

Automatisierte Logik und Programmierung

Einheit 11



Fortgeschrittene Konzepte der Typentheorie



1. “Nichtkonstruktive” Datentypen
2. Neuere Konstrukte
3. Ausblick

● Mengentheoretische Operationen

- Teilmengenbildung $\{ x:T \mid P[x] \}$
- Vereinigung $S \cup T$ und Durchschnitt $S \cap T$ zweier Mengen
- Vereinigung/Durchschnitt von Mengenfamilien $\bigcup_{x \in S} T[x]$, $\bigcap_{x \in S} T[x]$
- Elemente dieser Typen sind Elemente der ursprünglichen Mengen
- Informationen über Herkunft oder Selektionsprädikat sind **nur implizit vorhanden** (und nicht etwa in den Elementen gespeichert)

● Äquivalenzklassen

- Äquivalenzrelation E induziert Klassen von Elementen eines Typs T
- $[x]_E$ ist die Menge der zu x unter E äquivalenten Elemente
- Menge der Klassen bildet einen neuen Typ $T // E$
- Elemente von T sind Repräsentanten der Elemente von $T // E$
- E induziert Gleichheit auf $T // E$: $[x]_E = [y]_E$ gilt g.d.w. $E(x, y)$

Mit bisherigen Typkonstrukten nicht simulierbar

Konservative Definition gängiger Grundkonstrukte

- **Singulärer Typ:** $\mathbf{Unit} \equiv 0 \in \mathbb{Z}$
 - Einziges Element ist \mathbf{Ax}
 - Andere Bezeichnung: \mathbf{t} (“der Wahrheitstyp” Gegenstück zu \mathbf{f})
- **Boolescher Datentyp:** $\mathbb{B} \equiv \mathbf{Unit} + \mathbf{Unit}$
 - $\mathbf{T} \equiv \text{inl}(\mathbf{Ax})$, $\mathbf{F} \equiv \text{inr}(\mathbf{Ax})$
 - $\text{if } b \text{ then } s \text{ else } t \equiv \text{case } b \text{ of } \text{inl}(x) \mapsto s \mid \text{inr}(y) \mapsto t$
 - Einbettung boolescher Werte in logische Prädikate durch **Lifting**:
 $\uparrow b \equiv \text{if } b \text{ then } t \text{ else } f$
- **Rekursive Funktionen definiert durch Y-Kombinator**
 - $\mathbf{Y} \equiv \lambda f. (\lambda x. f (x x)) (\lambda x. f (x x))$
 - $\text{function } f(x) = t \equiv \mathbf{Y}(\lambda f. \lambda x. t)$ (geschlossener Term für f)
 - Nicht jede rekursiv definierte Funktion ist typisierbar
Typ von $\text{function } f(x) = t$ muß im Kontext nachgewiesen werden

TEILMENGENKONSTRUKT $\{x : S \mid P[x]\}$

- **Standardkonzept mit klarer intuitiver Bedeutung**
 - Ermöglicht natürliche Repräsentation von Teiltypen (\mathbb{N} , T $List^+$, ...) ohne ständige explizite Verwendung einschränkender Prädikate
 - z.B. $\forall n : \mathbb{N}. \exists r : \mathbb{N}. r^2 \leq n < (r+1)^2$ anstatt
 $\forall n : \mathbb{Z}. 0 \leq n \Rightarrow \exists r : \mathbb{Z}. 0 \leq r \wedge r^2 \leq n < (r+1)^2$
 - **Konstruktive Interpretation von $\{x : S \mid P[x]\}$ schwierig**
 - Elemente sind Elemente aus $s \in S$, welche Eigenschaft $P[s]$ besitzen
 - Evidenz für $P[s]$ nur implizit vorhanden (kein Bestandteil des Elements)
 - **Formale Ähnlichkeit zu $x : S \times P[x]$ nutzt wenig**
 - Elemente von $x : S \times P[x]$ sind Paare $\langle s, pf \rangle$ mit $s \in \{x : S \mid P[x]\}$
 - Evidenz $pf \in P[s]$ bleibt Bestandteil des Elements
 - Führt zu unnatürlichen Spezifikationen vieler Algorithmen
 - Nullstellen finden $\{f : \mathbb{N} \rightarrow \mathbb{N} \mid \exists r : \mathbb{N}. f(r) = 0\} \rightarrow \mathbb{N}$ (Suche nötig)
 - würde zu $f : (\mathbb{N} \rightarrow \mathbb{N}) \times \exists r : \mathbb{N}. f(r) = 0 \rightarrow \mathbb{N}$ (Projektion reicht)
- $x : S \times P[x]$ ist zur Beschreibung von Teilmengen nicht geeignet**

- **Formalisierung der mengentheoretischen Aspekte**

- Elemente von $\{x : S \mid P[x]\}$ sind Elemente von S
- Es muß Evidenz für $P[s]$ geben, die aber nicht explizit bekannt ist

- **Sonderform des unabhängigen Konstruktors $\{S \mid P\}$**

- Identisch mit S , wenn es Evidenz für P gibt, sonst leer

- **Häufigste Anwendung: Type-Squashing**

- Definiere $\downarrow P \equiv \{t \mid P\}$

Reduziert Struktur des Prädikats (Typs) P auf den “Wahrheitstyp” t

- Wenn P nicht leer ist, enthält $\downarrow P$ nur das Element Ax

Die Struktur der Elemente von P wird entfernt

Alle Elemente von $\downarrow P$ sind gleich und besagen “ P besitzt Evidenz”

- Entfernt Überstrukturierung aus benutzerdefinierten Prädikaten

(siehe Folie 24)

