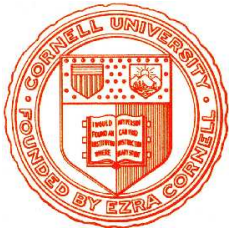
Automatisierte Logik und Programmierung

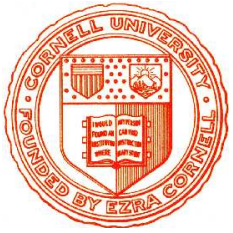
Teil II



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

Aufbau formaler konstruktiver Theorien

Vereinige Logik, Berechnung und Typstruktur

Vereinige Logik, Berechnung und Typstruktur

- **Den "einen richtigen Weg" gibt es nicht**
  - Der eleganteste Ansatz (abhängige Typen + $\mathbb{U} \in \mathbb{U}$) ist inkonsistent

Aufbau formaler konstruktiver Theorien

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

Aufbau formaler konstruktiver Theorien

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

Aufbau formaler konstruktiver Theorien

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**

Aufbau formaler konstruktiver Theorien

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

Aufbau formaler konstruktiver Theorien

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!

Aufbau formaler konstruktiver Theorien

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

Aufbau formaler konstruktiver Theorien

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

Aufbau formaler konstruktiver Theorien

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**

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

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 intutionistische Typentheorie**
  - Formalisierung wichtiger Konzepte aus Mathematik & Programmierung gekoppelt mit präzisiser Semantik und Beweiskalkül

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 intutionistische Typentheorie**
  - Formalisierung wichtiger Konzepte aus Mathematik & Programmierung gekoppelt mit präzisiser Semantik und Beweiskalkül
  - Signifikante Erweiterung von Martin-Löf's ursprünglicher Theorie

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

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 intutionistische Typentheorie**
  - Formalisierung wichtiger Konzepte aus Mathematik & Programmierung gekoppelt mit präzisiser Semantik und Beweiskalkül
  - Signifikante Erweiterung von Martin-Löf's ursprünglicher Theorie
  - Modifikation des Formalismus für Beweis- und Programmentwicklung
- **Implementiert im Nuprl Beweissystem**

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 intutionistische Typentheorie**
  - Formalisierung wichtiger Konzepte aus Mathematik & Programmierung gekoppelt mit präzisiser 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?**

Entwurfsentscheidungen sind zu treffen

1. **Umfang des Formalismus: minimal oder nicht?**
   – Wieviele Konzepte sollten innerhalb der Theorie vordefiniert sein?

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?

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?

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?

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?

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

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?

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

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

Entwurfsentscheidungen für CTT (1a): Umfang

- **Minimale Kalküle sind mathematisch elegant**
  - +: Gut für Entwickler: Nachweis von Eigenschaften leichter
  - +: Leicht zu erlernen: Regelsatz ist überschaubar

Entwurfsentscheidungen für CTT (1a): Umfang

- **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

Entwurfsentscheidungen für CTT (1a): Umfang

- **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?)

Entwurfsentscheidungen für CTT (1a): Umfang

- **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

Entwurfsentscheidungen für CTT (1a): Umfang

- **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

Entwurfsentscheidungen für CTT (1a): Umfang

- **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

Entwurfsentscheidungen für CTT (1a): Umfang

- **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

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,...

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, ...

Entwurfsentscheidungen für CTT (1b): Typsein / Gleichheit

- **Syntaktische Behandlung des Typseins**

  +: Einfache Repräsentation des Typuniversums

Entwurfsentscheidungen für CTT (1b): Typsein / Gleichheit

- **Syntaktische Behandlung des Typseins**
  - +: Einfache Repräsentation des Typuniversums
  - −: Objekte und Typen müssen syntaktisch getrennt werden

Entwurfsentscheidungen für CTT (1b): Typsein / Gleichheit

- **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

Entwurfsentscheidungen für CTT (1b): Typsein / Gleichheit

- **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 $\mathbb{U}=\mathbb{U}_1 \overset{\in}{\subset} \mathbb{U}_2 \overset{\in}{\subset} \mathbb{U}_3 \overset{\in}{\subset} \ldots$ erforderlich

Entwurfsentscheidungen für CTT (1b): Typsein / Gleichheit

- **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 $\mathbb{U}=\mathbb{U}_1 \in \subset \mathbb{U}_2 \in \subset \mathbb{U}_3 \in \subset \ldots$ erforderlich

Entwurfsentscheidungen für CTT (1b): Typsein / Gleichheit

- **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 $\mathbb{U}=\mathbb{U}_1 \in \subset \mathbb{U}_2 \in \subset \mathbb{U}_3 \in \subset \ldots$ erforderlich
- **Intensionale Gleichheit**
  - +: Leicht zu testen, da rein syntaktisches Konzept (textliche Identität)

Entwurfsentscheidungen für CTT (1b): Typsein / Gleichheit

- **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 $\mathbb{U}=\mathbb{U}_1 \in \subset \mathbb{U}_2 \in \subset \mathbb{U}_3 \in \subset \ldots$ erforderlich
- **Intensionale Gleichheit**
  - +: Leicht zu testen, da rein syntaktisches Konzept (textliche Identität)
  - −: Semantische Gleichheit muß vom Benutzer definiert werden

Entwurfsentscheidungen für CTT (1b): Typsein / Gleichheit

- **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 $\mathbb{U}{=}\mathbb{U}_1 \in \subset \mathbb{U}_2 \in \subset \mathbb{U}_3 \in \subset \ldots$ 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

Entwurfsentscheidungen für CTT (1b): Typsein / Gleichheit

- **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 $\mathbb{U}{=}\mathbb{U}_1 \in \mathbb{U}_2 \in \mathbb{U}_3 \in \ldots$ 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

Entwurfsentscheidungen für CTT (1b): Typsein / Gleichheit

- **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 $\mathbb{U}{=}\mathbb{U}_1 \in \mathbb{U}_2 \in \mathbb{U}_3 \in \ldots$ 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

Entwurfsentscheidungen für CTT (1c): Erweiterbarkeit

- **Geschlossene Formalisierung**
  - Theorie wird ein für alle Mal festgelegt

Entwurfsentscheidungen für CTT (1c): Erweiterbarkeit

- **Geschlossene Formalisierung**
  - – Theorie wird ein für alle Mal festgelegt
  - +: Erlaubt Beweise mit Fallunterscheidung über alle möglichen Terme

Entwurfsentscheidungen für CTT (1c): Erweiterbarkeit

- **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 ?

Entwurfsentscheidungen für CTT (1c): Erweiterbarkeit

- **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

Entwurfsentscheidungen für CTT (1c): Erweiterbarkeit

- **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

Entwurfsentscheidungen für CTT (1c): Erweiterbarkeit

- **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

Entwurfsentscheidungen für CTT (1c): Erweiterbarkeit

- **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

Leitbeispiel: Basiskonzepte der CTT

- **Variablensymbole**
  - Ascii-Bezeichner, einheitlich für Objekt- und Typvariablen
- **Natürliche Zahlen** $\mathbb{N}$
  - Vorläufig als Minimalkonzept wie bei Peano-Axiomen
  - Null $\mathsf{0}$, Nachfolger $\mathsf{s}(i)$, induktive Definition $\mathsf{PR}[f, n, x \mapsto g](i)$

Leitbeispiel: Basiskonzepte der CTT

- **Variablensymbole**
  - Ascii-Bezeichner, einheitlich für Objekt- und Typvariablen
- **Natürliche Zahlen** $\mathbb{N}$
  - Vorläufig als Minimalkonzept wie bei Peano-Axiomen
  - Null $\mathtt{0}$, Nachfolger $\mathtt{s}(i)$, induktive Definition $\mathsf{PR}[f, n, x{\mapsto}g](i)$
- **Funktionenräume** $x{:}S{\to}T$ **und** $S{\to}T$
  - Funktionsdefinition $\lambda x.t$ und -anwendung $f\,t$
  - Typ des Bilds einer Funktion kann vom Wert der Eingabe abhängen

Leitbeispiel: Basiskonzepte der CTT

- **Variablensymbole**
  - Ascii-Bezeichner, einheitlich für Objekt- und Typvariablen
- **Natürliche Zahlen** $\mathbb{N}$
  - Vorläufig als Minimalkonzept wie bei Peano-Axiomen
  - Null $\mathsf{0}$, Nachfolger $\mathsf{s}(i)$, induktive Definition $\mathsf{PR}[f, n, x{\mapsto}g](i)$
- **Funktionenräume** $x{:}S{\to}T$ **und** $S{\to}T$
  - 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 $\mathsf{let}\ \langle x, y\rangle = p\ \mathsf{in}\ t$
  - Typ der zweiten Komponente kann vom Wert der ersten abhängen

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 $\mathsf{s}(i)$, induktive Definition $\mathsf{PR}[f, n, x{\mapsto}g](i)$
- **Funktionenräume** $x{:}S{\to}T$ **und** $S{\to}T$
  - 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 $\mathsf{let}\ \langle x, y\rangle = p\ \mathsf{in}\ t$
  - Typ der zweiten Komponente kann vom Wert der ersten abhängen
- **Gleichheit als Datentyp** $s{=}t \in T$
  - Notwendig für konstruierenden Beweiskalkül

(siehe Folie 39)

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 $\mathsf{s}(i)$, induktive Definition $\mathsf{PR}[f, n, x{\mapsto}g](i)$
- **Funktionenräume** $x{:}S{\to}T$ **und** $S{\to}T$
  - 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 $\mathsf{let}\ \langle x, y\rangle = p\ \mathsf{in}\ t$
  - 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**

Entwurfsentscheidungen für CTT (2): Syntax

- **Maschinenlesbarkeit ist notwendig**
  - +: Leichte Erzeugung des abstrakten Syntaxbaums aus formalem Text
  - · Syntaxbaum essentiell für Verarbeitung von Ausdrücken

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

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**

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

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

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** ✓

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

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
- $\mathsf{s}(t)$ Ausdruck, falls t Ausdruck
- $\mathsf{PR}[f,\, n, x \mapsto g](t)$ Ausdruck, falls $x, n \in \mathcal{V}$ und f, g, und t Ausdrücke
- $x : S \to T$ und $S \to T$ 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
- $\mathsf{let}\ \ \langle x \,, y \rangle = p\ \ \mathsf{in}\ \ t$ 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 $i \geq 0$

Syntaxdefinition ohne Systematisierung

- x Ausdruck, falls $x \in \mathcal{V}$
- (t) Ausdruck, falls t Ausdruck
- $\mathbb{N}$ und 0 sind Ausdrücke
- $\mathsf{s}(t)$ Ausdruck, falls t Ausdruck
- $\mathsf{PR}[f,\ n, x \mapsto g](t)$ Ausdruck, falls $x, n \in \mathcal{V}$ und f, g, und t Ausdrücke
- $x : S \to T$ und $S \to T$ 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
- $\mathsf{let}\ \langle x\,, y \rangle = p\ \mathsf{in}\ t$ 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

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

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

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)

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

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**

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**$\{x : \mathtt{v}\}$

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 $\mathsf{var}\{x : \mathtt{v}\}$
- **Teilterme**
  - Die meisten Terme sind aus anderen Termen zusammengesetzt
  - $f\ t$ ist Abkürzung für einen Ausdruck der Form $\mathsf{apply}(f\,;t)$

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 $\mathsf{var}\{x : \mathtt{v}\}$
- **Teilterme**
  - Die meisten Terme sind aus anderen Termen zusammengesetzt
  - $f\,t$ ist Abkürzung für einen Ausdruck der Form $\mathsf{apply}(f\,;t)$
- **Gebundene Variablen in Teiltermen**
  - Teilterme können Variablen enthalten, die gebunden werden
  - In $\lambda x\,.\,t$ wird x in t gebunden – Ausdruck ist $\mathsf{lam}(x\,.\,t)$
  - In $x : S \to T$ wird x in T (nicht in S) gebunden
    Ausdruck ist $\mathsf{function}(S\,;\,x\,.\,T)$

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 $\mathsf{var}\{x:\mathtt{v}\}$
- **Teilterme**
  - Die meisten Terme sind aus anderen Termen zusammengesetzt
  - $f\,t$ ist Abkürzung für einen Ausdruck der Form $\mathsf{apply}(f\,;t)$
- **Gebundene Variablen in Teiltermen**
  - Teilterme können Variablen enthalten, die gebunden werden
  - In $\lambda x\,.\,t$ wird x in t gebunden – Ausdruck ist $\mathsf{lam}(x\,.\,t)$
  - In $x:S\to T$ wird x in T (nicht in S) gebunden
    Ausdruck ist $\mathsf{function}(S\,;\,x\,.\,T)$

Allgemeine Form: $\boldsymbol{opid}\{p_1\,..\,p_k\}(x_1^1..x_{m_1}^1\,.\,t_1;\,\ldots\,x_1^n..x_{m_n}^n\,.\,t_n)$

Terme der CTT: präzise Definition

Die Abstraktionsform eines Terms hat die Gestalt

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

wobei

Terme der CTT: präzise Definition

Die Abstraktionsform eines Terms hat die Gestalt

$$\boldsymbol{opid}\{p_1{:}F_1, ..p_k{:}F_k\}(x_1^1, ..x_{m_1}^1.t_1; \ldots x_1^n, ..x_{m_n}^n.t_n)$$

wobei

- $\boldsymbol{opid}\{p_1{:}F_1, ..p_k{:}F_k\}$ **Operator**:
  - *opid*: **Operatorname** → *Operatorentabelle*

    Operatorentabelle muß Operator var enthalten

Terme der CTT: präzise Definition

Die Abstraktionsform eines Terms hat die Gestalt

$$\boldsymbol{opid}\{\boldsymbol{p_1}:\boldsymbol{F_1},..\boldsymbol{p_k}:\boldsymbol{F_k}\}(\boldsymbol{x_1^1},..\boldsymbol{x_{m_1}^1}.\boldsymbol{t_1};\ldots\boldsymbol{x_1^n},..\boldsymbol{x_{m_n}^n}.\boldsymbol{t_n})$$

wobei

- $\boldsymbol{opid}\{\boldsymbol{p_1}:\boldsymbol{F_1},..\boldsymbol{p_k}:\boldsymbol{F_k}\}$ **Operator**:
  - $opid$: **Operatorname** $\rightarrow$ *Operatorentabelle*

    Operatorentabelle muß Operator var enthalten
  - $p_j:F_j$: **Parameter**
    - F_j: **Parametertyp** `(var, nat, tok, str, level)`
    - p_j: **Parameterwert**

Terme der CTT: präzise Definition

Die Abstraktionsform eines Terms hat die Gestalt

$$\boldsymbol{opid}\{\boldsymbol{p_1}:\boldsymbol{F_1},..\boldsymbol{p_k}:\boldsymbol{F_k}\}(\boldsymbol{x_1^1},..\boldsymbol{x_{m_1}^1}.\boldsymbol{t_1};\ldots\boldsymbol{x_1^n},..\boldsymbol{x_{m_n}^n}.\boldsymbol{t_n})$$

wobei

- $\boldsymbol{opid}\{\boldsymbol{p_1}:\boldsymbol{F_1},..\boldsymbol{p_k}:\boldsymbol{F_k}\}$ **Operator**:
  - $opid$: **Operatorname** $\rightarrow$ *Operatorentabelle*

    Operatorentabelle muß Operator var enthalten
  - $p_j:F_j$: **Parameter**
    - F_j: **Parametertyp** `(var, nat, tok, str, level)`
    - p_j: **Parameterwert**
- $x_1^i,\ldots,x_{m_i}^i.t_i$ **gebundener Term**, mit Term t_i und Variablen x_k^j

Terme der CTT: präzise Definition

Die Abstraktionsform eines Terms hat die Gestalt

$$\boldsymbol{opid\{p_1{:}F_1,..p_k{:}F_k\}(x_1^1,..x_{m_1}^1.t_1;\ldots x_1^n,..x_{m_n}^n.t_n)}$$

wobei

- $\boldsymbol{opid\{p_1{:}F_1,..p_k{:}F_k\}}$ **Operator**:
  - $opid$: **Operatorname** $\rightarrow$ *Operatorentabelle*

    Operatorentabelle muß Operator var enthalten
  - $p_j{:}F_j$: **Parameter**
    - F_j: **Parametertyp** `(var, nat, tok, str, level)`
    - p_j: **Parameterwert**
- $x_1^i,\ldots,x_{m_i}^i.t_i$ **gebundener Term**, mit Term t_i und Variablen x_k^j

  $(m_1,..,m_n)$ ist die **Arität** (Anzahlen der Variablen aller Teilterme)

