition
  - Erzeuge Display-Objekt für Präsentationsform des Terms
  - Erzeuge Inferenztaktiken auf Basis der Regeln für bisherige Terme

# ANHANG

# ANHANG: ÜBERSICHT (ABHÄNGIGER) FUNKTIONENRAUM

## Zentrales Konzept für Schließen über Berechnung

### Syntax:

Kanonisch:  $x:S \rightarrow T$       **function**{ } ( $S$ ;  $x.T$ )

$\lambda x.e$       **lambda**{ } ( $x.e$ )

Nichtkanonisch:  $\boxed{e_1} e_2$       **apply**{ } ( $\boxed{e_1}; e_2$ )

### Auswertung:

$(\lambda x.u) t \xrightarrow{\beta} u[t/x]$

### Semantik:

$x_1:S_1 \rightarrow T_1 = x_2:S_2 \rightarrow T_2 \quad \equiv \quad S_1=S_2 \text{ und } T_1[s_1/x_1]=T_2[s_2/x_2]$   
für alle Terme  $s_1, s_2$  mit  $s_1=s_2 \in S_1$ .

$\lambda x_1.t_1 = \lambda x_2.t_2 \in x:S \rightarrow T \quad \equiv \quad x:S \rightarrow T \text{ Typ und } t_1[s_1/x_1] = t_2[s_2/x_2] \in T[s_1/x]$   
für alle Terme  $s_1, s_2$  mit  $s_1=s_2 \in S$

Inferenzregeln und Taktiken im Appendix A.3.1 des Nuprl Manuals

# ANHANG: REGELN FÜR FUNKTIONENRAUM

$\Gamma \vdash \mathbb{U}_j \text{ [ext } x:S \rightarrow T]$   
**by dependent\_functionFormation**  $x \ S$

$\Gamma \vdash S \in \mathbb{U}_j \text{ [Ax]}$   
 $\Gamma, x:S \vdash \mathbb{U}_j \text{ [ext } T]$

$\Gamma \vdash x_1:S_1 \rightarrow T_1 = x_2:S_2 \rightarrow T_2 \in \mathbb{U}_j \text{ [Ax]}$   
**by functionEquality**  $x$

$\Gamma \vdash S_1 = S_2 \in \mathbb{U}_j \text{ [Ax]}$   
 $\Gamma, x:S_1 \vdash T_1[x/x_1] = T_2[x/x_2] \in \mathbb{U}_j \text{ [Ax]}$

$\Gamma \vdash \lambda x_1.t_1 = \lambda x_2.t_2 \in x:S \rightarrow T \text{ [Ax]}$   
**by lambdaEquality**  $j \ x'$

$\Gamma, x':S \vdash t_1[x'/x_1] = t_2[x'/x_2] \in T[x'/x] \text{ [Ax]}$   
 $\Gamma \vdash S \in \mathbb{U}_j \text{ [Ax]}$

$\Gamma \vdash f_1 t_1 = f_2 t_2 \in T[t_1/x] \text{ [Ax]}$   
**by applyEquality**  $x:S \rightarrow T$

$\Gamma \vdash f_1 = f_2 \in x:S \rightarrow T \text{ [Ax]}$   
 $\Gamma \vdash t_1 = t_2 \in S \text{ [Ax]}$

$\Gamma, f:x:S \rightarrow T, \Delta \vdash C \text{ [ext } t[fs, Ax/y, z]]$   
**by dependent\_functionElimination**  $i \ s \ y \ z$

$\Gamma, f:x:S \rightarrow T, \Delta \vdash s \in S \text{ [Ax]}$   
 $\Gamma, f:x:S \rightarrow T, y:T[s/x], z:y=fs \in T[s/x], \Delta \vdash C \text{ [ext } t]$

$\Gamma \vdash (\lambda x.t) s = t_2 \in T \text{ [Ax]}$   
**by applyReduce**

$\Gamma \vdash t[s/x] = t_2 \in T \text{ [Ax]}$

$\Gamma \vdash \mathbb{U}_j \text{ [ext } S \rightarrow T]$   
**by independent\_functionFormation**

$\Gamma \vdash \mathbb{U}_j \text{ [ext } S]$   
 $\Gamma \vdash \mathbb{U}_j \text{ [ext } T]$