FORMULIERUNG DES TEILMENGENTYPS

• Ausdrücke

Kanonisch: $\{x:S \mid T\}$ S, T Terme, x Variable
 $\{S \mid T\}$ S, T Terme

Nichtkanonisch: —

• Reduktion von Ausdrücken

– entfällt, da keine nichtkanonische Elemente für den Teilmengenkonstruktor definiert

• Urteile für Typ- und Elementgleichheit

$\{x_1:S_1 \mid T_1\} = \{x_2:S_2 \mid T_2\}$ falls $S_1=S_2$ und es gibt Terme p_1, p_2 und eine Variable x , die weder in T_1 noch in T_2 vorkommt, so daß
 $p_1 \in \forall x:S_1. T_1[x/x_1] \Rightarrow T_2[x/x_2]$
und $p_2 \in \forall x:S_1. T_2[x/x_2] \Rightarrow T_1[x/x_1]$

$\{x_1:S_1 \mid T_1\} = \{x_2:S_2 \mid T_2\} \in \mathbb{U}_j$ analog

Prinzip der getrennten Definition von Typen kann nicht eingehalten werden

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

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

$s = t \in \{x:S \mid T\}$ falls $\{x:S \mid T\}$ Typ und $s = t \in S$ und es gibt einen Term p mit der Eigenschaft $p \in T[s/x]$

- **Zahlenmengen und -intervalle:**

$$\mathbb{N} \equiv \{i:\mathbb{Z} \mid 0 \leq i\}$$

$$\mathbb{N}^+ \equiv \{i:\mathbb{Z} \mid 0 < i\}$$

$$\{i \dots\} \equiv \{j:\mathbb{Z} \mid i \leq j\}$$

$$\{\dots i\} \equiv \{j:\mathbb{Z} \mid j \leq i\}$$

$$\{i \dots j\} \equiv \{k:\mathbb{Z} \mid i \leq k \leq j\}$$

$$\{i \dots j^-\} \equiv \{k:\mathbb{Z} \mid i \leq k < j\}$$

- **Listen**

$$T \text{ list}^+ \equiv \{l:T \text{ list} \mid l \neq [] \in T \text{ list}\}$$

BEHANDLUNG VON TEILMENGEN IM INFERENZSYSTEM

- **Verwaltung von implizitem Wissen erforderlich**

- $\{x:T \mid P\}$ hat mehr Information als T und weniger als $x:T \times P$

Nullstellenbestimmung ist verschieden aufwendig für Elemente von

$\mathbb{N} \rightarrow \mathbb{N}$, $\{f:\mathbb{N} \rightarrow \mathbb{N} \mid \exists r:\mathbb{N}. f(r)=0\}$ und $f:(\mathbb{N} \rightarrow \mathbb{N}) \times \exists r:\mathbb{N}. f(r)=0$

- Für $s \in \{x:T \mid P\}$ wissen wir, daß $P[s]$ gilt, aber wir wissen nicht, wie die Evidenz für $P[s]$ konkret aussieht

- In Beweisen müssen wir das Wissen $P[s]$ verwenden können, ohne die Evidenz für $P[s]$ algorithmisch einzusetzen

Evidenz für $P[s]$ darf nicht im Extrakterm vorkommen

- **Unterstütze versteckte Hypothesen**

- Erzeugung durch Dekomposition der Annahme $z: \{x:S \mid T\}$

$H, z: \{x:S \mid T\}, H' \vdash C$	$\text{[ext } (\lambda y.t) z]$
$\text{setElimination } i \ y \ v$	
$H, z: \{x:S \mid T\}, y:S, \llbracket v \rrbracket:T[y/x], H'[y/z] \vdash C[y/z]$	$\text{[ext } t]$

- Freigabe der Hypothese nur in Teilzielen mit Extrakterm Ax

(Gleichheiten, Kleiner-Relation)

REGELN FÜR TEILMENGENTYPEN

$$\begin{array}{l}
 H \vdash \{x_1:S_1 \mid T_1\} = \{x_2:S_2 \mid T_2\} \in \mathbb{U}_j \quad \text{[Ax]} \\
 \text{setEquality } x \\
 H \vdash S_1 = S_2 \in \mathbb{U}_j \quad \text{[Ax]} \\
 H, x:S_1 \vdash T_1[x/x_1] = T_2[x/x_2] \in \mathbb{U}_j \quad \text{[Ax]}
 \end{array}$$

$$\begin{array}{l}
 H \vdash \mathbb{U}_j \quad \text{[ext } \{x:S \mid T\}] \\
 \text{setFormation } S \ x \\
 H \vdash S \in \mathbb{U}_j \quad \text{[Ax]} \\
 H, x:S \vdash \mathbb{U}_j \quad \text{[ext } T]
 \end{array}$$

$$\begin{array}{l}
 H \vdash s = t \in \{x:S \mid T\} \quad \text{[Ax]} \\
 \text{set_memberEquality } j \ x' \\
 H \vdash s = t \in S \quad \text{[Ax]} \\
 H \vdash T[s/x] \quad \text{[Ax]} \\
 H, x':S \vdash T[x'/x] \in \mathbb{U}_j \quad \text{[Ax]}
 \end{array}$$

$$\begin{array}{l}
 H \vdash \{x:S \mid T\} \quad \text{[ext } s] \\
 \text{set_memberFormation } j \ s \ x' \\
 H \vdash s \in S \quad \text{[Ax]} \\
 H \vdash T[s/x] \quad \text{[Ax]} \\
 H, x':S \vdash T[x'/x] \in \mathbb{U}_j \quad \text{[Ax]}
 \end{array}$$

$$\begin{array}{l}
 H, z: \{x:S \mid T\}, H' \vdash C \quad \text{[ext } (\lambda y.t) \ z] \\
 \text{setElimination } i \ y \ v \\
 H, z: \{x:S \mid T\}, y:S, \llbracket v \rrbracket : T[y/x], H'[y/z] \vdash C[y/z] \quad \text{[ext } t]
 \end{array}$$