Terme der CTT: präzise Definition

Die Abstraktionsform eines Terms hat die Gestalt

$$\boldsymbol{opid}\{\boldsymbol{p_1}:\boldsymbol{F_1},..\boldsymbol{p_k}:\boldsymbol{F_k}\}(\boldsymbol{x_1^1},..\boldsymbol{x_{m_1}^1}.\boldsymbol{t_1};\ldots\boldsymbol{x_1^n},..\boldsymbol{x_{m_n}^n}.\boldsymbol{t_n})$$

wobei

- $\boldsymbol{opid}\{\boldsymbol{p_1}:\boldsymbol{F_1},..\boldsymbol{p_k}:\boldsymbol{F_k}\}$ **Operator**:
  - $opid$: **Operatorname** $\rightarrow$ *Operatorentabelle*

    Operatorentabelle muß Operator var enthalten
  - $p_j:F_j$: **Parameter**
    - F_j: **Parametertyp** `(var, nat, tok, str, level)`
    - p_j: **Parameterwert**
- $x_1^i,\ldots,x_{m_i}^i.t_i$ **gebundener Term**, mit Term t_i und Variablen x_k^j

  $(m_1,..,m_n)$ ist die **Arität** (Anzahlen der Variablen aller Teilterme)

Abstraktionsform und Darstellungsform werden getrennt behandelt, aber simultan betrachtet

Auszug der Operatorentabelle

Operator und Termstruktur	*Standard-Darstellungsform*
var{x:v}()	x
Nat{}()	$\mathbb{N}$
zero{}()	0
suc{}(t)	s(t)
p_rec{}(t; f; n,x.g)	PR[f, n,$x \mapsto g$](t)
function{}(S; x.T)	x:$S \longrightarrow T$
lambda{}(x.t)	λx.t
apply{}(f;t)	$f\ t$
product{}(S; x.T)	x:$S \times T$
pair{}(s,t)	$\langle s, t \rangle$
spread{}(e; x,y.u)	let $\langle x, y \rangle = e$ in u
Axiom{}()	Ax
equal{}(s;t;T)	$s = t \in T$
universe{j:l}()	$\mathbb{U}_j$

Display Formen

- **Beschreibe Notation eines abstrakten Terms**

Display Formen

- **Beschreibe Notation eines abstrakten Terms**

```
EdAlias lam: λ<x:var>.<t:term:E> == lambda{}(<x>.<t>)
     #Hd A:: λ<x:var>,<#:term:E> == lambda{}(<x>.<#>)
     #Tl A:: <x:var>.<t:term:E>  == lambda{}(<x>.<t>)
     #Tl A:: <x:var>,<#:term:E>  == lambda{}(<x>.<#>)
```

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

Display Formen

- **Beschreibe Notation eines abstrakten Terms**

```
EdAlias lam: λ<x:var>.<t:term:E> == lambda{}(<x>.<t>)
     #Hd A:: λ<x:var>,<#:term:E> == lambda{}(<x>.<#>)
     #Tl A:: <x:var>.<t:term:E>  == lambda{}(<x>.<t>)
     #Tl A:: <x:var>,<#:term:E>  == lambda{}(<x>.<#>)
```

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

- **Sonderformen erhöhen Flexibilität**

Display Formen

- **Beschreibe Notation eines abstrakten Terms**

```
EdAlias lam: λ<x:var>.<t:term:E> == lambda{}(<x>.<t>)
     #Hd A:: λ<x:var>,<#:term:E> == lambda{}(<x>.<#>)
     #Tl A:: <x:var>.<t:term:E>  == lambda{}(<x>.<t>)
     #Tl A:: <x:var>,<#:term:E>  == lambda{}(<x>.<#>)
```

  - 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)

Display Formen

- **Beschreibe Notation eines abstrakten Terms**

```
EdAlias lam: λ<x:var>.<t:term:E> == lambda{}(<x>.<t>)
     #Hd A:: λ<x:var>,<#:term:E> == lambda{}(<x>.<#>)
     #Tl A:: <x:var>.<t:term:E>  == lambda{}(<x>.<t>)
     #Tl A:: <x:var>,<#:term:E>  == lambda{}(<x>.<#>)
```

  - 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 $\lambda x.\lambda y.t$

Display Formen

- **Beschreibe Notation eines abstrakten Terms**

```
EdAlias lam: λ<x:var>.<t:term:E> == lambda{}(<x>.<t>)
     #Hd A:: λ<x:var>,<#:term:E> == lambda{}(<x>.<#>)
     #Tl A:: <x:var>.<t:term:E>  == lambda{}(<x>.<t>)
     #Tl A:: <x:var>,<#:term:E>  == lambda{}(<x>.<#>)
```

  - 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

Display Formen

- **Beschreibe Notation eines abstrakten Terms**

```
EdAlias lam: λ<x:var>.<t:term:E> == lambda{}(<x>.<t>)
      #Hd A:: λ<x:var>,<#:term:E> == lambda{}(<x>.<#>)
      #Tl A:: <x:var>.<t:term:E>  == lambda{}(<x>.<t>)
      #Tl A:: <x:var>,<#:term:E>  == lambda{}(<x>.<#>)
```

  - 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(<t:int>) {->}where h(0) = <f:base>
             {\\}and h(n+1) = <g:step>(<n:var>,h(n)/<x:var>)
  == p_rec{}(<t>;<f>;<n>,<x>.<g>)
```

Erweiterungen der Syntax

- **Abstraktionen für konservative Erweiterungen**

Erweiterungen der Syntax

- **Abstraktionen für konservative Erweiterungen**
  - $\texttt{add\{\}(.i;.j)} \equiv \texttt{PR[i, n,add-in} \mapsto \texttt{s(add-in)](j)}$
  - $\texttt{mul\{\}(.i;.j)} \equiv \texttt{PR[0, n,mul-in} \mapsto \texttt{add\{\}(.mul-in;j)](j)}$

Erweiterungen der Syntax

- **Abstraktionen für konservative Erweiterungen**
  - `add{}(.i;.j)` $\equiv$ `PR[i, n,add-in` $\mapsto$ `s(add-in)](j)`
  - `mul{}(.i;.j)` $\equiv$ `PR[0, n,mul-in` $\mapsto$ `add{}(.mul-in;j)](j)`
  - Linke Seite beschreibt Erweiterung der Operatorentabelle
  - Rechte Seite beschreibt Semantik durch existierende Terme

Erweiterungen der Syntax

- **Abstraktionen für konservative Erweiterungen**
  - `add{}(.i;.j)` $\equiv$ `PR[i, n,add-in` $\mapsto$ `s(add-in)](j)`
  - `mul{}(.i;.j)` $\equiv$ `PR[0, n,mul-in` $\mapsto$ `add{}(.mul-in;j)](j)`
  - Linke Seite beschreibt Erweiterung der Operatorentabelle
  - Rechte Seite beschreibt Semantik durch existierende Terme
- **Systemeintrag für echte Theorieerweiterungen**

Erweiterungen der Syntax

- **Abstraktionen für konservative Erweiterungen**
  - $\texttt{add\{\}(.i;.j)} \equiv \texttt{PR[i, n,add-in} \mapsto \texttt{s(add-in)](j)}$
  - $\texttt{mul\{\}(.i;.j)} \equiv \texttt{PR[0, n,mul-in} \mapsto \texttt{add\{\}(.mul-in;j)](j)}$
  - Linke Seite beschreibt Erweiterung der Operatorentabelle
  - Rechte Seite beschreibt Semantik durch existierende Terme
- **Systemeintrag für echte Theorieerweiterungen**
  - Term wird direkt in Operatorentabelle vermerkt
  - Semantik und Inferenzregeln sind explizit zu beschreiben

Erweiterungen der Syntax

- **Abstraktionen für konservative Erweiterungen**
  - `add{}(.i;.j)` $\equiv$ `PR[i, n,add-in` $\mapsto$ `s(add-in)](j)`
  - `mul{}(.i;.j)` $\equiv$ `PR[0, n,mul-in` $\mapsto$ `add{}(.mul-in;j)](j)`
  - – Linke Seite beschreibt Erweiterung der Operatorentabelle
  - – Rechte Seite beschreibt Semantik durch existierende Terme
- **Systemeintrag für echte Theorieerweiterungen**
  - – Term wird direkt in Operatorentabelle vermerkt
  - – Semantik und Inferenzregeln sind explizit zu beschreiben
- **Displayformen sind in beiden Fällen anzugeben**
  - `EdAlias add: <i:int>+<j:int> == add{}(.<i>;.<j>)`
  - `EdAlias mul: <i:int>*<j:int> == mul{}(.<i>;.<j>)`

Erweiterungen der Syntax

- **Abstraktionen für konservative Erweiterungen**
  - `add{}(.i;.j)` ≡ `PR[i, n,add-in↦s(add-in)](j)`
  - `mul{}(.i;.j)` ≡ `PR[0, n,mul-in↦add{}(.mul-in;j)](j)`
  - Linke Seite beschreibt Erweiterung der Operatorentabelle
  - Rechte Seite beschreibt Semantik durch existierende Terme
- **Systemeintrag für echte Theorieerweiterungen**
  - Term wird direkt in Operatorentabelle vermerkt
  - Semantik und Inferenzregeln sind explizit zu beschreiben
- **Displayformen sind in beiden Fällen anzugeben**
  - `EdAlias add: <i:int>+<j:int> == add{}(.<i>;.<j>)`
  - `EdAlias mul: <i:int>*<j:int> == mul{}(.<i>;.<j>)`
- **Beispiel für Präsentation von Termen**

Erweiterungen der Syntax

- **Abstraktionen für konservative Erweiterungen**
  - `add{}(.i;.j)` ≡ `PR[i, n,add-in↦s(add-in)](j)`
  - `mul{}(.i;.j)` ≡ `PR[0, n,mul-in↦add{}(.mul-in;j)](j)`
  - Linke Seite beschreibt Erweiterung der Operatorentabelle
  - Rechte Seite beschreibt Semantik durch existierende Terme
- **Systemeintrag für echte Theorieerweiterungen**
  - Term wird direkt in Operatorentabelle vermerkt
  - Semantik und Inferenzregeln sind explizit zu beschreiben
- **Displayformen sind in beiden Fällen anzugeben**
  - `EdAlias add: <i:int>+<j:int> == add{}(.<i>;.<j>)`
  - `EdAlias mul: <i:int>*<j:int> == mul{}(.<i>;.<j>)`
- **Beispiel für Präsentation von Termen**
  - Interne Form