$\Gamma \vdash S_1 \rightarrow T_1 = S_2 \rightarrow T_2 \in \mathbb{U}_j \text{ [Ax]}$   
**by independent\_functionEquality**

$\Gamma \vdash S_1 = S_2 \in \mathbb{U}_j \text{ [Ax]}$   
 $\Gamma \vdash T_1 = T_2 \in \mathbb{U}_j \text{ [Ax]}$

$\Gamma \vdash x:S \rightarrow T \text{ [ext } \lambda x'.t]$   
**by lambdaFormation**  $j \ x'$

$\Gamma, x':S \vdash T[x'/x] \text{ [ext } t]$   
 $\Gamma \vdash S \in \mathbb{U}_j \text{ [Ax]}$

$\Gamma, f:S \rightarrow T, \Delta \vdash C \text{ [ext } t[fs, /y]]$   
**by independent\_functionElimination**  $i \ y$

$\Gamma, f:S \rightarrow T, \Delta \vdash S \text{ [ext } s]$   
 $\Gamma, f:S \rightarrow T, y:T, \Delta \vdash C \text{ [ext } t]$

$\Gamma \vdash f_1 = f_2 \in x:S \rightarrow T \text{ [ext } t]$   
**by functionExtensionality**  $j \ x_1:S_1 \rightarrow T_1 \ x_2:S_2 \rightarrow T_2 \ x'$

$\Gamma, x':S \vdash f_1 x' = f_2 x' \in T[x'/x] \text{ [ext } t]$   
 $\Gamma \vdash S \in \mathbb{U}_j \text{ [Ax]}$

$\Gamma \vdash f_1 \in x_1:S_1 \rightarrow T_1 \text{ [Ax]}$   
 $\Gamma \vdash f_2 \in x_2:S_2 \rightarrow T_2 \text{ [Ax]}$

# ÜBERSICHT (ABHÄNGIGES) KARTESISCHES PRODUKT

## Fundamentale Datenstruktur Abhängige Form für Beschreibung von Modulen u.ä.

### Syntax:

Kanonisch:  $x : S \times T$

$\langle e_1, e_2 \rangle$

Nichtkanonisch: **let**  $\langle x, y \rangle = \boxed{e}$  **in**  $u$

**product** $\{\}$   $(S; x.T)$

**pair** $\{\}$   $(e_1, e_2)$

**spread** $\{\}$   $(\boxed{e}; x, y.u)$

### Auswertung:

$\text{let } \langle x, y \rangle = \langle s, t \rangle \text{ in } u \xrightarrow{\beta} u[s, t/x, y]$

### Semantik:

$x_1 : S_1 \times T_1 = x_2 : S_2 \times T_2 \equiv S_1 = S_2 \text{ und } T_1[s_1/x_1] = T_2[s_2/x_2]$   
für alle Terme  $s_1, s_2$  mit  $s_1 = s_2 \in S_1$ .

$T = S_2 \times T_2 \equiv T = x_2 : S_2 \times T_2$  für ein beliebiges  $x_2 \in \mathcal{V}$

$S_1 \times T_1 = T \equiv x_1 : S_1 \times T_1 = T$  für ein beliebiges  $x_1 \in \mathcal{V}$

$\langle s_1, t_1 \rangle = \langle s_2, t_2 \rangle \in x : S \times T \equiv x : S \times T \text{ Typ und } s_1 = s_2 \in S \text{ und } t_1 = t_2 \in T[s_1/x]$

Inferenzregeln und Taktiken im Appendix A.3.2 des Nuprl Manuals

# ANHANG: REGELN FÜR PRODUKTRAUM

$\Gamma \vdash \mathbb{U}_j \text{ [ext } x:S \times T]$   
**by dependent\_productFormation**  $x \ S$

$\Gamma \vdash S \in \mathbb{U}_j \text{ [Ax]}$

$\Gamma, x:S \vdash \mathbb{U}_j \text{ [ext } T]$

$\Gamma \vdash x_1:S_1 \times T_1 = x_2:S_2 \times T_2 \in \mathbb{U}_j \text{ [Ax]}$

**by productEquality**  $x'$

$\Gamma \vdash S_1 = S_2 \in \mathbb{U}_j \text{ [Ax]}$

$\Gamma, x':S_1 \vdash T_1[x'/x_1] = T_2[x'/x_2] \in \mathbb{U}_j \text{ [Ax]}$

$\Gamma \vdash \langle s_1, t_1 \rangle = \langle s_2, t_2 \rangle \in x:S \times T \text{ [Ax]}$