*Die Regel **setEquality** ist restriktiver als die Semantik auf Folie 5.
Das macht sie unvollständig aber leichter anzuwenden.*

- **$\{x : S \mid P[x]\}$ verhält sich wie ein Existenzquantor**
 - $\{x : S \mid P[x]\}$ ist genau dann nicht leer, wenn $\exists x:S.P[x]$ gültig ist
 - Elemente $s \in \{x : S \mid P[x]\}$ sind Zeugen für Gültigkeit von $\exists x:S.P[x]$
Evidenz für $P[s]$ existiert, wird aber unterdrückt
 - $\{x : S \mid P[x]\}$ beschreibt nur den algorithmischen Aspekt des Quantors
 - \forall - \exists -Theorem auf Basis des Teilmengentyps liefern bessere Extraktterme
- **$\{P \mid Q\}$ verhält sich ähnlich wie die Konjunktion $P \wedge Q$**
 - $\{P \mid Q\}$ hat Elemente, wenn P und Q gültig sind
 - Elemente $p \in \{P \mid Q\}$ sind Elemente von P (partielle Evidenz)
Evidenz für Q existiert, wird aber unterdrückt
 - $\{P \mid Q\}$ beschreibt eine nichtkommutative Konjunktion
 - $\{P \mid \mathbf{t}\}$ ist isomorph zu P
 - $\{\mathbf{t} \mid Q\}$ hat genau dann Elemente, wenn dies für Q gilt
aber alle Elemente von $\{\mathbf{t} \mid Q\}$ sind gleich (“Squashing”)

QUADRATWURZELALGORITHMUS MIT TEILMENGENTYPEN

```
⊢ ∀n:ℕ. {r:ℕ | r2 ≤ n < (r+1)2}           allR
1. n:ℕ ⊢ {r:ℕ | r2 ≤ n < (r+1)2}           NatInd4 1
1.1. ⊢ {r:ℕ | r2 ≤ 0 < (r+1)2}             set_memberR [0] THEN Auto ✓
1.2. i:ℕ, r:ℕ, r2 ≤ i ÷ 4 < (r+1)2 ⊢ {r:ℕ | r2 ≤ i < (r+1)2}
                                           Decide [(2*r+1)2 ≤ i] THEN Auto
1.2.1. i:ℕ, r:ℕ, r2 ≤ i ÷ 4 < (r+1)2, (2*r+1)2 ≤ i ⊢ {r:ℕ | r2 ≤ i < (r+1)2}
                                           set_memberR [2*r+1] THEN Auto ✓
1.2.2. i:ℕ, r:ℕ, r2 ≤ i ÷ 4 < (r+1)2, ¬((2*r+1)2 ≤ i) ⊢ {r:ℕ | r2 ≤ i < (r+1)2}
                                           set_memberR [2*r] THEN Auto ✓
```

• Beweis hat die gleiche Struktur wie zuvor

- `set_memberR` ist Taktik für `set_memberFormation`
- Versteckte Hypothese $r^2 \leq i \div 4 < (r+1)^2$ wird für die Beweisziele $r^2 \leq i$ und $i < (r+1)^2$ freigegeben,
- Extrakt-Term enthält keine Beweiskomponenten
- Effizienter Algorithmus für $\lfloor \sqrt{n} \rfloor$ wird ohne Projektion generiert

- **Durchschnitt einer Familie von Typen $T[x]$ mit $x \in S$**
 - Verallgemeinerung des einfachen Durchschnitts $S \cap T$ zweier Typen
 - Elemente sind Objekte, die für alle $x \in S$ zum Typ $T[x]$ gehören
 - Keine Abhängigkeit der Elemente von der Wahl eines konkreten $x \in S$
- **Strukturell ähnlich zu $x:S \rightarrow T[x]$ (aber andere Semantik)**
 - Elemente von $x:S \rightarrow T[x]$ konstruieren für alle $x \in S$ ein $y \in T[x]$
 - $x \in S$ als Argument der Funktionen aus $x:S \rightarrow T[x]$ angegeben werden
 - Starke Abhängigkeit von $y \in T[x]$ von der Wahl des konkreten $x \in S$
- **Viele Anwendungen**
 - Record-Strukturen in Programmiersprachen
 - Definition eines Typs aller Terme der Typentheorie
 - Repräsentation eines echten Polymorphismus (ohne Abhängigkeiten)
 - Polymorphe Logik
 - “Guarded” Types

FORMULIERUNG DES DURCHSCHNITTSTYPS

- **Ausdrücke**

Kanonisch: $\cap x:S.T$ x Variable, S und T Terme

Nichtkanonisch: —

- **Reduktion von Ausdrücken**

– entfällt, da keine nichtkanonische Elemente für den Teilmengenkonstruktor definiert

- **Urteile für Typ- und Elementgleichheit**

$\cap x_1:S_1.T_1 = \cap x_2:S_2.T_2$ 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 $s_1=s_2 \in S_1$.