```
lambda{}(x.lambda{}(y.lambda{}(z.
                add{}(.var{x:v}();
                      .mul{}(.var{y:v}();.var{z:v}())))))
```

Erweiterungen der Syntax

- **Abstraktionen für konservative Erweiterungen**
  - `add{}(.i;.j)` ≡ `PR[i, n,add-in↦s(add-in)](j)`
  - `mul{}(.i;.j)` ≡ `PR[0, n,mul-in↦add{}(.mul-in;j)](j)`
  - Linke Seite beschreibt Erweiterung der Operatorentabelle
  - Rechte Seite beschreibt Semantik durch existierende Terme
- **Systemeintrag für echte Theorieerweiterungen**
  - Term wird direkt in Operatorentabelle vermerkt
  - Semantik und Inferenzregeln sind explizit zu beschreiben
- **Displayformen sind in beiden Fällen anzugeben**
  - `EdAlias add: <i:int>+<j:int> == add{}(.<i>;.<j>)`
  - `EdAlias mul: <i:int>*<j:int> == mul{}(.<i>;.<j>)`
- **Beispiel für Präsentation von Termen**
  - Interne Form
```
lambda{}(x.lambda{}(y.lambda{}(z.
                add{}(.var{x:v}();
                      .mul{}(.var{y:v}();.var{z:v}())))))
```
  - Externe Präsentation `λx,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.
- $\mathsf{s}(t)$: freie Vorkommen von x in t bleiben frei, gebundene Vorkommen bleiben gebunden.
- $\mathsf{PR}[f,\, n, x{\mapsto}g](t)$: freie Vorkommen von x und n in g werden gebunden;
  freie Vorkommen von $y{\neq}x/n$ in g bleiben frei, gebundene bleiben gebunden.
- $x : S{\to}T$: 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.
- $\mathsf{let}\ \ \langle x\,, y\rangle = p\ \mathsf{in}\ \ t$: 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.

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.
- $\mathsf{s}(t)$: freie Vorkommen von x in t bleiben frei, gebundene Vorkommen bleiben gebunden.
- $\mathsf{PR}[f,\ n, x{\mapsto}g](t)$: freie Vorkommen von x und n in g werden gebunden;
  freie Vorkommen von $y{\neq}x/n$ in g bleiben frei, gebundene bleiben gebunden.
- $x : S{\to}T$: 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.
- $\mathsf{let}\ \langle x\,,y\rangle = p\ \mathsf{in}\ t$: 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

Vorkommen von Variablen: einheitliche Definition

Definition nur auf Basis der uniformen Syntax

- $\mathsf{var}\{x : \mathtt{v}\}$: die Variable $x \in \mathcal{V}$ kommt frei vor;
  $y \neq x$ kommt nicht vor.

Vorkommen von Variablen: einheitliche Definition

Definition nur auf Basis der uniformen Syntax

- $\mathsf{var}\{x : \mathtt{v}\}$: 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.

Vorkommen von Variablen: einheitliche Definition

Definition nur auf Basis der uniformen Syntax

- $\mathsf{var}\{x : \mathtt{v}\}$: 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.
- $x_1, .., x_m \,.\, t$: freie und gebundene Vorkommen der x_i in t
  werden gebunden;
  freie Vorkommen von $y \neq x_i$ bleiben frei,
  gebundene Vorkommen bleiben gebunden

Vorkommen von Variablen: einheitliche Definition

Definition nur auf Basis der uniformen Syntax

- $\mathsf{var}\{x : \mathtt{v}\}$: 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.
- $x_1, .., x_m \,.\, t$: 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

Vorkommen von Variablen: einheitliche Definition

Definition nur auf Basis der uniformen Syntax

- $\mathsf{var}\{x : \mathtt{v}\}$: 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.
- $x_1, .., x_m \,.\, t$: 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 $\boldsymbol{\sigma}$ von Variablen in Terme

– $\boldsymbol{\sigma = [t_1, .., t_n/x_1, .., x_n]} \mathrel{\hat{=}} \sigma(x_1){=}t_1, \ldots, \sigma(x_n){=}t_n$

– $\boldsymbol{a\sigma}$: Anwendung von σ auf den Ausdruck a

Substitution $a[t/x]$: einheitliche Definition

Endliche Abbildung σ von Variablen in Terme

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

$$\mathsf{var}\{x : \mathrm{v}\}()[t/x] = t$$

Substitution $a[t/x]$: einheitliche Definition

Endliche Abbildung σ von Variablen in Terme

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

$$\mathsf{var}\{x:\mathrm{v}\}()[t/x] = t$$

$$\mathsf{var}\{x:\mathrm{v}\}()[t/y] = \mathsf{var}\{x:\mathrm{v}\}() \qquad (y{\neq}x)$$

Substitution $a[t/x]$: einheitliche Definition

Endliche Abbildung σ von Variablen in Terme

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

$$\mathsf{var}\{x:\mathrm{v}\}()[t/x] = t$$

$$\mathsf{var}\{x:\mathrm{v}\}()[t/y] = \mathsf{var}\{x:\mathrm{v}\}() \qquad (y{\neq}x)$$

$$(op(b_1;\ldots;b_n))[t/x] = op(b_1[t/x];\ldots;b_n[t/x])$$

Substitution $a[t/x]$: einheitliche Definition

Endliche Abbildung σ von Variablen in Terme

– $\sigma = [t_1, .., t_n/x_1, .., x_n] \hat{=} \sigma(x_1){=}t_1, \ldots, \sigma(x_n){=}t_n$

– $a\sigma$: Anwendung von σ auf den Ausdruck a

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

*: *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

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

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

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?

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 nachgeweisen werden

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 $\mathtt{0}, \langle \mathtt{0}, \mathtt{s(0)} \rangle, \lambda \mathtt{x.x}$ haben intuitive Bedeutung
    - Andere müssen erst "ausgerechnet" werden

Formulierung einer direkten Semantik für CTT

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

Formulierung einer direkten Semantik für CTT

- **Ausdrücke haben Bedeutung**
  - Sie beschreiben Objekte, Funktionen, Datentypen, Aussagen, ...
    - Ausdrücke wie $\mathtt{0}, \langle \mathtt{0}, \mathtt{s(0)} \rangle, \lambda \mathtt{x.x}$ 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)

Formulierung einer direkten Semantik für CTT

- **Ausdrücke haben Bedeutung**
  - Sie beschreiben Objekte, Funktionen, Datentypen, Aussagen, ...
    - Ausdrücke wie $\mathtt{0}, \langle \mathtt{0}, \mathtt{s(0)} \rangle, \lambda \mathtt{x.x}$ 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

Formulierung einer direkten Semantik für CTT

- **Ausdrücke haben Bedeutung**
  - Sie beschreiben Objekte, Funktionen, Datentypen, Aussagen, ...
    - Ausdrücke wie $\mathtt{0}, \langle \mathtt{0}, \mathtt{s(0)} \rangle, \lambda \mathtt{x.x}$ 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

Formulierung einer direkten Semantik für CTT

- **Ausdrücke haben Bedeutung**
  - Sie beschreiben Objekte, Funktionen, Datentypen, Aussagen, ...
    - Ausdrücke wie $\mathtt{0}, \langle \mathtt{0}, \mathtt{s(0)} \rangle, \lambda \mathtt{x.x}$ 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**

Formulierung einer direkten Semantik für CTT

- **Ausdrücke haben Bedeutung**
  - Sie beschreiben Objekte, Funktionen, Datentypen, Aussagen, ...
    - Ausdrücke wie $\mathtt{0}, \langle \mathtt{0}, \mathtt{s(0)} \rangle, \lambda \mathtt{x.x}$ 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

Formulierung einer direkten Semantik für CTT

- **Ausdrücke haben Bedeutung**
  - Sie beschreiben Objekte, Funktionen, Datentypen, Aussagen, ...
    - Ausdrücke wie $\mathtt{0}, \langle \mathtt{0}, \mathtt{s(0)} \rangle, \lambda \mathtt{x.x}$ 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 $\equiv$ $\mathtt{X}:\mathbb{U}_1\rightarrow(\mathtt{X}=\mathtt{X}\rightarrow\mathtt{X}\in\mathbb{U}_1)$ kann keine Elemente haben

Starke Normalisierbarkeit ist nicht möglich

- **Ein leerer Datentyp kann formuliert werden**
  - $\mathsf{Void} \equiv \mathtt{X}:\mathbb{U}_1 \to (\mathtt{X} = \mathtt{X} \to \mathtt{X} \in \mathbb{U}_1)$ 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* $\mathsf{Void} \to T$

Starke Normalisierbarkeit ist nicht möglich

- **Ein leerer Datentyp kann formuliert werden**
  - $\mathsf{Void} \equiv \mathtt{X}:\mathbb{U}_1 \to (\mathtt{X} = \mathtt{X} \to \mathtt{X} \in \mathbb{U}_1)$ 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* $\mathsf{Void} \to T$
  - Sei z eine Eingabe aus $\mathsf{Void} = \mathtt{X}:\mathbb{U}_1 \to (\mathtt{X} = \mathtt{X} \to \mathtt{X} \in \mathbb{U}_1)$

Starke Normalisierbarkeit ist nicht möglich

- **Ein leerer Datentyp kann formuliert werden**
  - $\mathsf{Void} \equiv \mathtt{X}:\mathbb{U}_1 \to (\mathtt{X} = \mathtt{X} \to \mathtt{X} \in \mathbb{U}_1)$ 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* $\mathsf{Void} \to T$
  - Sei z eine Eingabe aus $\mathsf{Void} = \mathtt{X}:\mathbb{U}_1 \to (\mathtt{X} = \mathtt{X} \to \mathtt{X} \in \mathbb{U}_1)$
  - Dann ist $z(T)$ ein Beweis für (Element von) $T = T \to T \in \mathbb{U}_1$

Starke Normalisierbarkeit ist nicht möglich

- **Ein leerer Datentyp kann formuliert werden**
  - $\mathsf{Void} \equiv \mathtt{X}:\mathbb{U}_1 \to (\mathtt{X} = \mathtt{X} \to \mathtt{X} \in \mathbb{U}_1)$ 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* $\mathsf{Void} \to T$
  - Sei z eine Eingabe aus $\mathsf{Void} = \mathtt{X}:\mathbb{U}_1 \to (\mathtt{X} = \mathtt{X} \to \mathtt{X} \in \mathbb{U}_1)$
  - Dann ist $z(T)$ ein Beweis für (Element von) $T = T \to T \in \mathbb{U}_1$
  - Also kann x gleichzeitig zu $T \to T$ und zu T gehören

Starke Normalisierbarkeit ist nicht möglich

- **Ein leerer Datentyp kann formuliert werden**
  - $\mathsf{Void} \equiv \mathtt{X}:\mathbb{U}_1 \to (\mathtt{X} = \mathtt{X} \to \mathtt{X} \in \mathbb{U}_1)$ 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* $\mathsf{Void} \to T$
  - Sei z eine Eingabe aus $\mathsf{Void} = \mathtt{X}:\mathbb{U}_1 \to (\mathtt{X} = \mathtt{X} \to \mathtt{X} \in \mathbb{U}_1)$
  - Dann ist $z(T)$ ein Beweis für (Element von) $T = T \to T \in \mathbb{U}_1$
  - Also kann x gleichzeitig zu $T \to T$ und zu T gehören
  - Damit $x\, x \in T$ und $\lambda x.\, x\, x \in T \to T = T$ also $(\lambda x.\, x\, x)(\lambda x.\, x\, x) \in T$

Starke Normalisierbarkeit ist nicht möglich

- **Ein leerer Datentyp kann formuliert werden**
  - $\mathsf{Void} \equiv \mathtt{X}:\mathbb{U}_1 \to (\mathtt{X} = \mathtt{X} \to \mathtt{X} \in \mathbb{U}_1)$ 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* $\mathsf{Void} \to T$
  - Sei z eine Eingabe aus $\mathsf{Void} = \mathtt{X}:\mathbb{U}_1 \to (\mathtt{X} = \mathtt{X} \to \mathtt{X} \in \mathbb{U}_1)$
  - Dann ist $z(T)$ ein Beweis für (Element von) $T = T \to T \in \mathbb{U}_1$
  - Also kann x gleichzeitig zu $T \to T$ und zu T gehören
  - Damit $x\, x \in T$ und $\lambda x.\, x\, x \in T \to T = T$ 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

Starke Normalisierbarkeit ist nicht möglich

- **Ein leerer Datentyp kann formuliert werden**
  - $\mathsf{Void} \equiv \mathtt{X}:\mathbb{U}_1 \to (\mathtt{X} = \mathtt{X} \to \mathtt{X} \in \mathbb{U}_1)$ 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* $\mathsf{Void} \to T$
  - Sei z eine Eingabe aus $\mathsf{Void} = \mathtt{X}:\mathbb{U}_1 \to (\mathtt{X} = \mathtt{X} \to \mathtt{X} \in \mathbb{U}_1)$
  - Dann ist $z(T)$ ein Beweis für (Element von) $T = T \to T \in \mathbb{U}_1$
  - Also kann x gleichzeitig zu $T \to T$ und zu T gehören
  - Damit $x\, x \in T$ und $\lambda x.\, x\, x \in T \to T = T$ 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 $z \in \mathsf{Void}$ bestimmt sie einen Wert aus T
    - Da es kein $z \in \mathsf{Void}$ gibt, muß $(\lambda x.\, x\, x)(\lambda x.\, x\, x)$ nie berechnet werden

Starke Normalisierbarkeit ist nicht möglich

- **Ein leerer Datentyp kann formuliert werden**
  - $\mathsf{Void} \equiv \mathtt{X}:\mathbb{U}_1 \to (\mathtt{X} = \mathtt{X} \to \mathtt{X} \in \mathbb{U}_1)$ 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* $\mathsf{Void} \to T$
  - Sei z eine Eingabe aus $\mathsf{Void} = \mathtt{X}:\mathbb{U}_1 \to (\mathtt{X} = \mathtt{X} \to \mathtt{X} \in \mathbb{U}_1)$
  - Dann ist $z(T)$ ein Beweis für (Element von) $T = T \to T \in \mathbb{U}_1$
  - Also kann x gleichzeitig zu $T \to T$ und zu T gehören
  - Damit $x\, x \in T$ und $\lambda x.\, x\, x \in T \to T = T$ 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 $z \in \mathsf{Void}$ bestimmt sie einen Wert aus T
    - Da es kein $z \in \mathsf{Void}$ 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

Auswertung von Ausdrücken

- **Extensionale Gleichheit macht SN unmöglich**
  - Extensionale Gleichheit ist für Anwendungen sehr wichtig
  - Starke und schwache Normalisierbarkeit sind nicht wirklich erforderlich

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 Paars werden erst bei einem Zugriff bestimmt

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 Paars 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 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 Paars 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

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

Auswertung von Ausdrücken

- **Definiere kanonische Terme**
  - $opid\{P_1, .., P_n\}(b_1, .., b_n)$ 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

Auswertung von Ausdrücken

- **Definiere kanonische Terme**
  - $opid\{P_1, .., P_n\}(b_1, .., b_n)$ 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*

Auswertung von Ausdrücken

- **Definiere kanonische Terme**
  - $opid\{P_1, .., P_n\}(b_1, .., b_n)$ 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

Auswertung von Ausdrücken

- **Definiere kanonische Terme**
  - $opid\{P_1, .., P_n\}(b_1, .., b_n)$ 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

Auswertung von Ausdrücken

- **Definiere kanonische Terme**
  - $opid\{P_1, .., P_n\}(b_1, .., b_n)$ 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

Auswertung von Ausdrücken

- **Definiere kanonische Terme**
  - $opid\{P_1, .., P_n\}(b_1, .., b_n)$ 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)

Operator und Termstruktur	*Standard-Darstellungsform*	*kanonisch?*
var$\{x{:}\mathrm{v}\}()$	x	ja
Nat$\{\}()$	$\mathbb{N}$	ja
zero$\{\}()$	0	ja
suc$\{\}(t)$	$\mathsf{s}(t)$	ja
p_rec$\{\}(\boxed{t}; f; n, x.g)$	$\mathsf{PR}[f,\ n, x \mapsto g](\boxed{t})$	nein
function$\{\}(S; x.T)$	$x : S \longrightarrow T$	ja
lambda$\{\}(x.t)$	$\lambda x.t$	ja
apply$\{\}(\boxed{f}; t)$	$\boxed{f}\ t$	nein
product$\{\}(S; x.T)$	$x : S \times T$	ja
pair$\{\}(s, t)$	$\langle s, t\rangle$	ja
spread$\{\}(\boxed{e}; x, y.u)$	$\mathsf{let}\ \langle x, y\rangle = \boxed{e}\ \mathsf{in}\ u$	nein
equal$\{\}(s; t; T)$	$s = t \in T$	ja
Axiom$\{\}()$	Ax	ja
universe$\{j{:}\mathtt{l}\}()$	$\mathbb{U}_j$	ja

Hauptargumente nichtkanonischer Terme sind umrandet

Redex–Kontrakta Tabelle (Auszug)

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

Algorithmus EVAL(t) für lässige Auswertung

- **Falls t kanonisch** → **Ausgabe t**

Algorithmus EVAL(t) für lässige Auswertung

- **Falls t kanonisch** → **Ausgabe t**
- **Falls t nichtkanonisch:**

Algorithmus EVAL(t) für lässige Auswertung

- **Falls t kanonisch** → **Ausgabe t**
- **Falls t nichtkanonisch:**
  - **Bestimme Hauptargumente $t_1, ..t_n$ von t**

Algorithmus EVAL(t) für lässige Auswertung

- **Falls t kanonisch** → **Ausgabe t**
- **Falls t nichtkanonisch:**
  - **Bestimme Hauptargumente $t_1, ..t_n$ von t**
  - **Berechne s_1:=EVAL(t_1), ..., s_n:=EVAL(t_n)**

Algorithmus EVAL(t) für lässige Auswertung

- **Falls t kanonisch** → **Ausgabe t**
- **Falls t nichtkanonisch:**
  - **Bestimme Hauptargumente $t_1, ..t_n$ von t**
  - **Berechne s_1:=EVAL(t_1), ..., s_n:=EVAL(t_n)**
  - **Bestimme $s := t[s_1, ..s_n/t_1, ..t_n]$** *

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

Algorithmus EVAL(t) für lässige Auswertung

- **Falls t kanonisch** → **Ausgabe t**
- **Falls t nichtkanonisch:**
  - **Bestimme Hauptargumente $t_1, ..t_n$ von t**
  - **Berechne s_1:=EVAL(t_1), ..., s_n:=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 := EVAL(u)** → **Ausgabe v**

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

Algorithmus EVAL(t) für lässige Auswertung

- Falls t kanonisch → Ausgabe t
- Falls t nichtkanonisch:
  - Bestimme Hauptargumente $t_1, ..t_n$ von t
  - Berechne s_1:=EVAL(t_1), ..., s_n:=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 := EVAL(u) → Ausgabe v
  - Falls s kein Redex ist:
    - Stoppe ohne Ergebnis: t besitzt keinen Wert

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

Algorithmus EVAL(t) für lässige Auswertung

- Falls t kanonisch → Ausgabe t
- Falls t nichtkanonisch:
  - Bestimme Hauptargumente $t_1, ..t_n$ von t
  - Berechne s_1:=EVAL(t_1), ..., s_n:=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 := EVAL(u) → Ausgabe v
  - Falls s kein Redex ist:
    - Stoppe ohne Ergebnis: t besitzt keinen Wert

Auswertungsbeispiel

- EVAL((λx.⟨(λy.s(y)) x, 0⟩) 0)

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

Algorithmus EVAL(t) für lässige Auswertung

- **Falls t kanonisch** → **Ausgabe t**
- **Falls t nichtkanonisch:**
  - **Bestimme Hauptargumente $t_1, ..t_n$ von t**
  - **Berechne s_1:=EVAL(t_1), ..., s_n:=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 := EVAL(u)** → **Ausgabe v**
  - **Falls s kein Redex ist:**
    - **Stoppe ohne Ergebnis: t besitzt keinen Wert**

Auswertungsbeispiel

- EVAL((λx.⟨(λy.**s**(y)) x, 0⟩) 0)
- = EVAL(EVAL(λx.⟨(λy.**s**(y)) x,0⟩) 0) Hauptargument kanonisch

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

Algorithmus EVAL(t) für lässige Auswertung

- **Falls t kanonisch** → **Ausgabe t**
- **Falls t nichtkanonisch:**
  - **Bestimme Hauptargumente $t_1, ..t_n$ von t**
  - **Berechne s_1:=EVAL(t_1), ..., s_n:=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 := EVAL(u)** → **Ausgabe v**
  - **Falls s kein Redex ist:**
    - **Stoppe ohne Ergebnis: t besitzt keinen Wert**

Auswertungsbeispiel

- EVAL((λx.⟨(λy.**s**(y)) x, 0⟩) 0)
- = EVAL(**(λx.⟨(λy.s(y)) x,0⟩)** 0) Gesamtterm ist ein Redex

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

Algorithmus EVAL(t) für lässige Auswertung

- **Falls t kanonisch** → **Ausgabe t**
- **Falls t nichtkanonisch:**
  - **Bestimme Hauptargumente $t_1, ..t_n$ von t**
  - **Berechne s_1:=EVAL(t_1), ..., s_n:=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 := EVAL(u)** → **Ausgabe v**
  - **Falls s kein Redex ist:**
    - **Stoppe ohne Ergebnis: t besitzt keinen Wert**

Auswertungsbeispiel

- EVAL((λx.⟨(λy.**s**(y)) x, 0⟩) 0)

= EVAL(⟨(λy.**s**(y)) 0, 0⟩) Kontraktum reduziert λ-Term

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

Algorithmus EVAL(t) für lässige Auswertung

- **Falls t kanonisch** → **Ausgabe t**
- **Falls t nichtkanonisch:**
  - **Bestimme Hauptargumente $t_1, ..t_n$ von t**
  - **Berechne s_1:=EVAL(t_1), ..., s_n:=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 := EVAL(u)** → **Ausgabe v**
  - **Falls s kein Redex ist:**
    - **Stoppe ohne Ergebnis: t besitzt keinen Wert**

Auswertungsbeispiel

- EVAL((λx.⟨(λy.**s**(y)) x, 0⟩) 0)

= ⟨(λy.**s**(y)) 0, 0⟩ Term ist kanonisch

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

Algorithmus EVAL(t) für lässige Auswertung

- **Falls t kanonisch** → **Ausgabe t**
- **Falls t nichtkanonisch:**
  - **Bestimme Hauptargumente $t_1, ..t_n$ von t**
  - **Berechne s_1:=EVAL(t_1), ..., s_n:=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 := EVAL(u)** → **Ausgabe v**
  - **Falls s kein Redex ist:**
    - **Stoppe ohne Ergebnis: t besitzt keinen Wert**