**by dependent\_pairEquality**  $j \ x'$

$\Gamma \vdash s_1 = s_2 \in S \text{ [Ax]}$

$\Gamma \vdash t_1 = t_2 \in T[s_1/x] \text{ [Ax]}$

$\Gamma, x':S \vdash T[x'/x] \in \mathbb{U}_j \text{ [Ax]}$

$\Gamma \vdash \langle s_1, t_1 \rangle = \langle s_1, t_1 \rangle \in S \times T \text{ [Ax]}$

**by independent\_pairEquality**

$\Gamma \vdash s_1 = s_2 \in S \text{ [Ax]}$

$\Gamma \vdash t_1 = t_2 \in T \text{ [Ax]}$

$\Gamma \vdash \text{let } \langle x_1, y_1 \rangle = e_1 \text{ in } t_1 = \text{let } \langle x_2, y_2 \rangle = e_2 \text{ in } t_2 \in C[e_1/z] \text{ [Ax]}$

**by spreadEquality**  $z \ C \ x:S \times T \ s \ t \ y$

$\Gamma \vdash e_1 = e_2 \in x:S \times T \text{ [Ax]}$

$\Gamma, s:S, t:T[s/x], y:e_1=\langle s, t \rangle \in x:S \times T \vdash t_1[s, t/x_1, y_1] = t_2[s, t/x_2, y_2] \in C[\langle s, t \rangle/z] \text{ [Ax]}$

$\Gamma, z:x:S \times T, \Delta \vdash C \text{ [ext let } \langle s, t \rangle = z \text{ in } u]$

**by productElimination**  $i \ s \ t$

$\Gamma, z:x:S \times T, s:S, t:T[s/x] \Delta[\langle s, t \rangle/z] \vdash C[\langle s, t \rangle/z] \text{ [ext } u]$

$\Gamma \vdash \mathbb{U}_j \text{ [ext } S \times T]$

**by independent\_productFormation**

$\Gamma \vdash \mathbb{U}_j \text{ [ext } S]$

$\Gamma \vdash \mathbb{U}_j \text{ [ext } T]$

$\Gamma \vdash S_1 \times T_1 = S_2 \times T_2 \in \mathbb{U}_j \text{ [Ax]}$

**by independent\_productEquality**

$\Gamma \vdash S_1 = S_2 \in \mathbb{U}_j \text{ [Ax]}$

$\Gamma \vdash T_1 = T_2 \in \mathbb{U}_j \text{ [Ax]}$

$\Gamma \vdash x:S \times T \text{ [ext } \langle s, t \rangle]$

**by dependent\_pairFormation**  $j \ s \ x'$

$\Gamma \vdash s \in S \text{ [Ax]}$

$\Gamma \vdash T[s/x] \text{ [ext } t]$

$\Gamma, x':S \vdash T[x'/x] \in \mathbb{U}_j \text{ [Ax]}$

$\Gamma \vdash S \times T \text{ [ext } \langle s, t \rangle]$

**by independent\_pairFormation**

$\Gamma \vdash S \text{ [ext } s]$

$\Gamma \vdash T \text{ [ext } t]$

$\Gamma \vdash \text{let } \langle x, y \rangle = \langle s, t \rangle \text{ in } u = t_2 \in T \text{ [Ax]}$

**by spreadReduce**

$\Gamma \vdash u[s, t/x, y] = t_2 \in T \text{ [Ax]}$

# ANHANG: ÜBERSICHT GLEICHHEIT

## Schließen über Werte von Ausdrücken

### Syntax:

Kanonisch:  $s = t \in T$       **equal**{ }(s;t;T)  
 $Ax$       **Axiom**{ }()

Nichtkanonisch: —

**Auswertung:** —

### Semantik:

$$s_1 = t_1 \in T_1 = s_2 = t_2 \in T_2 \equiv T_1 = T_2, s_1 = s_2 \in T_1, \text{ und } t_1 = t_2 \in T_1$$
$$Ax = Ax \in s = t \in T \equiv s = t \in T$$