$\cap x_1:S_1.T_1 = \cap x_2:S_2.T_2 \in \mathbb{U}_j$ analog

$t_1 = t_2 \in \cap x:S.T$ falls $\cap x:S.T$ Typ und
 $t_1=t_2 \in T[s/x]$ für alle Terme $s \in S$

REGELN FÜR DURCHSCHNITTSBILDUNG

$$\begin{array}{l}
 H \vdash \cap x_i : S_i . T_1 = \cap x_i : S_i . T_2 \in \mathbb{U}_j \quad \text{[Ax]} \\
 \text{isectEquality } x \\
 H \vdash S_1 = S_2 \in \mathbb{U}_j \quad \text{[Ax]} \\
 H, x : S_1 \vdash T_1[x/x_1] = T_2[x/x_2] \in \mathbb{U}_j \quad \text{[Ax]}
 \end{array}$$

$$\begin{array}{l}
 H \vdash \mathbb{U}_j \quad \text{[ext } \cap x : S . T \text{]} \\
 \text{isectFormation } x \ S \\
 H \vdash S \in \mathbb{U}_j \quad \text{[Ax]} \\
 H, x : S \vdash \mathbb{U}_j \quad \text{[ext } T \text{]}
 \end{array}$$

$$\begin{array}{l}
 H \vdash t_1 = t_2 \in \cap x : S . T \quad \text{[Ax]} \\
 \text{isect_memberEquality } j \ x' \\
 H, x' : S \vdash t_1 = t_2 \in T[x'/x] \quad \text{[Ax]} \\
 H \vdash S \in \mathbb{U}_j \quad \text{[Ax]}
 \end{array}$$

$$\begin{array}{l}
 H \vdash \cap x : S . T \quad \text{[ext } t \text{]} \\
 \text{isect_memberFormation } j \ x' \\
 H, x' : S \vdash T[x'/x] \quad \text{[ext } t \text{]} \\
 H \vdash S \in \mathbb{U}_j \quad \text{[Ax]}
 \end{array}$$

$$\begin{array}{l}
 H \vdash f_1 = f_2 \in T[t/x] \quad \text{[Ax]} \\
 \text{isect_member_caseEquality } \cap x : S . T \ t \\
 H \vdash f_1 = f_2 \in \cap x : S . T \quad \text{[Ax]} \\
 H \vdash t \in S \quad \text{[Ax]}
 \end{array}$$

$$\begin{array}{l}
 H, f : \cap x : S . T, H' \vdash C \quad \text{[ext } t[f, \text{Ax}/y, z] \text{]} \\
 \text{isectElimination } i \ s \ y \ z \\
 H, f : \cap x : S . T, H' \vdash s \in S \quad \text{[Ax]} \\
 H, f : \cap x : S . T, y : T[s/x], z : y = f \in T[s/x], H' \vdash C \quad \text{[ext } t \text{]}
 \end{array}$$

ANWENDUNGEN DES DURCHSCHNITTSTYPS

- **Einfacher (binärer) Durchschnitt** $S \cap T$

- Durchschnitt einer Mengenfamilie mit booleschem Index

$$S \cap T \equiv \cap b:\mathbb{B}. \text{ if } b \text{ then } S \text{ else } T$$

Achtung: $S \cap T$ ist nicht identisch mit $x:S \cap T$ mit T unabhängig von x

- **Top: Typ aller Terme der Typentheorie**

- Leerer Durchschnitt hinterläßt keine Bedingungen an Elemente

$$\text{Top} \equiv \cap x:\{\}. \{\}$$

- Wegen $t \in \text{Top}$, falls $t \in \{\}$ für alle $x \in \{\}$ gehört jeder Term t zu Top

- **Record Strukturen** $\{l_1:T_1; \dots; l_n:T_n\}$

- Durchschnitt einer Familie von Funktionen auf Labels l_i (Folie 26)

- $\{l_1:T_1; \dots; l_n:T_n\} \equiv \cap l:\text{Atom}. \text{ if } l=l_1 \text{ then } T_1 \text{ else } \dots \text{ if } l=l_n \text{ then } T_n \text{ else Top}$

- **Guarded types** $\cap x:S.T$, wobei T unabhängig von x

- Isomorph zu Top, wenn S leer ist, und sonst isomorph zu T

- Nützlich für Wohlformtheitsbeweise zu Mengenaussagen, die in pathologischen Fällen nicht einmal Typen wären (z.B. $x \in S \Rightarrow x \in T$)

- $\bigcap x:S.P[x]$ verhält sich wie ein Allquantor
 - $\bigcap x:S.P[x]$ ist genau dann nicht leer, wenn $\forall x:S.P[x]$ gültig ist
 - Elemente $p \in \bigcap x:S.P[x]$ sind **uniforme Evidenz** für alle $P[s]$
 - Anders als bei $\forall x:S.P[x]$ wird Evidenz für $P[s]$ nicht aus s konstruiert
algorithmischer Aspekt des Allquantors entfällt
 - Definition $\forall[x:S].P[x] \equiv \bigcap x:S.P[x]$ liefert **polymorphe Logik**
Viele Quantoren in Theoremen können polymorph gewählt werden
z.B. $\forall[i:\mathbb{Z}]. \forall[j:\mathbb{Z}]. i+j = j+i \in \mathbb{Z}$
- **Polymorphe Typdeklarationen sind möglich**
 - z.B. besagt $f \in \bigcap T:\mathbb{U}. T \rightarrow T$, daß f auf beliebigen Typen operiert
 - Typparameter T muß in Deklaration von f nicht verwendet werden
- $P \cap Q$ verhält sich wie Konjunktion
 - $P \cap Q$ ist genau dann nicht leer, wenn $P \wedge Q$ gültig ist
 - Elemente $p \in P \cap Q$ sind **uniforme Evidenz** für alle P und Q
 - Keine Paarbildung erforderlich