Auswertungsbeispiel

- EVAL((λx.⟨(λy.s(y)) x, 0⟩) 0)

= ⟨(λy.s(y)) 0, 0⟩ 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

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

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

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 **Typ**
  - Ausdruck T bezeichnet einen Typ (Menge)

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 **Typ**
  - Ausdruck T bezeichnet einen Typ (Menge)
- **Typgleichheit** geschrieben als $S{=}T$
  - Ausdrücke S und T bezeichnen den gleichen Typ

Semantik formaler Ausdrücke

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 **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

Iteration von Reduktion und Tabellensuche

- **Typzugehörigkeit**
  - $t \in T$, falls $t{=}t \in T$

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*

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 Typ, falls $T{=}T$

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 $s \stackrel{l}{\longrightarrow} s'$, $t \stackrel{l}{\longrightarrow} t'$, $T{=}T'$ und $s'{=}t' \in T'$ folgt aus $\rightarrow$ *Elementsemantiktabelle*
- **Typ-Sein**
  - T Typ, falls $T{=}T$
- **Typgleichheit**
  - $S{=}T$, falls es kanonische Terme S' und T' gibt mit $S \stackrel{l}{\longrightarrow} S'$, $T \stackrel{l}{\longrightarrow} T'$ und $S'{=}T'$ folgt aus $\rightarrow$ *Typsemantiktabelle*

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 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
$x_1{:}S_1{\to}T_1 = x_2{:}S_2{\to}T_2$	falls $S_1{=}S_2$ und $T_1[s_1/x_1]{=}T_2[s_2/x_2]$ für alle Terme s_1, s_2 mit $s_1{=}s_2 \in S_1$.
$T = S_2{\to}T_2$	falls $T = x_2{:}S_2{\to}T_2$ für ein beliebiges $x_2 \in \mathcal{V}$.
$S_1{\to}T_1 = T$	falls $x_1{:}S_1{\to}T_1 = T$ 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 $T_1[s_1/x_1]{=}T_2[s_2/x_2]$ 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)

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

Semantische Untersuchung von Eigenschaften

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

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

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

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**
  - $(\lambda x.\langle(\lambda y.\mathbf{s}(y))\,x,\,0\rangle)\,0 \xrightarrow{l} \langle(\lambda y.\mathbf{s}(y))\,0,\,0\rangle$
  - $\text{let } \langle x,y\rangle = \langle 0,\mathbf{s}(0)\rangle \text{ in } \langle y,x{*}y\rangle \xrightarrow{l} \langle \mathbf{s}(0),\,0{*}\mathbf{s}(0)\rangle$
  - $\mathbb{N}\times\mathbb{N} \xrightarrow{l} \mathbb{N}\times\mathbb{N}$
- **Semantiktabelle ergibt drei Bedingungen**
  1. $(\lambda y.\mathbf{s}(y))\,0 = \mathbf{s}(0) \in \mathbb{N}$

Semantische Untersuchung von Eigenschaften

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

- **Auswertung der beteiligten Terme**
  - $(\lambda x.\langle(\lambda y.\mathbf{s}(y))\,x,0\rangle)\,0 \xrightarrow{l} \langle(\lambda y.\mathbf{s}(y))\,0,0\rangle$
  - $\mathsf{let}\ \langle x,y\rangle = \langle 0,\mathbf{s}(0)\rangle\ \mathsf{in}\ \langle y,x{*}y\rangle \xrightarrow{l} \langle\mathbf{s}(0),0{*}\mathbf{s}(0)\rangle$
  - $\mathbb{N}\times\mathbb{N} \xrightarrow{l} \mathbb{N}\times\mathbb{N}$
- **Semantiktabelle ergibt drei Bedingungen**
  1. $(\lambda y.\mathbf{s}(y))\,0 = \mathbf{s}(0) \in \mathbb{N}$
     Nach Auswertung $\mathbf{s}(0) = \mathbf{s}(0) \in \mathbb{N}$

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

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

Semantische Untersuchung von Eigenschaften

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

- **Auswertung der beteiligten Terme**
  - $(\lambda x.\langle(\lambda y.\mathtt{s}(y))\,x,0\rangle)\,0 \xrightarrow{l} \langle(\lambda y.\mathtt{s}(y))\,0,0\rangle$
  - $\text{let } \langle x,y\rangle = \langle 0,\mathtt{s}(0)\rangle \text{ in } \langle y,x{*}y\rangle \xrightarrow{l} \langle \mathtt{s}(0),0{*}\mathtt{s}(0)\rangle$
  - $\mathbb{N}\times\mathbb{N} \xrightarrow{l} \mathbb{N}\times\mathbb{N}$
- **Semantiktabelle ergibt drei Bedingungen**
  1. $(\lambda y.\mathtt{s}(y))\,0 = \mathtt{s}(0) \in \mathbb{N}$
     Nach Auswertung $\mathtt{s}(0) = \mathtt{s}(0) \in \mathbb{N}$
     Bedingung laut Tabelle $0 = 0 \in \mathbb{N}$ erfüllt, laut Tabelle
  2. $0 = 0{*}\mathtt{s}(0) \in \mathbb{N}$

Semantische Untersuchung von Eigenschaften

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

- **Auswertung der beteiligten Terme**
  - $(\lambda x.\langle(\lambda y.\mathtt{s}(y))\,x,0\rangle)\,0 \xrightarrow{l} \langle(\lambda y.\mathtt{s}(y))\,0,0\rangle$
  - $\text{let } \langle x,y\rangle = \langle 0,\mathtt{s}(0)\rangle \text{ in } \langle y,x{*}y\rangle \xrightarrow{l} \langle \mathtt{s}(0),0{*}\mathtt{s}(0)\rangle$
  - $\mathbb{N}\times\mathbb{N} \xrightarrow{l} \mathbb{N}\times\mathbb{N}$
- **Semantiktabelle ergibt drei Bedingungen**
  1. $(\lambda y.\mathtt{s}(y))\,0 = \mathtt{s}(0) \in \mathbb{N}$
     Nach Auswertung $\mathtt{s}(0) = \mathtt{s}(0) \in \mathbb{N}$
     Bedingung laut Tabelle $0 = 0 \in \mathbb{N}$ erfüllt, laut Tabelle
  2. $0 = 0{*}\mathtt{s}(0) \in \mathbb{N}$
     Nach Auswertung $0 = 0 \in \mathbb{N}$

Semantische Untersuchung von Eigenschaften

Zeige $(\lambda x.\langle(\lambda y.\mathtt{s}(y))\,x,0\rangle)\,0$
$= \mathsf{let}\ \langle x,y\rangle = \langle 0,\mathtt{s}(0)\rangle\ \mathsf{in}\ \langle y,x{*}y\rangle \in \mathbb{N}\times\mathbb{N}$

- **Auswertung der beteiligten Terme**
  - $(\lambda x.\langle(\lambda y.\mathtt{s}(y))\,x,0\rangle)\,0 \xrightarrow{l} \langle(\lambda y.\mathtt{s}(y))\,0,0\rangle$
  - $\mathsf{let}\ \langle x,y\rangle = \langle 0,\mathtt{s}(0)\rangle\ \mathsf{in}\ \langle y,x{*}y\rangle \xrightarrow{l} \langle \mathtt{s}(0),0{*}\mathtt{s}(0)\rangle$
  - $\mathbb{N}\times\mathbb{N} \xrightarrow{l} \mathbb{N}\times\mathbb{N}$
- **Semantiktabelle ergibt drei Bedingungen**
  1. $(\lambda y.\mathtt{s}(y))\,0 = \mathtt{s}(0) \in \mathbb{N}$
     Nach Auswertung $\mathtt{s}(0) = \mathtt{s}(0) \in \mathbb{N}$
     Bedingung laut Tabelle $0 = 0 \in \mathbb{N}$ erfüllt, laut Tabelle
  2. $0 = 0{*}\mathtt{s}(0) \in \mathbb{N}$
     Nach Auswertung $0 = 0 \in \mathbb{N}$ erfüllt, laut Tabelle

Semantische Untersuchung von Eigenschaften

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

- **Auswertung der beteiligten Terme**
  - $(\lambda x.\langle(\lambda y.\mathtt{s}(y))\,x,0\rangle)\,0 \xrightarrow{l} \langle(\lambda y.\mathtt{s}(y))\,0,0\rangle$
  - $\text{let } \langle x,y\rangle = \langle 0,\mathtt{s}(0)\rangle \text{ in } \langle y,x*y\rangle \xrightarrow{l} \langle \mathtt{s}(0),0*\mathtt{s}(0)\rangle$
  - $\mathbb{N}\times\mathbb{N} \xrightarrow{l} \mathbb{N}\times\mathbb{N}$
- **Semantiktabelle ergibt drei Bedingungen**
  1. $(\lambda y.\mathtt{s}(y))\,0 = \mathtt{s}(0) \in \mathbb{N}$
     Nach Auswertung $\mathtt{s}(0) = \mathtt{s}(0) \in \mathbb{N}$
     Bedingung laut Tabelle $0 = 0 \in \mathbb{N}$ erfüllt, laut Tabelle
  2. $0 = 0*\mathtt{s}(0) \in \mathbb{N}$
     Nach Auswertung $0 = 0 \in \mathbb{N}$ erfüllt, laut Tabelle
  3. $\mathbb{N}\times\mathbb{N}$ Typ

Semantische Untersuchung von Eigenschaften

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

- **Auswertung der beteiligten Terme**
  - $(\lambda x.\langle(\lambda y.\mathtt{s}(y))\,x,0\rangle)\,0 \xrightarrow{l} \langle(\lambda y.\mathtt{s}(y))\,0,0\rangle$
  - $\text{let } \langle x,y\rangle = \langle 0,\mathtt{s}(0)\rangle \text{ in } \langle y,x*y\rangle \xrightarrow{l} \langle \mathtt{s}(0),0*\mathtt{s}(0)\rangle$
  - $\mathbb{N}\times\mathbb{N} \xrightarrow{l} \mathbb{N}\times\mathbb{N}$
- **Semantiktabelle ergibt drei Bedingungen**
  1. $(\lambda y.\mathtt{s}(y))\,0 = \mathtt{s}(0) \in \mathbb{N}$
     Nach Auswertung $\mathtt{s}(0) = \mathtt{s}(0) \in \mathbb{N}$
     Bedingung laut Tabelle $0 = 0 \in \mathbb{N}$ erfüllt, laut Tabelle
  2. $0 = 0*\mathtt{s}(0) \in \mathbb{N}$
     Nach Auswertung $0 = 0 \in \mathbb{N}$ erfüllt, laut Tabelle
  3. $\mathbb{N}\times\mathbb{N}$ `Typ`
     Bedingung laut Tabelle $\mathbb{N}$ `Typ` (zweimal)

Semantische Untersuchung von Eigenschaften

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

- **Auswertung der beteiligten Terme**
  - $(\lambda x.\langle(\lambda y.\mathtt{s}(y))\,x,0\rangle)\,0 \xrightarrow{l} \langle(\lambda y.\mathtt{s}(y))\,0,0\rangle$
  - $\text{let } \langle x,y\rangle = \langle 0,\mathtt{s}(0)\rangle \text{ in } \langle y,x*y\rangle \xrightarrow{l} \langle \mathtt{s}(0),0*\mathtt{s}(0)\rangle$
  - $\mathbb{N}\times\mathbb{N} \xrightarrow{l} \mathbb{N}\times\mathbb{N}$
- **Semantiktabelle ergibt drei Bedingungen**
  1. $(\lambda y.\mathtt{s}(y))\,0 = \mathtt{s}(0) \in \mathbb{N}$
     Nach Auswertung $\mathtt{s}(0) = \mathtt{s}(0) \in \mathbb{N}$
     Bedingung laut Tabelle $0 = 0 \in \mathbb{N}$ erfüllt, laut Tabelle
  2. $0 = 0*\mathtt{s}(0) \in \mathbb{N}$
     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 $\lambda \mathtt{f}.\lambda \mathtt{x}.\mathtt{f}\,\mathtt{x} \in (S \to T) \to S \to T$ für beliebige S, T

Semantische Untersuchung von Eigenschaften II

Zeige $\lambda \mathtt{f}.\lambda \mathtt{x}.\mathtt{f}\,\mathtt{x} \in (S \to T) \to S \to T$ für beliebige S, T

Analysiere Bedingungen der Semantiktabelle

1. $\lambda \mathtt{f}.\lambda \mathtt{x}.\mathtt{f}\,\mathtt{x}$ ist kanonisch und es gilt

Semantische Untersuchung von Eigenschaften II

Zeige $\lambda \mathtt{f}.\lambda \mathtt{x}.\mathtt{f}\,\mathtt{x} \in (S \to T) \to S \to T$ für beliebige S, T

Analysiere Bedingungen der Semantiktabelle

1. $\lambda \mathtt{f}.\lambda \mathtt{x}.\mathtt{f}\,\mathtt{x}$ ist kanonisch und es gilt

   $\lambda \mathtt{f}.\lambda \mathtt{x}.\mathtt{f}\,\mathtt{x} \in (S \to T) \to S \to T$

   falls $\lambda \mathtt{x}.g\,\mathtt{x} \in S \to T$ für alle Terme $g \in S \to T$

Semantische Untersuchung von Eigenschaften II

Zeige $\lambda \mathtt{f}.\lambda \mathtt{x}.\mathtt{f}\,\mathtt{x} \in (S \to T) \to S \to T$ für beliebige S, T

Analysiere Bedingungen der Semantiktabelle

1. $\lambda \mathtt{f}.\lambda \mathtt{x}.\mathtt{f}\,\mathtt{x}$ ist kanonisch und es gilt

   $\lambda \mathtt{f}.\lambda \mathtt{x}.\mathtt{f}\,\mathtt{x} \in (S \to T) \to S \to T$

   falls $\lambda \mathtt{x}.g\,\mathtt{x} \in S \to T$ für alle Terme $g \in S \to T$
2. $\lambda \mathtt{x}.g\,\mathtt{x}$ ist kanonisch und es gilt

Semantische Untersuchung von Eigenschaften II

Zeige $\lambda \mathtt{f}.\lambda \mathtt{x}.\mathtt{f}\,\mathtt{x} \in (S \to T) \to S \to T$ für beliebige S, T

Analysiere Bedingungen der Semantiktabelle

1. $\lambda \mathtt{f}.\lambda \mathtt{x}.\mathtt{f}\,\mathtt{x}$ ist kanonisch und es gilt

   $\lambda \mathtt{f}.\lambda \mathtt{x}.\mathtt{f}\,\mathtt{x} \in (S \to T) \to S \to T$

   falls $\lambda \mathtt{x}.g\,\mathtt{x} \in S \to T$ für alle Terme $g \in S \to T$

2. $\lambda \mathtt{x}.g\,\mathtt{x}$ ist kanonisch und es gilt

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

Semantische Untersuchung von Eigenschaften II

Zeige $\lambda \mathtt{f}.\lambda \mathtt{x}.\mathtt{f}\,\mathtt{x} \in (S \to T) \to S \to T$ für beliebige S, T

Analysiere Bedingungen der Semantiktabelle

1. $\lambda \mathtt{f}.\lambda \mathtt{x}.\mathtt{f}\,\mathtt{x}$ ist kanonisch und es gilt

   $\lambda \mathtt{f}.\lambda \mathtt{x}.\mathtt{f}\,\mathtt{x} \in (S \to T) \to S \to T$

   falls $\lambda \mathtt{x}.g\,\mathtt{x} \in S \to T$ für alle Terme $g \in S \to T$
2. $\lambda \mathtt{x}.g\,\mathtt{x}$ ist kanonisch und es gilt

   $\lambda \mathtt{x}.g\,\mathtt{x} \in S \to T$ falls $g\,s \in T$ für alle Terme $g \in S \to T$, $s \in S$
3. $g\,s$ ist nicht kanonisch und kann nicht reduziert werden

Semantische Untersuchung von Eigenschaften II

Zeige $\lambda \mathtt{f}.\lambda \mathtt{x}.\mathtt{f}\,\mathtt{x} \in (S \to T) \to S \to T$ für beliebige S, T

Analysiere Bedingungen der Semantiktabelle

1. $\lambda \mathtt{f}.\lambda \mathtt{x}.\mathtt{f}\,\mathtt{x}$ ist kanonisch und es gilt

   $\lambda \mathtt{f}.\lambda \mathtt{x}.\mathtt{f}\,\mathtt{x} \in (S \to T) \to S \to T$

   falls $\lambda \mathtt{x}.g\,\mathtt{x} \in S \to T$ für alle Terme $g \in S \to T$

2. $\lambda \mathtt{x}.g\,\mathtt{x}$ ist kanonisch und es gilt

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

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

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

Semantische Untersuchung von Eigenschaften II

Zeige $\lambda \mathtt{f}.\lambda \mathtt{x}.\mathtt{f}\,\mathtt{x} \in (S{\to}T){\to}S{\to}T$ für beliebige S, T

Analysiere Bedingungen der Semantiktabelle

1. $\lambda \mathtt{f}.\lambda \mathtt{x}.\mathtt{f}\,\mathtt{x}$ ist kanonisch und es gilt

   $\lambda \mathtt{f}.\lambda \mathtt{x}.\mathtt{f}\,\mathtt{x} \in (S{\to}T){\to}S{\to}T$

   falls $\lambda \mathtt{x}.g\,\mathtt{x} \in S{\to}T$ für alle Terme $g \in S{\to}T$
2. $\lambda \mathtt{x}.g\,\mathtt{x}$ ist kanonisch und es gilt

   $\lambda \mathtt{x}.g\,\mathtt{x} \in S{\to}T$ falls $g\,s \in T$ für alle Terme $g \in S{\to}T$, $s \in S$
3. $g\,s$ ist nicht kanonisch und kann nicht reduziert werden

   aber $g \in S{\to}T$ 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 II

Zeige $\lambda \mathtt{f}.\lambda \mathtt{x}.\mathtt{f}\,\mathtt{x} \in (S \to T) \to S \to T$ für beliebige S, T

Analysiere Bedingungen der Semantiktabelle

1. $\lambda \mathtt{f}.\lambda \mathtt{x}.\mathtt{f}\,\mathtt{x}$ ist kanonisch und es gilt

   $\lambda \mathtt{f}.\lambda \mathtt{x}.\mathtt{f}\,\mathtt{x} \in (S \to T) \to S \to T$

   falls $\lambda \mathtt{x}.g\,\mathtt{x} \in S \to T$ für alle Terme $g \in S \to T$
2. $\lambda \mathtt{x}.g\,\mathtt{x}$ ist kanonisch und es gilt

   $\lambda \mathtt{x}.g\,\mathtt{x} \in S \to T$ falls $g\,s \in T$ für alle Terme $g \in S \to T$, $s \in S$
3. $g\,s$ ist nicht kanonisch und kann nicht reduziert werden

   aber $g \in S \to T$ 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 $\lambda \mathtt{x}.\mathtt{x}\,\mathtt{x} \notin S \to T$ für jeden (kanonischen) Typ S, T

Semantische Untersuchung von Eigenschaften III

Zeige $\lambda \mathtt{x.xx} \notin S \to T$ für jeden (kanonischen) Typ S, T

Analysiere Bedingungen der Semantiktabelle

1. $\lambda \mathtt{x.xx}$ ist kanonisch und es gilt

Semantische Untersuchung von Eigenschaften III

Zeige $\lambda \mathtt{x.x\,x} \notin S \to T$ für jeden (kanonischen) Typ S, T

Analysiere Bedingungen der Semantiktabelle

1. $\lambda \mathtt{x.x\,x}$ ist kanonisch und es gilt

   $\lambda \mathtt{x.x\,x} \in S \to T$ falls $s\,s \in T$ für alle Terme $s \in S$

Semantische Untersuchung von Eigenschaften III

Zeige $\lambda\mathtt{x.x\,x} \notin S{\to}T$ für jeden (kanonischen) Typ S, T

Analysiere Bedingungen der Semantiktabelle

1. $\lambda\mathtt{x.x\,x}$ ist kanonisch und es gilt

   $\lambda\mathtt{x.x\,x} \in S{\to}T$ 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 $S = Y{\to}Z$ für Typen Y, Z ist und $t[s/y] \in Z$ für alle $s \in Y$ gilt

Semantische Untersuchung von Eigenschaften III

Zeige $\lambda \mathtt{x.x\,x} \notin S{\to}T$ für jeden (kanonischen) Typ S, T

Analysiere Bedingungen der Semantiktabelle

1. $\lambda \mathtt{x.x\,x}$ ist kanonisch und es gilt

   $\lambda \mathtt{x.x\,x} \in S{\to}T$ 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

   $S = Y{\to}Z$ 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 $s \in Y$

Semantische Untersuchung von Eigenschaften III

Zeige $\lambda\mathtt{x.x\,x} \notin S{\to}T$ für jeden (kanonischen) Typ S, T

Analysiere Bedingungen der Semantiktabelle

1. $\lambda\mathtt{x.x\,x}$ ist kanonisch und es gilt

   $\lambda\mathtt{x.x\,x} \in S{\to}T$ 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

   $S = Y{\to}Z$ 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 $S = S{\to}T$ sein

Semantische Untersuchung von Eigenschaften III

Zeige $\lambda\mathtt{x.x\,x} \notin S{\to}T$ für jeden (kanonischen) Typ S, T

Analysiere Bedingungen der Semantiktabelle

1. $\lambda\mathtt{x.x\,x}$ ist kanonisch und es gilt

   $\lambda\mathtt{x.x\,x} \in S{\to}T$ 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

   $S = Y{\to}Z$ 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 $S = S{\to}T$ sein

   Für die Behauptung $S = S{\to}T$ gibt es keine Unterstützung

Semantische Untersuchung von Eigenschaften III

Zeige $\lambda \mathtt{x.x\,x} \notin S{\to}T$ für jeden (kanonischen) Typ S, T

Analysiere Bedingungen der Semantiktabelle

1. $\lambda \mathtt{x.x\,x}$ ist kanonisch und es gilt

   $\lambda \mathtt{x.x\,x} \in S{\to}T$ 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

   $S = Y{\to}Z$ 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 $S = S{\to}T$ sein

   Für die Behauptung $S = S{\to}T$ 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

Semantische Untersuchung von Eigenschaften III

Zeige $\lambda \mathtt{x.x\,x} \notin S{\to}T$ für jeden (kanonischen) Typ S, T

Analysiere Bedingungen der Semantiktabelle

1. $\lambda \mathtt{x.x\,x}$ ist kanonisch und es gilt

   $\lambda \mathtt{x.x\,x} \in S{\to}T$ 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

   $S = Y{\to}Z$ 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 $S = S{\to}T$ sein

   Für die Behauptung $S = S{\to}T$ 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. $\lambda \mathtt{x.x\,x} \in S{\to}T$ **gilt nicht** ✓

Generische Eigenschaften von Urteilen

- **Typgleichheit $S{=}T$ ist transitiv und symmetrisch**
  - aber nicht reflexiv (S und T müssen Typausdrücke sein)

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 $S'{=}T$**

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)

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 $s'{=}t \in T$**

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 $(T{=}T)$**

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 $s{=}t \in S$**

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

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

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

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 $\lambda x.t \in \mathsf{Void} \rightarrow T$ stellt keine "echten" Bedingugen an t
    - Ausdrücke mit $\mathbf{Y}$-Kombinator sind typisierbar, wenn sie terminieren

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 $\lambda x.t \in \mathsf{Void} \rightarrow T$ stellt keine "echten" Bedingugen an t
    - Ausdrücke mit $\mathbf{Y}$-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

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 $\lambda x.t \in \mathsf{Void} \rightarrow T$ stellt keine "echten" Bedingugen an t
    - Ausdrücke mit $\mathbf{Y}$-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

Hypothetische Urteile

Fundamentales Konzept für semantische Argumente

Hypothetische Urteile

Fundamentales Konzept für semantische Argumente

- **Urteil unter Annahmen:** $x_1{:}T_1, \ldots, x_n{:}T_n \models J$
  - Das Urteil J wird gefällt unter der Annahme, daß die x_i Variablen vom Typ T_i sind