Inferenzregeln und Taktiken im Appendix A.3.5 des Nuprl Manuals



# ANHANG: ÜBERSICHT UNIVERSEN

## Repräsentation der Typ-Eigenschaft

### Syntax:

Kanonisch:  $\mathbb{U}_j$       **universe** $\{j:1\}()$

Nichtkanonisch: —

### Auswertung: —

### Semantik:

$$\mathbb{U}_{j_1} = \mathbb{U}_{j_2} \quad \equiv \quad j_1=j_2 \quad (\text{als natürliche Zahl})$$

$$x_1:S_1 \rightarrow T_1 = x_2:S_2 \rightarrow T_2 \in \mathbb{U}_j \quad \equiv \quad S_1=S_2 \in \mathbb{U}_j \text{ und } T_1[s_1/x_1]=T_2[s_2/x_2] \in \mathbb{U}_j \\ \text{für alle Terme } s_1, s_2 \text{ mit } s_1=s_2 \in S_1.$$

$$s_1=t_1 \in T_1 = s_2=t_2 \in T_2 \in \mathbb{U}_j \quad \equiv \quad T_1=T_2 \in \mathbb{U}_j, \quad s_1=s_2 \in T_1, \text{ und } t_1=t_2 \in T_1$$

$$\vdots \quad \equiv \quad \text{— analog für alle anderen Datentypen —}$$

Inferenzregeln und Taktiken im Appendix A.3.4 des Nuprl Manuals

# ANHANG: REGELN FÜR GLEICHHEIT UND UNIVERSEN

$\Gamma \vdash \mathbb{U}_j$  **[ext**  $s=t \in T$ ]  
**by** equalityFormation  $T$

$\Gamma \vdash T \in \mathbb{U}_j$   $\text{[Ax]}$

$\Gamma \vdash T$  **[ext**  $s$ ]

$\Gamma \vdash T$  **[ext**  $t$ ]

$\Gamma \vdash Ax = Ax \in s=t \in T$   $\text{[Ax]}$   
**by** axiomEquality

$\Gamma \vdash s = t \in T$   $\text{[Ax]}$

$\Gamma, z: s=t \in T, \Delta \vdash C$  **[ext**  $u$ ]  
**by** equalityElimination  $i$

$\Gamma, z: s=t \in T, \Delta[Ax/z] \vdash C[Ax/z]$  **[ext**  $u$ ]

$\Gamma, x:T, \Delta \vdash x = x \in T$   $\text{[Ax]}$   
**by** hypothesisEquality  $i$

$\Gamma \vdash \mathbb{U}_k$  **[ext**  $\mathbb{U}_j$ ]  
**by** universeFormation  $j$  \*

$\Gamma \vdash T \in \mathbb{U}_k$   $\text{[Ax]}$   
**by** cumulativity  $j$  \*

$\Gamma \vdash T \in \mathbb{U}_j$   $\text{[Ax]}$

\*: proviso  $j < k$

$\Gamma \vdash s_1=t_1 \in T_1 = s_2=t_2 \in T_2 \in \mathbb{U}_j$   $\text{[Ax]}$   
**by** equalityEquality

$\Gamma \vdash T_1 = T_2 \in \mathbb{U}_j$   $\text{[Ax]}$

$\Gamma \vdash s_1 = s_2 \in T_1$   $\text{[Ax]}$

$\Gamma \vdash t_1 = t_2 \in T_1$   $\text{[Ax]}$

$\Gamma \vdash C[s/x]$  **[ext**  $u$ ]  
**by** substitution  $j$   $s=t \in T$   $x$   $C$

$\Gamma \vdash s=t \in T$   $\text{[Ax]}$

$\Gamma \vdash C[t/x]$  **[ext**  $u$ ]

$\Gamma, x:T \vdash C \in \mathbb{U}_j$   $\text{[Ax]}$

$\Gamma \vdash s = t \in T$   $\text{[Ax]}$

**by** equality *Entscheidungsprozedur*

$\Gamma \vdash \mathbb{U}_j = \mathbb{U}_j \in \mathbb{U}_k$   $\text{[Ax]}$   
**by** universeEquality \*