VEREINIGUNGSKONSTRUKTOR $\cup x : S . T[x]$

- **Vereinigung einer Familie von Typen $T[x]$ mit $x \in S$**
 - Verallgemeinerung der einfachen Vereinigung $S \cup T$ zweier Typen
 - Elemente sind Objekte y , die für ein $x \in S$ zum Typ $T[x]$ gehören
 - Keine Angabe eines konkreten $x \in S$ erforderlich
- **Strukturell ähnlich zu $x : S \times T[x]$** (aber andere Semantik)
 - Elemente von $x : S \times T[x]$ sind Paare (x, y) mit $x \in S$ und ein $y \in T[x]$
 - Starke Abhängigkeit von $y \in T[x]$ von der Wahl des konkreten $x \in S$
 - $x \in S$ muß als Teil des Elements mit angegeben werden
- **Nichttriviale Formalisierung**
 - Bei nicht disjunkten Typen muß Urteil der Elementgleichheit im Überlappungsbereich auf allen Typen übereinstimmen

- $\cup x:S.P[x]$ verhält sich wie ein Existenzquantor
 - $\cup x:S.P[x]$ ist genau dann nicht leer, wenn $\exists x:S.P[x]$ gültig ist
 - Elemente $p \in \cup x:S.P[x]$ sind Evidenz für ein $P[s]$
 - Anders als bei $\exists x:S.P[x]$ fehlt der Zeuge $s \in S$ für den $P[s]$ gilt
algorithmischer Aspekt des Existenzquantors entfällt
 - $\exists[x:S].P[x] \equiv \cup x:S.P[x]$ liefert semi-klassischen Existenzquantor
- $P \cup Q$ verhält sich wie Disjunktion
 - $P \cup Q$ ist genau dann nicht leer, wenn $P \vee Q$ gültig ist
 - Elemente $p \in P \cup Q$ sind Evidenz für P oder für Q
 - Keine Kennzeichnung, welche der beiden Aussagen bewiesen wurde
 - $P[\vee]Q \equiv P \cup Q$ liefert semi-klassische Disjunktion

QUOTIENTENTYPEN

- **Modifikation der Gleichheit auf Typen**

- Rationale Zahlen: Paare ganzer Zahlen mit $\langle z_1, n_1 \rangle = \langle z_2, n_2 \rangle$,
falls $z_1 * n_2 = z_2 * n_1$
- Reelle Zahlen: (rationale) Cauchyfolgen mit gleichem Grenzwert
- Restklassenräume: $\mathbb{Z} \bmod k$

- **Entspricht Faktorisierung modulo einer Äquivalenz**

- Elemente s, t werden aus Typ T ausgewählt
- Gleichheit von s und t wird über Äquivalenzrelation E neu definiert
- Benutzerdefinierte Gleichheit wird in das Typsystem eingebettet
Substitutions- und Gleichheitsregeln werden direkt anwendbar



Quotiententypen wichtig für formale Mathematik

QUOTIENTENTYPEN, FORMAL

- **Ausdrücke**

Kanonisch: $x, y : T // E$ E, T Terme, x, y Variablen

Nichtkanonisch: —

- **Reduktion von Ausdrücken**

– entfällt, da keine nichtkanonische Elemente für Quotiententypen definiert

- **Urteile für Typ- und Elementgleichheit**

$x_1, y_1 : T_1 // E_1$

$= x_2, y_2 : T_2 // E_2$

falls

$T_1 = T_2$ und es gibt (verschiedene) Variablen x, y, z , die weder in E_1 noch in E_2 vorkommen, und Terme p_1, p_2, r, s und t mit der Eigenschaft

$p_1 \in \forall x : T_1 . \forall y : T_1 . E_1[x, y/x_1, y_1] \Rightarrow E_2[x, y/x_2, y_2]$

und $p_2 \in \forall x : T_1 . \forall y : T_1 . E_2[x, y/x_2, y_2] \Rightarrow E_1[x, y/x_1, y_1]$

und $r \in \forall x : T_1 . E_1[x, x/x_1, y_1]$

und $s \in \forall x : T_1 . \forall y : T_1 . E_1[x, y/x_1, y_1] \Rightarrow E_1[y, x/x_1, y_1]$

und $t \in \forall x : T_1 . \forall y : T_1 . \forall z : T_1 .$

$E_1[x, y/x_1, y_1] \Rightarrow E_1[y, z/x_1, y_1] \Rightarrow E_1[x, z/x_1, y_1]$

$s = t \in x, y : T // E$ falls $x, y : T // E$ Typ und $s \in T$ und $t \in T$ und es gibt einen Term p mit der Eigenschaft $p \in E[s, t/x, y]$

BEHANDLUNG VON QUOTIENTEN IM INFERENZSYSTEM

- **Gleichheit $E[s, t / x, y]$ ist implizites Wissen**

- Wir wissen $E[s, t/x, y]$ wenn $s = t \in x, y : T // E$
- In Beweisen müssen wir das Wissen $E[s, t/x, y]$ verwenden können, ohne die Evidenz für $E[s, t/x, y]$ algorithmisch einzusetzen

Evidenz für $E[s, t/x, y]$ darf nicht im Extraktterm vorkommen

- **Versteckte Hypothesen erforderlich**

- Dekomposition einer Quotientengleichheit muß versteckte Hypothesen generieren

$H, v: s = t \in x, y : T // E, H' \vdash C$	[ext u]
<code>quotient_equalityElimination</code> i j v'	
$H, v: s = t \in x, y : T // E, \llbracket v' \rrbracket : E[s, t/x, y], H' \vdash C$	[ext u]
$H, v: s = t \in x, y : T // E, H' \vdash E[s, t/x, y] \in \mathbb{U}_j$	[Ax]