Hypothetische Urteile

Fundamentales Konzept für semantische Argumente

- **Urteil unter Annahmen:** $x_1{:}T_1, \ldots, x_n{:}T_n \models J$
  - 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

Hypothetische Urteile

Fundamentales Konzept für semantische Argumente

- **Urteil unter Annahmen:** $x_1{:}T_1,\ldots,x_n{:}T_n \models J$
  - 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 $t_i \in T_i$

Hypothetische Urteile

Fundamentales Konzept für semantische Argumente

- **Urteil unter Annahmen:** $x_1{:}T_1,\ldots,x_n{:}T_n \models J$
  - 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)

Hypothetische Urteile

Fundamentales Konzept für semantische Argumente

- **Urteil unter Annahmen:** $x_1{:}T_1,\ldots,x_n{:}T_n \models J$
  - 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**

Inferenzsystem für CTT

- **Syntaktische Simulation logischer Schlüsse**
  - Formaler Nachweis, daß bestimmte Urteile gefällt werden können
  - Inferenzregeln ersetzten semantische Argumente auf Basis der Tabellen

Inferenzsystem für CTT

- **Syntaktische Simulation logischer Schlüsse**
  - Formaler Nachweis, daß bestimmte Urteile gefällt werden können
  - Inferenzregeln ersetzten semantische Argumente auf Basis der Tabellen
- **Beweise operieren zielorientiert**
  - Oberstes Beweisziel ist die zu beweisende Behauptung ("*Urteil J 'gilt'* ")

Inferenzsystem für CTT

- **Syntaktische Simulation logischer Schlüsse**
  - Formaler Nachweis, daß bestimmte Urteile gefällt werden können
  - Inferenzregeln ersetzten 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

Inferenzsystem für CTT

- **Syntaktische Simulation logischer Schlüsse**
  - Formaler Nachweis, daß bestimmte Urteile gefällt werden können
  - Inferenzregeln ersetzten 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

Inferenzsystem für CTT

- **Syntaktische Simulation logischer Schlüsse**
  - Formaler Nachweis, daß bestimmte Urteile gefällt werden können
  - Inferenzregeln ersetzten 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 $x_1{:}T_1, \ldots x_n{:}T_n \vdash C$

Inferenzsystem für CTT

- **Syntaktische Simulation logischer Schlüsse**
  - Formaler Nachweis, daß bestimmte Urteile gefällt werden können
  - Inferenzregeln ersetzten 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 $x_1{:}T_1, \ldots x_n{:}T_n \vdash C$
  - Urteile werden durch Sequenzen ohne Hypothesen beschrieben

    Oberstes Beweisziel hat die Form $\vdash C$

Inferenzsystem für CTT

- **Syntaktische Simulation logischer Schlüsse**
  - Formaler Nachweis, daß bestimmte Urteile gefällt werden können
  - Inferenzregeln ersetzten 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 $x_1{:}T_1, \ldots x_n{:}T_n \vdash C$
  - 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

Entwurfsentscheidungen für CTT (4a): Beweiskalkül

- **Nur eine syntaktische Klasse für Konklusionen**
  - Vermeidet Aufblähung des syntaktischen Apparates
  - Benötigt einheitliche Repräsentation der vier Urteilsklassen

Entwurfsentscheidungen für CTT (4a): Beweiskalkül

- **Nur eine syntaktische Klasse für Konklusionen**
  - Vermeidet Aufblähung des syntaktischen Apparates
  - Benötigt einheitliche Repräsentation der vier Urteilsklassen
- **Universen repräsentieren Typ-Eigenschaft**
  - Urteile T Typ und $S = T$ werden repräsentiert als $T \in \mathbb{U}_i$ und $S{=}T \in \mathbb{U}_i$

Entwurfsentscheidungen für CTT (4a): Beweiskalkül

- **Nur eine syntaktische Klasse für Konklusionen**
  - Vermeidet Aufblähung des syntaktischen Apparates
  - Benötigt einheitliche Repräsentation der vier Urteilsklassen
- **Universen repräsentieren Typ-Eigenschaft**
  - Urteile T Typ und $S = T$ werden repräsentiert als $T \in \mathbb{U}_i$ und $S{=}T \in \mathbb{U}_i$
  - Kumulative Universenhierarchie $\mathbb{U}{=}\mathbb{U}_1 \in \subset \mathbb{U}_2 \in \subset \mathbb{U}_3 \in \subset \ldots$ erforderlich

Entwurfsentscheidungen für CTT (4a): Beweiskalkül

- **Nur eine syntaktische Klasse für Konklusionen**
  - Vermeidet Aufblähung des syntaktischen Apparates
  - Benötigt einheitliche Repräsentation der vier Urteilsklassen
- **Universen repräsentieren Typ-Eigenschaft**
  - Urteile T Typ und $S = T$ werden repräsentiert als $T \in \mathbb{U}_i$ und $S{=}T \in \mathbb{U}_i$
  - Kumulative Universenhierarchie $\mathbb{U}{=}\mathbb{U}_1 \overset{\in}{\subset} \mathbb{U}_2 \overset{\in}{\subset} \mathbb{U}_3 \overset{\in}{\subset} \ldots$ erforderlich
- **Gleichheit wird dargestellt als Datentyp $s{=}t \in T$**
  - Ermöglicht Verwendung von Gleichheiten in Hypothesen
    - z.B. $h_1 : \mathtt{x{=}4} \in \mathbb{N}$, $h_2 : \mathtt{f(4){=}18} \in \mathbb{N} \vdash \mathtt{f(x){=}18} \in \mathbb{N}$

Entwurfsentscheidungen für CTT (4a): Beweiskalkül

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

Entwurfsentscheidungen für CTT (4a): Beweiskalkül

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

    $\mapsto$ **Propositionen werden als Datentypen beschrieben**

Entwurfsentscheidungen für CTT (4a): Beweiskalkül

- **Nur eine syntaktische Klasse für Konklusionen**
  - Vermeidet Aufblähung des syntaktischen Apparates
  - Benötigt einheitliche Repräsentation der vier Urteilsklassen
- **Universen repräsentieren Typ-Eigenschaft**
  - Urteile T Typ und $S = T$ werden repräsentiert als $T \in \mathbb{U}_i$ und $S{=}T \in \mathbb{U}_i$
  - Kumulative Universenhierarchie $\mathbb{U}{=}\mathbb{U}_1 \overset{\in}{\subset} \mathbb{U}_2 \overset{\in}{\subset} \mathbb{U}_3 \overset{\in}{\subset} \ldots$ erforderlich
- **Gleichheit wird dargestellt als Datentyp $s{=}t \in T$**
  - Ermöglicht Verwendung von Gleichheiten in Hypothesen
    - z.B. $h_1 : \texttt{x=4} \in \mathbb{N}$, $h_2 : \texttt{f(4)=18} \in \mathbb{N} \vdash \texttt{f(x)=18} \in \mathbb{N}$
  - 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 $\mathsf{Ax} \in s{=}t \in T$

Prüfender oder konstruierender Kalkül?

Prüfender oder konstruierender Kalkül?

- **Überprüfung (Verifikation) von Typzugehörigkeit**
  Typ T und Term t sind vorgegeben, Beweis prüft $t \in T$

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

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

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

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

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

  −: Reine Verifikation nicht möglich

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

  −: Reine Verifikation nicht möglich

- **Implizite Darstellung von Elementen** ✓

  – Ersetze $t \in T$ durch $T\ \lfloor \mathbf{ext}\ t \rfloor$ '*T hat ein (noch unbekanntes) Element t*'

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

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

  −: Reine Verifikation nicht möglich

- **Implizite Darstellung von Elementen** ✓

  – Ersetze $t \in T$ durch T ⌊**ext** t⌋ '*T hat ein (noch unbekanntes) Element t*'

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

  – Explizite Verifikation möglich durch Beweis von $t = t \in T$ ⌊**ext** Ax⌋

Unterstützte Formen logischer Schlüsse

- **Verifikation: Überprüfung expliziter Urteile**
  - *Typ-Sein* "T Typ": zeige $\vdash\ T = T \in \mathbb{U}_j\ \lfloor \mathbf{ext}\ p \rfloor$

    (Zeige Existenz eines Terms p und eines Levels j mit $p \in T = T \in \mathbb{U}_j$)
  - *Typgleichheit* "$S{=}T$": zeige $\vdash\ S = T \in \mathbb{U}_j\ \lfloor \mathbf{ext}\ p \rfloor$
  - *Typzugehörigkeit* "$t \in T$": zeige $\vdash\ t = t \in T\ \lfloor \mathbf{ext}\ p \rfloor$
  - *Elementgleichheit* "$s{=}t \in T$": zeige $\vdash\ s = t \in T\ \lfloor \mathbf{ext}\ p \rfloor$

Unterstützte Formen logischer Schlüsse

- **Verifikation: Überprüfung expliziter Urteile**
  - *Typ-Sein* "T Typ": zeige $\vdash\ T = T \in \mathbb{U}_j\ \lfloor \mathbf{ext}\ p \rceil$

    (Zeige Existenz eines Terms p und eines Levels j mit $p\ \in\ T = T \in \mathbb{U}_j$)
  - *Typgleichheit* "$S{=}T$": zeige $\vdash\ S = T \in \mathbb{U}_j\ \lfloor \mathbf{ext}\ p \rceil$
  - *Typzugehörigkeit* "$t \in T$": zeige $\vdash\ t = t \in T\ \lfloor \mathbf{ext}\ p \rceil$
  - *Elementgleichheit* "$s{=}t \in T$": zeige $\vdash\ s = t \in T\ \lfloor \mathbf{ext}\ p \rceil$
- **Synthese: Konstruktion von Termen**
  - *Konstruktion von Datentypen*: zeige $\vdash\ \mathbb{U}_j\ \lfloor \mathbf{ext}\ T \rceil$
  - *Konstruktion von Elementen eines Typs*: zeige $\vdash\ T\ \lfloor \mathbf{ext}\ t \rceil$

    (Zeige Existenz eines Terms t mit $t \in T$)

Beweiskalkül: Präzisierung der Konzepte (I)

- **Sequenz:** $\boldsymbol{x_1{:}T_1, \ldots, x_n{:}T_n \vdash C}$

  "*Unter der Annahme, daß x_i Variablen vom Typ T_i sind, gilt C*"

  - $\boldsymbol{x_i{:}T_i}$ **Deklaration** "x_i ist Variable vom Typ T_i" (x_i Variable, T_i Term)
  - $\boldsymbol{\Gamma \equiv x_1{:}T_1, \ldots, x_n{:}T_n}$ **Hypothesenliste**
  - $\boldsymbol{C}$ **Konklusion** (C Term)

Beweiskalkül: Präzisierung der Konzepte (I)

- **Sequenz:** $\boldsymbol{x_1 : T_1, \ldots, x_n : T_n \vdash C}$

  "*Unter der Annahme, daß x_i Variablen vom Typ T_i sind, gilt C*"

  – $\boldsymbol{x_i : T_i}$ **Deklaration** "x_i ist Variable vom Typ T_i" (x_i Variable, T_i Term)

  – $\boldsymbol{\Gamma \equiv x_1 : T_1, \ldots, x_n : T_n}$ **Hypothesenliste**

  – $\boldsymbol{C}$ **Konklusion** (C Term)

- **Geschlossene Sequenz** $\boldsymbol{\Gamma \vdash C}$

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

Beweiskalkül: Präzisierung der Konzepte (I)

- **Sequenz:** $\boldsymbol{x_1{:}T_1, \ldots, x_n{:}T_n \vdash C}$

  "*Unter der Annahme, daß x_i Variablen vom Typ T_i sind, gilt C*"

  – $\boldsymbol{x_i{:}T_i}$ **Deklaration** "x_i ist Variable vom Typ T_i" (x_i Variable, T_i Term)

  – $\boldsymbol{\Gamma \equiv x_1{:}T_1, \ldots, x_n{:}T_n}$ **Hypothesenliste**

  – $\boldsymbol{C}$ **Konklusion** (C Term)

- **Geschlossene Sequenz** $\boldsymbol{\Gamma \vdash C}$

  – Jede freie Variable x in C oder T_i zuvor durch $x{:}T_j$ deklariert

- **Reine Sequenz** $\boldsymbol{\Gamma \vdash C}$

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

Beweiskalkül: Präzisierung der Konzepte (I)

- **Sequenz:** $\boldsymbol{x_1{:}T_1, \ldots, x_n{:}T_n \vdash C}$

  "*Unter der Annahme, daß x_i Variablen vom Typ T_i sind, gilt C*"

  – $\boldsymbol{x_i{:}T_i}$ **Deklaration** "x_i ist Variable vom Typ T_i" (x_i Variable, T_i Term)

  – $\boldsymbol{\Gamma \equiv x_1{:}T_1, \ldots, x_n{:}T_n}$ **Hypothesenliste**

  – $\boldsymbol{C}$ **Konklusion** (C Term)

- **Geschlossene Sequenz** $\boldsymbol{\Gamma \vdash C}$

  – Jede freie Variable x in C oder T_i zuvor durch $x{:}T_j$ deklariert

- **Reine Sequenz** $\boldsymbol{\Gamma \vdash C}$

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

- **Gültige Sequenz** $\boldsymbol{\Gamma \vdash C}$

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

  – Notation $\boldsymbol{\Gamma \vdash C\ \lfloor ext\ t \rfloor}$, falls t bekannt

Beweiskalkül: Präzisierung der Konzepte (I)

- **Sequenz:** $x_1{:}T_1, \ldots, x_n{:}T_n \vdash C$

  "*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)

  – $\Gamma \equiv x_1{:}T_1, \ldots, x_n{:}T_n$ **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 $\Gamma \vdash C$ ⌊ext t⌋, falls t bekannt

Definition direkt umsetzbar in Implementierung

Implementierung von Sequenzen

Struktur: $\boldsymbol{x_1}:\boldsymbol{T_1},\ldots,\boldsymbol{x_n}:\boldsymbol{T_n} \vdash \boldsymbol{C}$

x_i	Variable,
T_i, C	Term
$x_i:T_i$	Deklaration
$x_1:T_1,\ldots,x_n:T_n$	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

Inferenzregeln simulieren Semantikdefinition

- **Verfeinerung des Beweisziels**

$\Gamma \vdash \lambda x_1.t_1 = \lambda x_2.t_2 \in x{:}S{\to}T$ $\qquad\qquad \lambda x_1.t_1 = \lambda x_2.t_2 \in x{:}S{\to}T$

by `lambdaEquality` j

$\Gamma \vdash S \in \mathbb{U}_j$ $\qquad\qquad$ falls $x{:}S{\to}T$ Typ und

$\Gamma,\ x'{:}S \vdash t_1[x'/x_1] = t_2[x'/x_2] \in T[x'/x]$ $\qquad\qquad t_1[s_1/x_1] = t_2[s_2/x_2] \in T[s_1/x]$

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

Inferenzregeln simulieren Semantikdefinition

- **Verfeinerung des Beweisziels**

$\Gamma \vdash \lambda x_1.t_1 = \lambda x_2.t_2 \in x{:}S{\to}T$ $\qquad\qquad$ $\lambda x_1.t_1 = \lambda x_2.t_2 \in x{:}S{\to}T$

by `lambdaEquality` j

$\Gamma \vdash S \in \mathbb{U}_j$ $\qquad\qquad$ falls $x{:}S{\to}T$ Typ und

$\Gamma,\ x'{:}S \vdash t_1[x'/x_1] = t_2[x'/x_2] \in T[x'/x]$ $\qquad\qquad$ $t_1[s_1/x_1] = t_2[s_2/x_2] \in T[s_1/x]$

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

  - Regel spiegelt Tabelleneintrag in vereinfachter Form wieder
    - $x{:}S{\to}T$ Typ falls S Typ und $T[s_1/x]$ Typ für alle $s_1 \in S$
    - Letzteres ist auch Bedingung für $t_1[s_1/x_1] = t_2[s_2/x_2] \in T[s_1/x]$

Inferenzregeln simulieren Semantikdefinition

- **Verfeinerung des Beweisziels**

  $\Gamma \vdash \lambda x_1.t_1 = \lambda x_2.t_2 \in x{:}S{\to}T$ — $\lambda x_1.t_1 = \lambda x_2.t_2 \in x{:}S{\to}T$

  by `lambdaEquality` j

  $\Gamma \vdash S \in \mathbb{U}_j$ — falls $x{:}S{\to}T$ 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
    · $x{:}S{\to}T$ 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

Inferenzregeln simulieren Semantikdefinition

- **Verfeinerung des Beweisziels**

  $\Gamma \vdash \lambda x_1.t_1 = \lambda x_2.t_2 \in x{:}S{\to}T$ $\qquad\qquad \lambda x_1.t_1 = \lambda x_2.t_2 \in x{:}S{\to}T$

  by `lambdaEquality` j

  $\Gamma \vdash S \in \mathbb{U}_j$ $\qquad\qquad$ falls $x{:}S{\to}T$ Typ und

  $\Gamma,\ x'{:}S \vdash t_1[x'/x_1] = t_2[x'/x_2] \in T[x'/x]$ $\qquad\qquad t_1[s_1/x_1] = t_2[s_2/x_2] \in T[s_1/x]$

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

  – Regel spiegelt Tabelleneintrag in vereinfachter Form wieder

  · $x{:}S{\to}T$ 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**

  $x{:}S{\to}T$ $\lfloor$**ext** $\lambda x'.t\rfloor$

  by `lambdaFormation` j

  $\Gamma,\ x'{:}S \vdash T[x'/x]$ $\lfloor$**ext** $t\rfloor$

  $\Gamma \vdash S \in \mathbb{U}_j$ $\lfloor$Ax$\rfloor$

Inferenzregeln simulieren Semantikdefinition

- **Verfeinerung des Beweisziels**

  $\Gamma \vdash \lambda x_1.t_1 = \lambda x_2.t_2 \in x{:}S{\to}T$ $\qquad\qquad \lambda x_1.t_1 = \lambda x_2.t_2 \in x{:}S{\to}T$

  by `lambdaEquality` j

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

  $\Gamma,\ x'{:}S \vdash t_1[x'/x_1] = t_2[x'/x_2] \in T[x'/x]$ $\qquad$ falls $x{:}S{\to}T$ 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 $s_1{=}s_2 \in S$

  - Regel spiegelt Tabelleneintrag in vereinfachter Form wieder
    - $x{:}S{\to}T$ 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**

  $x{:}S{\to}T$ $\lfloor$**ext** $\lambda x'.t\rfloor$

  by `lambdaFormation` j

  $\Gamma,\ x'{:}S \vdash T[x'/x]$ $\lfloor$**ext** $t\rfloor$

  $\Gamma \vdash S \in \mathbb{U}_j$ $\lfloor$Ax$\rfloor$

  - 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

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

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

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

  Für jede reine Sequenz $Z = \Gamma \vdash C$ gilt
  - Ist $\text{dec}(Z)=[Z_1,..,Z_n]$, dann sind alle Teilziele $Z_i = \Gamma_i \vdash C_i$ rein
  - Die Gültigkeit von Z folgt aus der Gültigkeit aller Z_i
  - Gilt $\Gamma_i \vdash C \lfloor \textbf{ext}\ t_i \rfloor$ für alle Z_i, so folgt $\Gamma \vdash C \lfloor \textbf{ext}\ t \rfloor$, für $t = \text{val}([(Z_1,t_1),\ldots,(Z_n,t_n)])$

Beweiskalkül: Präzisierung der Konzepte (III)

Beweise sind markierte Inferenzbäume

Beweiskalkül: Präzisierung der Konzepte (III)

Beweise sind markierte Inferenzbäume

- **Beweis π mit Wurzel Z** ($Z = \Gamma \vdash C$ Sequenz)
  1. $\pi = Z$ (unvollständiger Beweis) oder
  2. $\pi = (Z, r, [\pi_1,..,\pi_n])$ und
     - $r = (\mathrm{dec},\mathrm{val})$ korrekte Inferenzregel,
     - $\pi_1,\ldots,\pi_n$ Beweise, deren Wurzeln die Teilziele von $\mathrm{dec}(Z)$ sind

Beweiskalkül: Präzisierung der Konzepte (III)

Beweise sind markierte Inferenzbäume

- **Beweis π mit Wurzel Z** ($Z = \Gamma \vdash C$ Sequenz)

  1. $\pi = Z$ (unvollständiger Beweis) oder
  2. $\pi = (Z, r, [\pi_1,..,\pi_n])$ und
     - $r = (\mathrm{dec},\mathrm{val})$ korrekte Inferenzregel,
     - $\pi_1,\ldots,\pi_n$ Beweise, deren Wurzeln die Teilziele von $\mathrm{dec}(Z)$ sind

  π vollständig, falls π ohne unvollständigen Teilbeweise

Beweiskalkül: Präzisierung der Konzepte (III)

Beweise sind markierte Inferenzbäume

- **Beweis π mit Wurzel Z** ($Z = \Gamma \vdash C$ Sequenz)

  1. $\pi = Z$ (unvollständiger Beweis) oder
  2. $\pi = (Z, r, [\pi_1,..,\pi_n])$ und
     - $r = (\text{dec,val})$ korrekte Inferenzregel,
     - $\pi_1,\ldots,\pi_n$ Beweise, deren Wurzeln die Teilziele von $\text{dec}(Z)$ sind

  π vollständig, falls π ohne unvollständigen Teilbeweise

- **Extrakt-Term des vollständigen Beweises π**

  – $\texttt{EXT}(Z, (\text{dec,val}), [\pi_1,..,\pi_n]) = \text{val}([\,(Z_1,\texttt{EXT}(\pi_1)),..,(Z_n,\texttt{EXT}(\pi_n))\,])$

  (Z_i Wurzel von π_i)

Beweiskalkül: Präzisierung der Konzepte (III)

Beweise sind markierte Inferenzbäume

- **Beweis π mit Wurzel Z** ($Z = \Gamma \vdash C$ Sequenz)

  1. $\pi = Z$ (unvollständiger Beweis) oder
  2. $\pi = (Z, r, [\pi_1,..,\pi_n])$ und
     - $r = (\text{dec,val})$ korrekte Inferenzregel,
     - $\pi_1,\ldots,\pi_n$ Beweise, deren Wurzeln die Teilziele von $\text{dec}(Z)$ sind

  π vollständig, falls π ohne unvollständigen Teilbeweise

- **Extrakt-Term des vollständigen Beweises π**
  - $\texttt{EXT}(Z, (\text{dec,val}), [\pi_1,..,\pi_n]) = \text{val}([\,(Z_1,\texttt{EXT}(\pi_1)),..,(Z_n,\texttt{EXT}(\pi_n))\,])$

    (Z_i Wurzel von π_i)

- **Initialsequenz**
  - Geschlossene Sequenz der Form $\vdash C$

Beweiskalkül: Präzisierung der Konzepte (III)

Beweise sind markierte Inferenzbäume

- **Beweis π mit Wurzel Z** ($Z = \Gamma \vdash C$ Sequenz)

  1. $\pi = Z$ (unvollständiger Beweis) oder
  2. $\pi = (Z, r, [\pi_1,..,\pi_n])$ und
     - $r = (\mathrm{dec},\mathrm{val})$ korrekte Inferenzregel,
     - $\pi_1,\ldots,\pi_n$ Beweise, deren Wurzeln die Teilziele von $\mathrm{dec}(Z)$ sind

  π vollständig, falls π ohne unvollständigen Teilbeweise

- **Extrakt-Term des vollständigen Beweises π**
  - $\mathtt{EXT}(Z, (\mathrm{dec},\mathrm{val}), [\pi_1,..,\pi_n]) = \mathrm{val}([\,(Z_1,\mathtt{EXT}(\pi_1)),..,(Z_n,\mathtt{EXT}(\pi_n))\,])$

    (Z_i Wurzel von π_i)

- **Initialsequenz**
  - Geschlossene Sequenz der Form $\vdash C$

- **Theorem**
  - Vollständiger Beweis mit einer Initialsequenz als Wurzel

Implementierung von Regeln und Beweisen

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

- dec Dekomposition: Abbildung von Sequenzen in Listen von Sequenzen
- val Validierung: Abbildung von Listen von Termen und Sequenzen in Terme

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

- Z Sequenz
- r Inferenzregel
- $\pi_1,\ldots,\pi_n$ Beweise, deren Wurzeln die Teilziele von dec(Z) sind

---