# ANHANG: REGELN FÜR NATÜRLICHE ZAHLEN

$$\Gamma \vdash \bigcup_j [\text{ext } \mathbb{N}]$$

**by natFormation**

$$\Gamma \vdash 0 = 0 \in \mathbb{N} \text{ [Ax]}$$

**by zeroEquality**

$$\Gamma \vdash s(t_1) = s(t_2) \in \mathbb{N} \text{ [Ax]}$$

**by succEquality**

$$\Gamma \vdash t_1 = t_2 \in \mathbb{N}$$

$$\Gamma \vdash \text{PR}[f_1, n_1, x_1 \mapsto g_1](t_1) = \text{PR}[f_2, n_2, x_2 \mapsto g_2](t_2) \in T[t_1/z] \text{ [Ax]}$$

**by prEquality**  $z$   $T$   $n$   $h_n$

$$\Gamma \vdash t_1 = t_2 \in \mathbb{N} \text{ [Ax]}$$

$$\Gamma \vdash f_1 = f_2 \in T[0/z] \text{ [Ax]}$$

$$\Gamma, n:\mathbb{N}, h_n:T[x/z] \vdash g_1(s(n), h_n) = g_2(s(n), h_n) \in T[s(n)/z] \text{ [Ax]}$$

$$\Gamma, z:\mathbb{N}, \Delta \vdash C [\text{ext } \text{PR}[f, n, x \mapsto g](z)]$$

**by natElimination**  $i$   $n$   $h_n$

$$\Gamma, z:\mathbb{N}, \Delta \vdash C[0/z] [\text{ext } f]$$

$$\Gamma, z:\mathbb{N}, \Delta, n:\mathbb{N}, h_n:C[n/z] \vdash C[s(n)/z] [\text{ext } g(n, h_n)]$$

$$\Gamma \vdash \text{PR}[f, n, x \mapsto g](0) = t_2 \in T \text{ [Ax]}$$

**by prReduceBase**

$$\Gamma \vdash f = t_2 \in T \text{ [Ax]}$$

$$\Gamma \vdash \mathbb{N} \in \bigcup_j \text{[Ax]}$$

**by natEquality**

$$\Gamma \vdash \mathbb{N} [\text{ext } 0]$$

**by zeroFormation**

$$\Gamma \vdash \mathbb{N} [\text{ext } s(t)]$$

**by succFormation**

$$\Gamma \vdash \mathbb{N} [\text{ext } t]$$

$$\Gamma \vdash \text{PR}[f, n, x \mapsto g](s(t)) = t_2 \in T \text{ [Ax]}$$

**by prReduceUp**

$$\Gamma \vdash g(t, \text{PR}[f, n, x \mapsto g](t)) = t_2 \in T \text{ [Ax]}$$

# ANHANG: WEITERE REGELN

$$\Gamma, x:T, \Delta \vdash T \text{ [ext } x]$$

**by hypothesis**  $i$

$$\Gamma, \Delta \vdash C \text{ [ext } (\lambda x.t) s]$$

**by cut**  $i$   $T$   $x$

$$\Gamma, \Delta \vdash T \text{ [ext } s]$$

$$\Gamma, x:T, \Delta \vdash C \text{ [ext } t]$$

$$\Gamma, z:T, \Delta \vdash C \text{ [ext } t]$$

**by hyp\_replacement**  $i$   $S$   $j$

$$\Gamma, z:S, \Delta \vdash C \text{ [ext } t]$$

$$\Gamma, z:T, \Delta \vdash T = S \in \mathbb{U}_j \text{ [Ax]}$$

$$\Gamma \vdash C \text{ [ext } t]$$

**by lemma** "theorem-name"

$$\Gamma \vdash C \text{ [ext } t[\sigma]]$$

**by instantiate**  $\Gamma'$   $C'$   $\sigma$

$$\Gamma' \vdash C' \text{ [ext } t]$$

$$\Gamma, x:T, \Delta \vdash C \text{ [ext } t]$$

**by thin**  $i$

$$\Gamma, \Delta \vdash C \text{ [ext } t]$$

$$\Gamma \vdash T \text{ [ext } t]$$

**by introduction**  $t$

$$\Gamma \vdash t \in T \text{ [Ax]}$$

$$\Gamma \vdash t \in T \text{ [Ax]}$$

**by extract** "theorem-name"

$$\Gamma \vdash C \text{ [ext } t[y/x]]$$

**by rename**  $y$   $x$

$$\Gamma[x/y] \vdash C[x/y] \text{ [ext } t]$$