- Freigabe versteckter Hypothesen wie zuvor

REGELN FÜR QUOTIENTENTYPEN

$H \vdash \mathbb{U}_j$	[ext $x, y : T // E$]
quotientFormation $T E x y z v v'$	
$H \vdash T \in \mathbb{U}_j$	[Ax]
$H, x:T, y:T \vdash E \in \mathbb{U}_j$	[Ax]
$H, x:T, \vdash E[x, x/x, y]$	[Ax]
$H, x:T, y:T, v: E[x, y/x, y] \vdash E[y, x/x, y]$	[Ax]
$H, x:T, y:T, z:T, v: E[x, y/x, y], v': E[y, z/x, y] \vdash E[x, z/x, y]$	[Ax]

$H \vdash x_1, y_1 : T_1 // E_1 = x_2, y_2 : T_2 // E_2 \in \mathbb{U}_j$	[Ax]
quotientWeakEquality $x y z v v'$	
$H \vdash T_1 = T_2 \in \mathbb{U}_j$	[Ax]
$H, x:T_1, y:T_1 \vdash E_1[x, y/x_1, y_1] = E_2[x, y/x_2, y_2] \in \mathbb{U}_j$	[Ax]
$H, x:T_1 \vdash E_1[x, x/x_1, y_1]$	[Ax]
$H, x:T_1, y:T_1, v: E_1[x, y/x_1, y_1] \vdash E_1[y, x/x_1, y_1]$	[Ax]
$H, x:T_1, y:T_1, z:T_1, v: E_1[x, y/x_1, y_1], v': E_1[y, z/x_1, y_1] \vdash E_1[x, z/x_1, y_1]$	[Ax]

$H \vdash x_1, y_1 : T_1 // E_1 = x_2, y_2 : T_2 // E_2 \in \mathbb{U}_j$	[Ax]
quotientEquality	
$H \vdash x_1, y_1 : T_1 // E_1 \in \mathbb{U}_j$	[Ax]
$H \vdash x_2, y_2 : T_2 // E_2 \in \mathbb{U}_j$	[Ax]
$H \vdash T_1 = T_2 \in \mathbb{U}_j$	[Ax]
$H, v: T_1 = T_2 \in \mathbb{U}_j, x:T_1, y:T_1 \vdash E_1[x, y/x_1, y_1] \Rightarrow E_2[x, y/x_2, y_2]$	[Ax]
$H, v: T_1 = T_2 \in \mathbb{U}_j, x:T_1, y:T_1 \vdash E_2[x, y/x_2, y_2] \Rightarrow E_1[x, y/x_1, y_1]$	[Ax]

$H \vdash s = t \in x, y : T // E$	[Ax]
quotient_memberWeakEquality j	
$H \vdash x, y : T // E \in \mathbb{U}_j$	[Ax]
$H \vdash s = t \in T$	[Ax]

$H \vdash x, y : T // E$	[ext t]
quotient_memberFormation j	
$H \vdash x, y : T // E \in \mathbb{U}_j$	[Ax]
$H \vdash T$	[ext t]

REGELN FÜR QUOTIENTENTYPEN II

$H \vdash s = t \in x, y : T // E$	Ax
$\text{quotient_memberEquality } j$	
$H \vdash x, y : T // E \in \mathbb{U}_j$	Ax
$H \vdash s \in T$	Ax
$H \vdash t \in T$	Ax
$H \vdash E[s, t/x, y]$	Ax

$H, v : s = t \in x, y : T // E, H' \vdash C$	ext u
$\text{quotient_equalityElimination } i \ j \ v'$	
$H, v : s = t \in x, y : T // E, \llbracket v' \rrbracket : E[s, t/x, y], H' \vdash C$	ext u
$H, v : s = t \in x, y : T // E, H' \vdash E[s, t/x, y] \in \mathbb{U}_j$	Ax

$H, z : x, y : T // E, H' \vdash s = t \in S$	Ax
$\text{quotientElimination } i \ j \ x' \ y' \ v$	
$H, z : x, y : T // E, H', x' : T, y' : T \vdash E[x', y'/x, y] \in \mathbb{U}_j$	Ax
$H, z : x, y : T // E, H' \vdash S \in \mathbb{U}_j$	Ax
$H, z : x, y : T // E, H', x' : T, y' : T, v : E[x', y'/x, y] \vdash s[x'/z] = t[y'/z] \in S[x'/z]$	Ax

$H, z : x, y : T // E, H' \vdash s = t \in S$	Ax
$\text{quotientElimination}_2 \ i \ j \ x' \ y' \ v$	
$H, z : x, y : T // E, H', x' : T, y' : T \vdash E[x', y'/x, y] \in \mathbb{U}_j$	Ax
$H, z : x, y : T // E, H' \vdash S \in \mathbb{U}_j$	Ax
$H, z : x, y : T // E, x' : T, y' : T, v : E[x', y'/x, y], H'[x'/z] \vdash s[x'/z] = t[y'/z] \in S[x'/z]$	Ax