```
abstype  rule     = .....
absrectype  proof = sequent # rule # proof list
  with make_proof_node decs t = abs_proof((decs,t), ◇,[])
  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

Systematik für Aufbau des Regelsystems

Vier Klassen von Regeln für jeden Datentyp

- **Typ-Formationsregeln:**
  - Wann sind zwei Typen gleich (*type*`Equality`) $\Gamma \vdash S = T \in \mathbb{U}_j$
  - Wie ist ein Typ zusammengesetzt? (*type*`Formation`) $\Gamma \vdash \mathbb{U}_j \ [\textbf{ext}\ T]$

Systematik für Aufbau des Regelsystems

Vier Klassen von Regeln für jeden Datentyp

- **Typ-Formationsregeln:**
  - Wann sind zwei Typen gleich (*type*`Equality`) $\Gamma \vdash S = T \in \mathbb{U}_j$
  - Wie ist ein Typ zusammengesetzt? (*type*`Formation`) $\Gamma \vdash \mathbb{U}_j \ [\textbf{ext}\ T]$
- **Kanonische Regeln:**
  - Wann sind kanonische Elemente gleich? (*element*`Equality`) $\Gamma \vdash s = t \in T$
  - Wie sind Elemente zusammengesetzt? (*element*`Formation`) $\Gamma \vdash T \ [\textbf{ext}\ t]$