WICHTIGE BENUTZERDEFINIERTER QUOTIENTENTYPEN

• Rationale Zahlen

$$x_1 =_q x_2 \equiv \text{match } x_1 \text{ with } \langle z_1, n_1 \rangle \mapsto \text{match } x_2 \text{ with } \langle z_2, n_2 \rangle \mapsto z_1 * n_2 = z_2 * n_1$$

$$\mathbb{Q} \equiv x, y : \mathbb{Z} \times \mathbb{N}^+ // x =_q y$$

$$x_1 + x_2 \equiv \text{match } x_1 \text{ with } \langle z_1, n_1 \rangle \mapsto \text{match } x_2 \text{ with } \langle z_2, n_2 \rangle \mapsto \langle z_1 * n_2 + z_2 * n_1, n_1 * n_2 \rangle$$

$$x_1 - x_2 \equiv \text{match } x_1 \text{ with } \langle z_1, n_1 \rangle \mapsto \text{match } x_2 \text{ with } \langle z_2, n_2 \rangle \mapsto \langle z_1 * n_2 - z_2 * n_1, n_1 * n_2 \rangle$$

$$x_1 * x_2 \equiv \text{match } x_1 \text{ with } \langle z_1, n_1 \rangle \mapsto \text{match } x_2 \text{ with } \langle z_2, n_2 \rangle \mapsto \langle z_1 * z_2, n_1 * n_2 \rangle$$

$$x_1 <_q x_2 \equiv \text{match } x_1 \text{ with } \langle z_1, n_1 \rangle \mapsto \text{match } x_2 \text{ with } \langle z_2, n_2 \rangle \mapsto z_1 * n_2 < z_2 * n_1$$

• Restklassenräume

$$x_1 = x_2 \bmod k \equiv k \text{ divides } x_1 - x_2$$

$$\mathbb{Z} \bmod k \equiv x, y : \mathbb{Z} // x = y \bmod k$$

Operationen auf $\mathbb{Z} \bmod k$ können direkt von \mathbb{Z} übernommen werden

BEHANDLUNG ÜBERSTRUKTURIERTER PRÄDIKATE

● Manche Prädikate auf \mathbb{Q} enthalten zu viel Struktur

– $x_1 <_q x_2$ muß im Quotiententyp (!) wohlgeformt sein

– Gilt $x_1 <_q x_2 = x'_1 <_q x'_2$, wenn $x_1 = x'_1 \in \mathbb{Q}$ und $x_2 = x'_2 \in \mathbb{Q}$?

$z_1 * n_2 < z_2 * n_1 = z'_1 * n'_2 < z'_2 * n'_1$ verlangt $z_1 * n_2 = z'_1 * n'_2 \in \mathbb{Z}$
und $z_2 * n_1 = z'_2 * n'_1 \in \mathbb{Z}$

Nicht gültig für $x_1 = \langle 2, 1 \rangle$, $x'_1 = \langle 4, 2 \rangle$, $x_2 = x'_2 = \langle 3, 1 \rangle$

– Unabhängigkeit vom Repräsentanten nicht mehr gegeben

– Definition von $<_q$ enthält zu viel Struktur, wo nur “Wahrheit” nötig ist

● Type-Squashing erforderlich (neue Definition von $<_q$)

$x_1 <_q x_2 \equiv \downarrow \text{match } x_1 \text{ with } \langle z_1, n_1 \rangle \mapsto \text{match } x_2 \text{ with } \langle z_2, n_2 \rangle \mapsto z_1 * n_2 < z_2 * n_1$

● Alternative: $=_q, <_q$ als geliftete boolesche Operationen

$i_1 =_{\mathbb{Z}} i_2 \equiv \text{if } i_1 = i_2 \text{ then } \mathbf{T} \text{ else } \mathbf{F}$

$i_1 <_{\mathbb{Z}} i_2 \equiv \text{if } i_1 < i_2 \text{ then } \mathbf{T} \text{ else } \mathbf{F}$

$x_1 =_q x_2 \equiv \text{match } x_1 \text{ with } \langle z_1, n_1 \rangle \mapsto \text{match } x_2 \text{ with } \langle z_2, n_2 \rangle \mapsto \uparrow (z_1 * n_2 =_{\mathbb{Z}} z_2 * n_1)$

$x_1 <_q x_2 \equiv \text{match } x_1 \text{ with } \langle z_1, n_1 \rangle \mapsto \text{match } x_2 \text{ with } \langle z_2, n_2 \rangle \mapsto \uparrow (z_1 * n_2 <_{\mathbb{Z}} z_2 * n_1)$

DEFINITION REELLER ZAHLEN MIT QUOTIENTENTYPEN

$$x_1 \leq_q x_2 \equiv x_1 < x_2 \vee x_1 = x_2 \in \mathbb{Q}$$

$$z/n \equiv \langle z, n \rangle$$

$$|x| \equiv \text{match } x \text{ with } \langle z, n \rangle \mapsto \text{if } z < 0 \text{ then } \langle -z, n \rangle \text{ else } \langle z, n \rangle$$

$$\mathbb{R}_{pre} \equiv \{f : \mathbb{N}^+ \rightarrow \mathbb{Q} \mid \forall m, n : \mathbb{N}^+. |f(n) - f(m)| \leq 1/m + 1/n\}$$

$$x_1 =_r x_2 \equiv \forall n : \mathbb{N}^+. |x_1(n) - x_2(n)| \leq 2/n$$

$$\mathbb{R} \equiv x, y : \mathbb{R}_{pre} // x =_r y$$

$$x_1 + x_2 \equiv \lambda n. x_1(n) + x_2(n)$$

$$x_1 - x_2 \equiv \lambda n. x_1(n) - x_2(n)$$

$$|x| \equiv \lambda n. |x(n)|$$

Elegante Beweise erfordern viele Spezialtaktiken

ATOMARE BEZEICHNER

- **Verwaltung eindeutiger Namen**

- **Token**: Textketten, die (anders als Strings) keine Struktur haben
- Einzig mögliche Analyse ist Test auf Gleichheit von Token
- Wichtig für Programmiersprachen und Sicherheitsprotokolle

- **Ausdrücke**

Kanonisch: **Atom**

"token"

token Text-Token

Nichtkanonisch: **if $a=b$ then s else t**

a, b, s, t Terme

- **Reduktion von Ausdrücken**

$\text{if } a=b \text{ then } s \text{ else } t \xrightarrow{\beta} s, \text{ falls } a = b, \text{ sonst } t$

- **Urteile für Typ- und Elementgleichheit**

Atom = Atom

"token" = "token" \in Atom — für alle token! —

Atom = Atom $\in \mathbb{U}_j$ — für alle j ! —

REGELN FÜR ATOMARE BEZEICHNER

$H \vdash \text{Atom} = \text{Atom} \in \mathbb{U}_j$ [Ax]
`atomEquality`

$H \vdash \mathbb{U}_j$ [ext Atom]
`atomFormation`

$H \vdash \text{"token"} = \text{"token"} \in \text{Atom}$ [Ax]
`tokenEquality`

$H \vdash \text{Atom}$ [ext "token"]
`tokenFormation "token"`

$H \vdash \text{if } u_1=v_1 \text{ then } s_1 \text{ else } t_1 = \text{if } u_2=v_2 \text{ then } s_2 \text{ else } t_2 \in T$ [Ax]
`atom_eqEquality v`
 $H \vdash u_1 = u_2 \in \text{Atom}$ [Ax]
 $H \vdash v_1 = v_2 \in \text{Atom}$ [Ax]
 $H, v: u_1=v_1 \in \text{Atom} \vdash s_1 = s_2 \in T$ [Ax]
 $H, v: \neg(u_1=v_1 \in \text{Atom}) \vdash t_1 = t_2 \in T$ [Ax]

$H \vdash \text{if } u=v \text{ then } s \text{ else } t = t_2 \in T$ [Ax]
`atom_eqReduceTrue`
 $H \vdash s = t_2 \in T$ [Ax]
 $H \vdash u = v \in \text{Atom}$ [Ax]

$H \vdash \text{if } u=v \text{ then } s \text{ else } t = t_2 \in T$ [Ax]
`atom_eqReduceFalse`
 $H \vdash t = t_2 \in T$ [Ax]
 $H \vdash \neg(u = v \in \text{Atom})$ [Ax]

- **Abhängiger Durchschnitt** $x : S \cap T[x]$ Bisher nur in MetaPRL
 - Element s muß zu S und gleichzeitig zu $T[s]$ gehören (Selbstreferenz!)
 - Gut für Formalisierung von Abstrakten Datentypen, Record-Strukturen mit internen Abhängigkeiten, Objekten
- **Squiggle-Equality** $s \sim t$
 - Einfachere, syntaktischer Gleichheitstyp, ohne Abhängigkeit vom Typ
 - $s \sim t$ gilt, wenn s und t zum gleichen Term reduzierbar sind oder in \mathbb{Z} oder Atom semantisch gleich sind
 - Substitutionregel gilt auch für Terme die squiggle-gleich sind
- **Stark abhängige Funktionen** $\{f \mid x : S \rightarrow T[f, x]\}$
 - Selbstreferenz: Bildbereich hängt ab von Eingabe und Funktion f selbst
 - Mächtiger als abhängiger Durchschnitt, aber Beweise werden aufwendig
- **Aktuell in Entwicklung**
 - **Logic of Events**: Schließen über Kommunikation und verteilte Prozesse
 - **Reflektion**: Schließen über Beweisverfahren und das Meta-Level der CTT

Inferenzkalkül für Mathematik & Programmierung

● **Ausdrucksstarkes Inferenzsystem**

- Vereinheitlicht und erweitert Logik, λ -Kalkül und einfache Typentheorie
- Formalisierung “natürlicher” Gesetze der zentralen Konzepte
- **Direkte Darstellung** anstatt Simulation
- **Umfangreiche Theorie** bestehend aus
 - Prädikatenlogik (höherer Stufe)
 - Mathematischen Grundkonzepten
 - Grundkonstrukten der Programmierung, einschließlich Rekursion

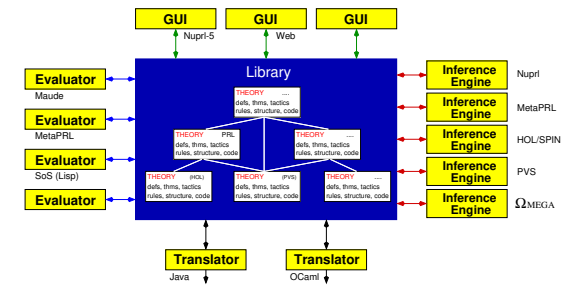
● **Praktische Probleme**

- Beweise erfordern **viel Schreibearbeit** → *Interaktive Beweissysteme*
- Beweise sind **unübersichtlich** (viele Regelanwendungen)
- Beweise sind **schwer zu finden** (viele Regeln und Parameter)
→ *Automatisierung der Beweisführung*

Konstruiere semiautomatische Beweissysteme

● Aufbau von Beweissystemen

- Implementierung interaktiver Beweisassistenten
- Das **NuPRL Logical Programming Environment**



● Automatisierung des formalen Schließens

- Taktisches Beweisen
- Entscheidungsprozeduren
- Integration externer Systeme in den Inferenzmechanismus

● Anwendungen & Demonstrationen

- Formalisierung mathematischen Wissens
- Synthese effizienter Algorithmen aus formalen Spezifikationen

⋮