Systematik für Aufbau des Regelsystems

Vier Klassen von Regeln für jeden Datentyp

- **Typ-Formationsregeln:**
  - Wann sind zwei Typen gleich (*type*`Equality`) $\Gamma \vdash S = T \in \mathbb{U}_j$
  - Wie ist ein Typ zusammengesetzt? (*type*`Formation`) $\Gamma \vdash \mathbb{U}_j \ [\textbf{ext}\ T]$
- **Kanonische Regeln:**
  - Wann sind kanonische Elemente gleich? (*element*`Equality`) $\Gamma \vdash s = t \in T$
  - Wie sind Elemente zusammengesetzt? (*element*`Formation`) $\Gamma \vdash T \ [\textbf{ext}\ t]$
- **Nichtkanonische Regeln:**
  - Wann gehört ein Ausdruck zu einem Typ? (*nichtkanonisch*`Equality`) $\Gamma \vdash s = t \in T$
  - Wie kann eine Variable benutzt werden? (*typ*`Elimination`) $\Gamma, x:S, \Delta \vdash T \ [\textbf{ext}\ t]$

Systematik für Aufbau des Regelsystems

Vier Klassen von Regeln für jeden Datentyp

- **Typ-Formationsregeln:**
  - Wann sind zwei Typen gleich (*type*`Equality`) $\Gamma \vdash S = T \in \mathbb{U}_j$
  - Wie ist ein Typ zusammengesetzt? (*type*`Formation`) $\Gamma \vdash \mathbb{U}_j \ [\textbf{ext}\ T]$
- **Kanonische Regeln:**
  - Wann sind kanonische Elemente gleich? (*element*`Equality`) $\Gamma \vdash s = t \in T$
  - Wie sind Elemente zusammengesetzt? (*element*`Formation`) $\Gamma \vdash T \ [\textbf{ext}\ t]$
- **Nichtkanonische Regeln:**
  - Wann gehört ein Ausdruck zu einem Typ? (*nichtkanonisch*`Equality`) $\Gamma \vdash s = t \in T$
  - Wie kann eine Variable benutzt werden? (*typ*`Elimination`) $\Gamma, x:S, \Delta \vdash T \ [\textbf{ext}\ t]$
- **Berechnungsregeln:**
  - Reduktion von Redizes in Gleichheit (*nichtkanonisch*`Reduce*`) $\Gamma \vdash \mathit{redex} = t \in T$

Systematik für Aufbau des Regelsystems

Vier Klassen von Regeln für jeden Datentyp

- **Typ-Formationsregeln:**
  - Wann sind zwei Typen gleich (*type*`Equality`) $\Gamma \vdash S = T \in \mathbb{U}_j$
  - Wie ist ein Typ zusammengesetzt? (*type*`Formation`) $\Gamma \vdash \mathbb{U}_j$ [**ext** T]
- **Kanonische Regeln:**
  - Wann sind kanonische Elemente gleich? (*element*`Equality`) $\Gamma \vdash s = t \in T$
  - Wie sind Elemente zusammengesetzt? (*element*`Formation`) $\Gamma \vdash T$ [**ext** t]
- **Nichtkanonische Regeln:**
  - Wann gehört ein Ausdruck zu einem Typ? (*nichtkanonisch*`Equality`) $\Gamma \vdash s = t \in T$
  - Wie kann eine Variable benutzt werden? (*typ*`Elimination`) $\Gamma, x{:}S, \Delta \vdash T$ [**ext** t]
- **Berechnungsregeln:**
  - Reduktion von Redizes in Gleichheit (*nichtkanonisch*`Reduce*`) $\Gamma \vdash redex = t \in T$
- **Spezielle Regeln**

Beispiel: Regeln für Funktionenraum

$\Gamma \vdash \mathbb{U}_j$ ⌊**ext** $x{:}S{\to}T$⌋
by `dependent_functionFormation` x S
$\Gamma \vdash S \in \mathbb{U}_j$ ⌊Ax⌋
$\Gamma, x{:}S \vdash \mathbb{U}_j$ ⌊**ext** T⌋

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

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

$\Gamma \vdash f_1 t_1 = f_2 t_2 \in T[t_1/x]$ ⌊Ax⌋
by `applyEquality` $x{:}S{\to}T$
$\Gamma \vdash f_1 = f_2 \in x{:}S{\to}T$ ⌊Ax⌋
$\Gamma \vdash t_1 = t_2 \in S$ ⌊Ax⌋

$\Gamma, f{:}x{:}S{\to}T, \Delta \vdash C$ ⌊**ext** $t[f\,s, \mathsf{Ax}/y, z]$⌋
by `dependent_functionElimination` i s y z
$\Gamma, f{:}x{:}S{\to}T, \Delta \vdash s \in S$ ⌊Ax⌋
$\Gamma, f{:}x{:}S{\to}T, y{:}T[s/x], z{:}y{=}f\,s \in T[s/x], \Delta \vdash C$ ⌊**ext** t⌋

$\Gamma \vdash (\lambda x.t)\,s = t_2 \in T$ ⌊Ax⌋
by `applyReduce`
$\Gamma \vdash t[s/x] = t_2 \in T$ ⌊Ax⌋

$\Gamma \vdash \mathbb{U}_j$ ⌊**ext** $S{\to}T$⌋
by `independent_functionFormation`
$\Gamma \vdash \mathbb{U}_j$ ⌊**ext** S⌋
$\Gamma \vdash \mathbb{U}_j$ ⌊**ext** T⌋

$\Gamma \vdash S_1{\to}T_1 = S_2{\to}T_2 \in \mathbb{U}_j$ ⌊Ax⌋
by `independent_functionEquality`
$\Gamma \vdash S_1 = S_2 \in \mathbb{U}_j$ ⌊Ax⌋
$\Gamma \vdash T_1 = T_2 \in \mathbb{U}_j$ ⌊Ax⌋

$\Gamma \vdash x{:}S{\to}T$ ⌊**ext** $\lambda x'.t$⌋
by `lambdaFormation` j x'
$\Gamma, x'{:}S \vdash T[x'/x]$ ⌊**ext** t⌋
$\Gamma \vdash S \in \mathbb{U}_j$ ⌊Ax⌋

$\Gamma, f{:}S{\to}T, \Delta \vdash C$ ⌊**ext** $t[f\,s, /y]$⌋
by `independent_functionElimination` i y
$\Gamma, f{:}S{\to}T, \Delta \vdash S$ ⌊**ext** s⌋
$\Gamma, f{:}S{\to}T, y{:}T, \Delta \vdash C$ ⌊**ext** t⌋

$\Gamma \vdash f_1 = f_2 \in x{:}S{\to}T$ ⌊**ext** t⌋
by `functionExtensionality` j $x_1{:}S_1{\to}T_1$ $x_2{:}S_2{\to}T_2$ x'
$\Gamma, x'{:}S \vdash f_1 x' = f_2 x' \in T[x'/x]$ ⌊**ext** t⌋
$\Gamma \vdash S \in \mathbb{U}_j$ ⌊Ax⌋
$\Gamma \vdash f_1 \in x_1{:}S_1{\to}T_1$ ⌊Ax⌋
$\Gamma \vdash f_2 \in x_2{:}S_2{\to}T_2$ ⌊Ax⌋

Inferenzregeln brauchen Steuerungsparameter

- **Position i einer zu verwendenden Hypothese**

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

Inferenzregeln brauchen Steuerungsparameter

- **Position i einer zu verwendenden Hypothese**

$$\Gamma, x{:}T, \Delta \vdash T \quad \lfloor \textbf{ext}\ x \rfloor$$
by `hypothesis` i

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

$$\Gamma \vdash x_1{:}S_1 \to T_1 = x_2{:}S_2 \to T_2 \in \mathbb{U}_j \quad \lfloor \mathsf{Ax} \rfloor$$
by `functionEquality` x
$$\Gamma \vdash S_1 = S_2 \in \mathbb{U}_j \quad \lfloor \mathsf{Ax} \rfloor$$
$$\Gamma, x{:}S_1 \vdash T_1[x/x_1] = T_2[x/x_2] \in \mathbb{U}_j \quad \lfloor \mathsf{Ax} \rfloor$$

Inferenzregeln brauchen Steuerungsparameter

- **Position i einer zu verwendenden Hypothese**

  $\Gamma, x{:}T, \Delta \vdash T$ ⌊**ext** x⌋
  by `hypothesis` i

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

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

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

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

Steuerungsparameter für Inferenzregeln (II)

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

$\Gamma,\ f\colon x{:}S\to T,\ \Delta \vdash C$ ⌊**ext** $t[fs, \mathsf{Ax}/y, z]$⌋

by `dependent_functionElimination` i s y z

$\Gamma,\ f\colon x{:}S\to T,\ \Delta \vdash s \in S$ ⌊Ax⌋

$\Gamma,\ f\colon x{:}S\to T,\ y{:}T[s/x],\ z\colon y{=}f\ s \in T[s/x],\ \Delta \vdash C$ ⌊**ext** t⌋

Steuerungsparameter für Inferenzregeln (II)

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

  $\Gamma, f{:}\,x{:}S{\to}T, \Delta \vdash C$ ⌊**ext** $t[fs, \mathsf{Ax}/y, z]$⌋
  by `dependent_functionElimination` i s y z
  $\Gamma, f{:}\,x{:}S{\to}T, \Delta \vdash s \in S$ ⌊Ax⌋
  $\Gamma, f{:}\,x{:}S{\to}T, y{:}T[s/x], z{:}\,y{=}f\ s \in T[s/x], \Delta \vdash C$ ⌊**ext** t⌋

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

  $\Gamma \vdash f_1 t_1 = f_2 t_2 \in T[t_1/x]$ ⌊Ax⌋
  by `applyEquality` $x{:}S{\to}T$
  $\Gamma \vdash f_1 = f_2 \in x{:}S{\to}T$ ⌊Ax⌋
  $\Gamma \vdash t_1 = t_2 \in S$ ⌊Ax⌋

Steuerungsparameter für Inferenzregeln (II)

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

  $\Gamma, f{:}\,x{:}S{\to}T, \Delta \vdash C$ ⌊**ext** $t[fs, \mathsf{Ax}/y, z]$⌋
  by `dependent_functionElimination` i s y z
  $\Gamma, f{:}\,x{:}S{\to}T, \Delta \vdash s \in S$ ⌊Ax⌋
  $\Gamma, f{:}\,x{:}S{\to}T, y{:}T[s/x], z{:}\,y{=}f\ s \in T[s/x], \Delta \vdash C$ ⌊**ext** t⌋

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

  $\Gamma \vdash f_1 t_1 = f_2 t_2 \in T[t_1/x]$ ⌊Ax⌋
  by `applyEquality` $x{:}S{\to}T$
  $\Gamma \vdash f_1 = f_2 \in x{:}S{\to}T$ ⌊Ax⌋
  $\Gamma \vdash t_1 = t_2 \in S$ ⌊Ax⌋

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

  $\Gamma \vdash C[s/x]$ ⌊**ext** u⌋
  by `substitution` j $s{=}t \in T$ x C
  $\Gamma \vdash s{=}t \in T$ ⌊Ax⌋
  $\Gamma \vdash C[t/x]$ ⌊**ext** u⌋
  $\Gamma, x{:}T \vdash C \in \mathbb{U}_j$ ⌊Ax⌋

Vereinfachungen des Regelsystems

- **Vollständiges Regelsystem wird sehr umfangreich**
  - Durchschnittlich 9 Regeln pro Datentyp
  - Namensvielfalt trotz Systematik zu groß und schwer zu merken

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

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

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

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"*

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**

Zusammenfassung: Entwurfsentscheidungen

- **CTT wird als erweiterbare Theorie formalisiert**
- **CTT ist eine Grundlagentheorie mit direkter Semantik**

Zusammenfassung: Entwurfsentscheidungen

- **CTT wird als erweiterbare Theorie formalisiert**
- **CTT ist eine Grundlagentheorie mit direkter Semantik**
- **Abstraktion und Darstellungsform von Termen wird getrennt**

Zusammenfassung: Entwurfsentscheidungen

- CTT wird als erweiterbare Theorie formalisiert
- CTT ist eine Grundlagentheorie mit direkter Semantik
- Abstraktion und Darstellungsform von Termen wird getrennt
- Die Semantik von CTT basiert auf Lazy Evaluation

Zusammenfassung: Entwurfsentscheidungen

- CTT wird als erweiterbare Theorie formalisiert
- CTT ist eine Grundlagentheorie 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

Zusammenfassung: Entwurfsentscheidungen

- CTT wird als erweiterbare Theorie formalisiert
- CTT ist eine Grundlagentheorie 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

Zusammenfassung: Entwurfsentscheidungen

- CTT wird als erweiterbare Theorie formalisiert
- CTT ist eine Grundlagentheorie 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

Zusammenfassung: Entwurfsentscheidungen

- CTT wird als erweiterbare Theorie formalisiert
- CTT ist eine Grundlagentheorie 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

Zusammenfassung: Entwurfsentscheidungen

- CTT wird als erweiterbare Theorie formalisiert
- CTT ist eine Grundlagentheorie 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 $T\ [\mathsf{ext}\ t]$ dargestellt

Zusammenfassung: Entwurfsentscheidungen

- CTT wird als erweiterbare Theorie formalisiert
- CTT ist eine Grundlagentheorie 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 $T \lfloor \mathsf{ext}\ t \rfloor$ 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

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

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

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

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

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

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

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:	$x : S \to T$	**function**{}$(S\,;\, x\,.\,T)$
	$\lambda x\,.\,e$	**lambda**{}$(x\,.\,e)$
Nichtkanonisch:	$\boxed{e_1}\ e_2$	**apply**{}$(\boxed{e_1}\,;e_2)$

Anhang: Übersicht (abhängiger) Funktionenraum

Zentrales Konzept für Schließen über Berechnung

Syntax:

Kanonisch:	$x : S \to T$	**function**{}$(S\,;\, x\,.\,T)$
	$\lambda x\,.\,e$	**lambda**{}$(x\,.\,e)$
Nichtkanonisch:	$\boxed{e_1}\ e_2$	**apply**{}$(\boxed{e_1}\,;e_2)$

Auswertung:

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

Anhang: Übersicht (abhängiger) Funktionenraum

Zentrales Konzept für Schließen über Berechnung

Syntax:

Kanonisch:	$x:S \to T$	**function**{}$(S\,;\, x\,.\,T)$
	$\lambda x\,.\,e$	**lambda**{}$(x\,.\,e)$
Nichtkanonisch:	$\boxed{e_1}\ e_2$	**apply**{}$(\boxed{e_1}\,;e_2)$

Auswertung:

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

Semantik:

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

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

Anhang: Übersicht (abhängiger) Funktionenraum

Zentrales Konzept für Schließen über Berechnung

Syntax:

Kanonisch:	$x:S \to T$	**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:

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

$$\lambda x_1.t_1 = \lambda x_2.t_2 \in x{:}S \to T \equiv x{:}S \to T \text{ Typ und } t_1[s_1/x_1] = t_2[s_2/x_2] \in T[s_1/x]$$
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

$\Gamma \vdash \mathbb{U}_j$ ⌊**ext** $x\!:\!S\!\to\!T$⌋
by `dependent_functionFormation` x S
$\Gamma \vdash S \in \mathbb{U}_j$ ⌊Ax⌋
$\Gamma, x\!:\!S \vdash \mathbb{U}_j$ ⌊**ext** T⌋

$\Gamma \vdash x_1\!:\!S_1\!\to\!T_1 = x_2\!:\!S_2\!\to\!T_2 \in \mathbb{U}_j$ ⌊Ax⌋
by `functionEquality` x
$\Gamma \vdash S_1 = S_2 \in \mathbb{U}_j$ ⌊Ax⌋
$\Gamma, x\!:\!S_1 \vdash T_1[x/x_1] = T_2[x/x_2] \in \mathbb{U}_j$ ⌊Ax⌋

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

$\Gamma \vdash f_1 t_1 = f_2 t_2 \in T[t_1/x]$ ⌊Ax⌋
by `applyEquality` $x\!:\!S\!\to\!T$
$\Gamma \vdash f_1 = f_2 \in x\!:\!S\!\to\!T$ ⌊Ax⌋
$\Gamma \vdash t_1 = t_2 \in S$ ⌊Ax⌋

$\Gamma, f\!:\!x\!:\!S\!\to\!T, \Delta \vdash C$ ⌊**ext** $t[f\,s, \mathsf{Ax}/y, z]$⌋
by `dependent_functionElimination` i s y z
$\Gamma, f\!:\!x\!:\!S\!\to\!T, \Delta \vdash s \in S$ ⌊Ax⌋
$\Gamma, f\!:\!x\!:\!S\!\to\!T, y\!:\!T[s/x], z\!:\!y\!=\!f\,s \in T[s/x], \Delta \vdash C$ ⌊**ext** t⌋

$\Gamma \vdash (\lambda x.t)\,s = t_2 \in T$ ⌊Ax⌋
by `applyReduce`
$\Gamma \vdash t[s/x] = t_2 \in T$ ⌊Ax⌋

$\Gamma \vdash \mathbb{U}_j$ ⌊**ext** $S\!\to\!T$⌋
by `independent_functionFormation`
$\Gamma \vdash \mathbb{U}_j$ ⌊**ext** S⌋
$\Gamma \vdash \mathbb{U}_j$ ⌊**ext** T⌋

$\Gamma \vdash S_1\!\to\!T_1 = S_2\!\to\!T_2 \in \mathbb{U}_j$ ⌊Ax⌋
by `independent_functionEquality`
$\Gamma \vdash S_1 = S_2 \in \mathbb{U}_j$ ⌊Ax⌋
$\Gamma \vdash T_1 = T_2 \in \mathbb{U}_j$ ⌊Ax⌋

$\Gamma \vdash x\!:\!S\!\to\!T$ ⌊**ext** $\lambda x'.t$⌋
by `lambdaFormation` j x'
$\Gamma, x'\!:\!S \vdash T[x'/x]$ ⌊**ext** t⌋
$\Gamma \vdash S \in \mathbb{U}_j$ ⌊Ax⌋

$\Gamma, f\!:\!S\!\to\!T, \Delta \vdash C$ ⌊**ext** $t[f\,s, /y]$⌋
by `independent_functionElimination` i y
$\Gamma, f\!:\!S\!\to\!T, \Delta \vdash S$ ⌊**ext** s⌋
$\Gamma, f\!:\!S\!\to\!T, y\!:\!T, \Delta \vdash C$ ⌊**ext** t⌋

$\Gamma \vdash f_1 = f_2 \in x\!:\!S\!\to\!T$ ⌊**ext** t⌋
by `functionExtensionality` j $x_1\!:\!S_1\!\to\!T_1$ $x_2\!:\!S_2\!\to\!T_2$ x'
$\Gamma, x'\!:\!S \vdash f_1 x' = f_2 x' \in T[x'/x]$ ⌊**ext** t⌋
$\Gamma \vdash S \in \mathbb{U}_j$ ⌊Ax⌋
$\Gamma \vdash f_1 \in x_1\!:\!S_1\!\to\!T_1$ ⌊Ax⌋
$\Gamma \vdash f_2 \in x_2\!:\!S_2\!\to\!T_2$ ⌊Ax⌋

Übersicht (abhängiges) Kartesisches Produkt

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

Syntax:

Kanonisch:	$x : S \times T$	**product**{}(S; x.T)
	$\langle e_1, e_2 \rangle$	**pair**{}(e_1, e_2)
Nichtkanonisch:	let $\langle x, y \rangle = \boxed{e}$ in u	**spread**{}($\boxed{e}$; x, y.u)

Übersicht (abhängiges) Kartesisches Produkt

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

Syntax:

Kanonisch:	$x : S \times T$	**product**$\{\}(S\,;\, x\,.\,T)$
	$\langle e_1, e_2 \rangle$	**pair**$\{\}(e_1, e_2)$
Nichtkanonisch:	let $\langle x\,, y \rangle = \boxed{e}$ in u	**spread**$\{\}(\boxed{e}\,;\, x\,, y\,.\,u)$

Auswertung:

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

Übersicht (abhängiges) Kartesisches Produkt

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

Syntax:

Kanonisch:	$x:S\times T$	**product**$\{\}(S\,;\,x\,.\,T)$
	$\langle e_1, e_2\rangle$	**pair**$\{\}(e_1, e_2)$
Nichtkanonisch:	$\mathsf{let}\ \langle x\,,y\rangle = \boxed{e}\ \mathsf{in}\ u$	**spread**$\{\}(\boxed{e}\,;\,x\,,y\,.\,u)$

Auswertung:

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

Semantik:

$x_1{:}S_1{\times}T_1 = x_2{:}S_2{\times}T_2 \equiv S_1{=}S_2$ 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 $x_1 \in \mathcal{V}$

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

Übersicht (abhängiges) Kartesisches Produkt

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

Syntax:

Kanonisch:	$x:S\times T$	**product**$\{\}(S;\, x.T)$
	$\langle e_1, e_2\rangle$	**pair**$\{\}(e_1, e_2)$
Nichtkanonisch:	let $\langle x, y\rangle = \boxed{e}$ in u	**spread**$\{\}(\boxed{e};\, x, y.u)$

Auswertung:

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

Semantik:

$x_1{:}S_1\times T_1 = x_2{:}S_2\times T_2$	$\equiv$	$S_1{=}S_2$ 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 $x_1 \in \mathcal{V}$
$\langle s_1, t_1\rangle = \langle s_2, t_2\rangle \in x:S\times T$	$\equiv$	$x:S\times T$ Typ und $s_1{=}s_2 \in S$ 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

$\Gamma \vdash \mathbb{U}_j$ ⌊**ext** $x{:}S{\times}T$⌋
by `dependent_productFormation` x S
$\Gamma \vdash S \in \mathbb{U}_j$ ⌊Ax⌋
$\Gamma, x{:}S \vdash \mathbb{U}_j$ ⌊**ext** T⌋

$\Gamma \vdash x_1{:}S_1{\times}T_1 = x_2{:}S_2{\times}T_2 \in \mathbb{U}_j$ ⌊Ax⌋
by `productEquality` x'
$\Gamma \vdash S_1 = S_2 \in \mathbb{U}_j$ ⌊Ax⌋
$\Gamma, x'{:}S_1 \vdash T_1[x'/x_1] = T_2[x'/x_2] \in \mathbb{U}_j$ ⌊Ax⌋

$\Gamma \vdash \langle s_1, t_1\rangle = \langle s_2, t_2\rangle \in x{:}S{\times}T$ ⌊Ax⌋
by `dependent_pairEquality` j x'
$\Gamma \vdash s_1 = s_2 \in S$ ⌊Ax⌋
$\Gamma \vdash t_1 = t_2 \in T[s_1/x]$ ⌊Ax⌋
$\Gamma, x'{:}S \vdash T[x'/x] \in \mathbb{U}_j$ ⌊Ax⌋

$\Gamma \vdash \langle s_1, t_1\rangle = \langle s_1, t_1\rangle \in S{\times}T$ ⌊Ax⌋
by `independent_pairEquality`
$\Gamma \vdash s_1 = s_2 \in S$ ⌊Ax⌋
$\Gamma \vdash t_1 = t_2 \in T$ ⌊Ax⌋

$\Gamma \vdash \mathsf{let}\ \langle x_1, y_1\rangle = e_1\ \mathsf{in}\ t_1 = \mathsf{let}\ \langle x_2, y_2\rangle = e_2\ \mathsf{in}\ t_2 \in C[e_1/z]$ ⌊Ax⌋
by `spreadEquality` z C $x{:}S{\times}T$ s t y
$\Gamma \vdash e_1 = e_2 \in x{:}S{\times}T$ ⌊Ax⌋
$\Gamma, s{:}S, t{:}T[s/x], y{:}\, e_1{=}\langle s,t\rangle \in x{:}S{\times}T \vdash t_1[s,t/x_1,y_1] = t_2[s,t/x_2,y_2] \in C[\langle s,t\rangle/z]$ ⌊Ax⌋

$\Gamma, z{:}\, x{:}S{\times}T, \Delta \vdash C$ ⌊**ext** $\mathsf{let}\ \langle s,t\rangle = z\ \mathsf{in}\ u$⌋
by `productElimination` i s t
$\Gamma, z{:}\, x{:}S{\times}T, s{:}S, t{:}T[s/x]\ \Delta[\langle s,t\rangle/z] \vdash C[\langle s,t\rangle/z]$ ⌊**ext** u⌋

$\Gamma \vdash \mathbb{U}_j$ ⌊**ext** $S{\times}T$⌋
by `independent_productFormation`
$\Gamma \vdash \mathbb{U}_j$ ⌊**ext** S⌋
$\Gamma \vdash \mathbb{U}_j$ ⌊**ext** T⌋

$\Gamma \vdash S_1{\times}T_1 = S_2{\times}T_2 \in \mathbb{U}_j$ ⌊Ax⌋
by `independent_productEquality`
$\Gamma \vdash S_1 = S_2 \in \mathbb{U}_j$ ⌊Ax⌋
$\Gamma \vdash T_1 = T_2 \in \mathbb{U}_j$ ⌊Ax⌋

$\Gamma \vdash x{:}S{\times}T$ ⌊**ext** $\langle s,t\rangle$⌋
by `dependent_pairFormation` j s x'
$\Gamma \vdash s \in S$ ⌊Ax⌋
$\Gamma \vdash T[s/x]$ ⌊**ext** t⌋
$\Gamma, x'{:}S \vdash T[x'/x] \in \mathbb{U}_j$ ⌊Ax⌋

$\Gamma \vdash S{\times}T$ ⌊**ext** $\langle s,t\rangle$⌋
by `independent_pairFormation`
$\Gamma \vdash S$ ⌊**ext** s⌋
$\Gamma \vdash T$ ⌊**ext** t⌋

$\Gamma \vdash \mathsf{let}\ \langle x,y\rangle = \langle s,t\rangle\ \mathsf{in}\ u = t_2 \in T$ ⌊Ax⌋
by `spreadReduce`
$\Gamma \vdash u[s,t/x,y] = t_2 \in T$ ⌊Ax⌋

Anhang: Übersicht Gleichheit

Schließen über Werte von Ausdrücken

Syntax:

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

Anhang: Übersicht Gleichheit

Schließen über Werte von Ausdrücken

Syntax:

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

Ax **Axiom**{}()

Nichtkanonisch: —

Auswertung: —

Anhang: Übersicht Gleichheit

Schließen über Werte von Ausdrücken

Syntax:

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

Ax **Axiom**{}()

Nichtkanonisch: —

Auswertung: —

Semantik:

$s_1 = t_1 \in T_1 = s_2 = t_2 \in T_2 \equiv T_1 = T_2,\ s_1 = s_2 \in T_1,$ und $t_1 = t_2 \in T_1$

Ax $=$ Ax $\in$ $s = t \in T \equiv s = t \in T$

Anhang: Übersicht Gleichheit

Schließen über Werte von Ausdrücken

Syntax:

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

Ax **Axiom**{}()

Nichtkanonisch: —

Auswertung: —

Semantik:

$s_1 = t_1 \in T_1 = s_2 = t_2 \in T_2 \equiv T_1 = T_2,\ s_1 = s_2 \in T_1,$ 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: —

Anhang: Übersicht Universen

Repräsentation der Typ-Eigenschaft

Syntax:

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

Nichtkanonisch: —

Auswertung: —

Repräsentation der Typ-Eigenschaft

Syntax:

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

Nichtkanonisch: —

Auswertung: —

Semantik:

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

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

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

$\vdots \quad \equiv$ — *analog für alle anderen Datentypen* —

Anhang: Übersicht Universen

Repräsentation der Typ-Eigenschaft

Syntax:

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

Nichtkanonisch: —

Auswertung: —

Semantik:

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

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

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

$\vdots$ $\equiv$ — *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
$\Gamma \vdash T \in \mathbb{U}_j$ ⌊Ax⌋
$\Gamma \vdash T$ ⌊**ext** s⌋
$\Gamma \vdash T$ ⌊**ext** t⌋

$\Gamma \vdash \mathsf{Ax} = \mathsf{Ax} \in s = t \in T$ ⌊Ax⌋
by `axiomEquality`
$\Gamma \vdash s = t \in T$ ⌊Ax⌋

$\Gamma, z\colon s = t \in T, \Delta \vdash C$ ⌊**ext** u⌋
by `equalityElimination` i
$\Gamma, z\colon s = t \in T, \Delta[\mathsf{Ax}/z] \vdash C[\mathsf{Ax}/z]$ ⌊**ext** u⌋

$\Gamma, x{:}T, \Delta \vdash x = x \in T$ ⌊Ax⌋
by `hypothesisEquality` i

$\Gamma \vdash \mathbb{U}_k$ ⌊**ext** $\mathbb{U}_j$⌋
by `universeFormation` j *

$\Gamma \vdash T \in \mathbb{U}_k$ ⌊Ax⌋
by `cumulativity` j *
$\Gamma \vdash T \in \mathbb{U}_j$ ⌊Ax⌋

*: *proviso* $j < k$

$\Gamma \vdash s_1 = t_1 \in T_1 = s_2 = t_2 \in T_2 \in \mathbb{U}_j$ ⌊Ax⌋
by `equalityEquality`
$\Gamma \vdash T_1 = T_2 \in \mathbb{U}_j$ ⌊Ax⌋
$\Gamma \vdash s_1 = s_2 \in T_1$ ⌊Ax⌋
$\Gamma \vdash t_1 = t_2 \in T_1$ ⌊Ax⌋

$\Gamma \vdash C[s/x]$ ⌊**ext** u⌋
by `substitution` j $s{=}t \in T$ x C
$\Gamma \vdash s{=}t \in T$ ⌊Ax⌋
$\Gamma \vdash C[t/x]$ ⌊**ext** u⌋
$\Gamma, x{:}T \vdash C \in \mathbb{U}_j$ ⌊Ax⌋

$\Gamma \vdash s = t \in T$ ⌊Ax⌋
by `equality` *Entscheidungsprozedur*

$\Gamma \vdash \mathbb{U}_j = \mathbb{U}_j \in \mathbb{U}_k$ ⌊Ax⌋
by `universeEquality` *

Anhang: Regeln für Natürliche Zahlen

$\Gamma \vdash \mathbb{U}_j$ ⌊**ext** $\mathbb{N}$⌋
by natFormation

$\Gamma \vdash \mathbb{N} \in \mathbb{U}_j$ ⌊Ax⌋
by natEquality

$\Gamma \vdash 0 = 0 \in \mathbb{N}$ ⌊Ax⌋
by zeroEquality

$\Gamma \vdash \mathbb{N}$ ⌊**ext** 0⌋
by zeroFormation

$\Gamma \vdash \mathtt{s}(t_1) = \mathtt{s}(t_2) \in \mathbb{N}$ ⌊Ax⌋
by succEquality

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

$\Gamma \vdash \mathbb{N}$ ⌊**ext** $\mathtt{s}(t)$⌋
by succFormation

$\Gamma \vdash \mathbb{N}$ ⌊**ext** t⌋

$\Gamma \vdash \mathsf{PR}[f_1,\ n_1, x_1 \mapsto g_1](t_1) = \mathsf{PR}[f_2,\ n_2, x_2 \mapsto g_2](t_2) \in T[t_1/z]$ ⌊Ax⌋
by prEquality z T n h_n

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

$\Gamma \vdash f_1 = f_2 \in T[0/z]$ ⌊Ax⌋

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

$\Gamma, z{:}\mathbb{N}, \Delta \vdash C$ ⌊**ext** $\mathsf{PR}[f,\ n, x \mapsto g](z)$⌋
by natElimination i n h_n

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

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

$\Gamma \vdash \mathsf{PR}[f,\ n, x \mapsto g](0) = t_2 \in T$ ⌊Ax⌋
by prReduceBase

$\Gamma \vdash f = t_2 \in T$ ⌊Ax⌋

$\Gamma \vdash \mathsf{PR}[f,\ n, x \mapsto g](\mathtt{s}(\mathtt{t})) = t_2 \in T$ ⌊Ax⌋
by prReduceUp

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

Anhang: Weitere Regeln

$\Gamma, x{:}T, \Delta \vdash T$ ⌊**ext** x⌋
by `hypothesis` i

$\Gamma, \Delta \vdash C$ ⌊**ext** $(\lambda x.\, t)\ s$⌋
by `cut` i T x
$\Gamma, \Delta \vdash T$ ⌊**ext** s⌋
$\Gamma, x{:}T, \Delta \vdash C$ ⌊**ext** t⌋

$\Gamma, z{:}T, \Delta \vdash C$ ⌊**ext** t⌋
by `hyp_replacement` i S j
$\Gamma, z{:}S, \Delta \vdash C$ ⌊**ext** t⌋
$\Gamma, z{:}T, \Delta \vdash T = S \in \mathbb{U}_j$ ⌊Ax⌋

$\Gamma \vdash C$ ⌊**ext** t⌋
by `lemma` "*theorem-name*"

$\Gamma \vdash C$ ⌊**ext** $t[\sigma]$⌋
by `instantiate` Γ' C' σ
$\Gamma' \vdash C'$ ⌊**ext** t⌋

$\Gamma, x{:}T, \Delta \vdash C$ ⌊**ext** t⌋
by `thin` i
$\Gamma, \Delta \vdash C$ ⌊**ext** t⌋

$\Gamma \vdash T$ ⌊**ext** t⌋
by `introduction` t
$\Gamma \vdash t \in T$ ⌊Ax⌋

$\Gamma \vdash t \in T$ ⌊Ax⌋
by `extract` "*theorem-name*"

$\Gamma \vdash C$ ⌊**ext** $t[y/x]$⌋
by `rename` y x
$\Gamma[x/y] \vdash C[x/y]$ ⌊**ext** t⌋