Definitorische Konstruktion induktiver Datentypen in Isabelle/HOL

Stefan Berghofer

TU München

Motivation und Hintergrund

Mögliche Anwendungsgebiete von Datentypen:

- Modellierung von Datenstrukturen (Listen, Bäume, ...)
- Darstellung der Syntax von Programmiersprachen (z.B. Bali)
- ...

Verschiedene Möglichkeiten der Einführung von Datentypen:

- Axiomatisch (unsicher!)
- Definitorisch (bevorzugte Methode in HOL, Modellerhaltung)

Zu betrachtende Fragestellungen:

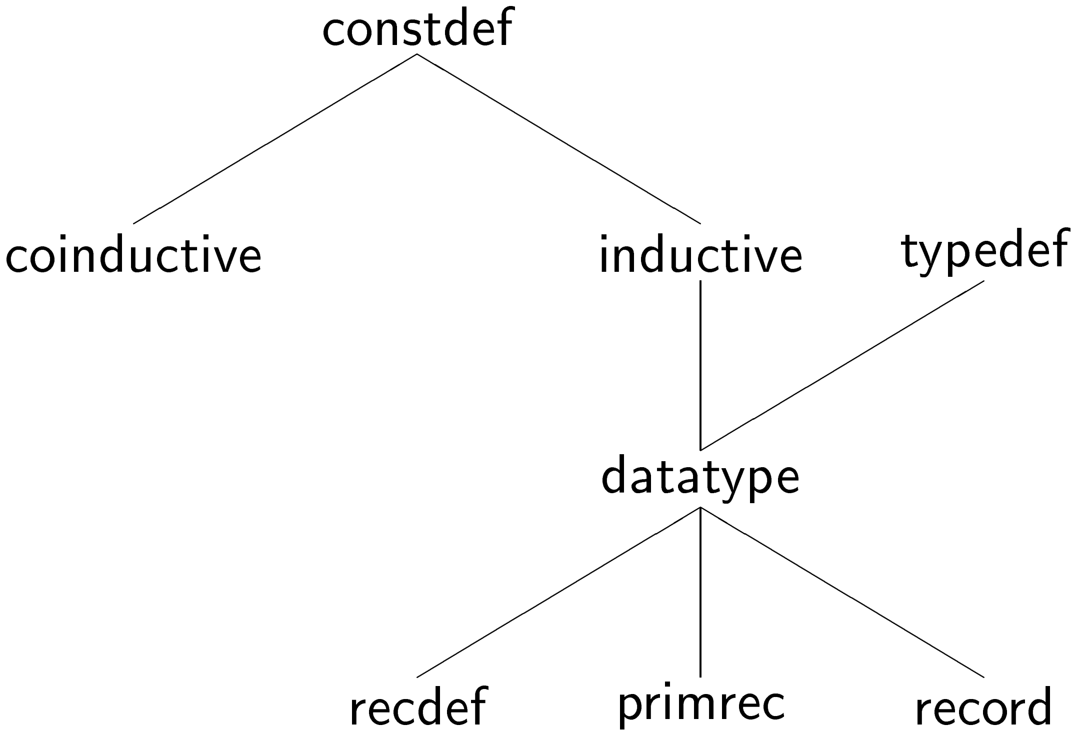
- Konstruierbare Klassen von Datentypen
- Charakteristische Eigenschaften von Datentypen
- Konstruktionsverfahren
- Erweiterungsmöglichkeiten

Beispiele

$$
\begin{array}{llcl}
\textsf{datatype} & \alpha\ \textsf{aexp} & = & \textsf{If\_then\_else}\ (\alpha\ \textsf{bexp})\ (\alpha\ \textsf{aexp})\ (\alpha\ \textsf{aexp}) \\
 & & \mid & \textsf{Sum}\ (\alpha\ \textsf{aexp})\ (\alpha\ \textsf{aexp}) \\
 & & \mid & \textsf{Diff}\ (\alpha\ \textsf{aexp})\ (\alpha\ \textsf{aexp}) \\
 & & \mid & \textsf{Var}\ \alpha \\
 & & \mid & \textsf{Num nat} \\
\textsf{and} & \alpha\ \textsf{bexp} & = & \textsf{Less}\ (\alpha\ \textsf{aexp})\ (\alpha\ \textsf{aexp}) \\
 & & \mid & \textsf{And}\ (\alpha\ \textsf{bexp})\ (\alpha\ \textsf{bexp}) \\
 & & \mid & \textsf{Or}\ (\alpha\ \textsf{bexp})\ (\alpha\ \textsf{bexp})
\end{array}
$$

$$
\begin{array}{llcl}
\textsf{datatype} & (\alpha, \beta)\textsf{term} & = & \textsf{Var}\ \alpha \\
 & & \mid & \textsf{App}\ \beta\ (((\alpha, \beta)\textsf{term})\textsf{list})
\end{array}
$$

Mechanismen zur definitorischen Theorieerweiterung



Charakteristische Eigenschaften von Datentypen

$$\begin{array}{lcl} \text{datatype } \alpha \text{ list} & = & \text{Nil} \\ & | & \text{Cons } \alpha \ (\alpha \text{ list}) \end{array}$$

- Konstruktoren:

$$\begin{array}{rcl} \text{Nil} & :: & \alpha \text{ list} \\ \text{Cons} & :: & \alpha \to \alpha \text{ list} \to \alpha \text{ list} \end{array}$$

  - Verschiedenheit: Nil $\neq$ Cons $x\ xs$
  - Injektivität: Cons $x\ xs$ = Cons $y\ ys \Longleftrightarrow x = y \wedge xs = ys$

- Strukturelle Induktion:

$$\frac{P \text{ Nil} \quad \forall x\ xs.\ P\ xs \Longrightarrow P\ (\text{Cons } x\ xs)}{P\ ys}$$

Weitere abgeleitete Eigenschaften

- Fallunterscheidung

$$\frac{ys = \mathsf{Nil} \Longrightarrow Q \quad \forall x\ xs.\ ys = \mathsf{Cons}\ x\ xs \Longrightarrow Q}{Q}$$

- Primitive Rekursion

$$\begin{array}{lcl} \mathsf{list\_rec}\ f_1\ f_2\ \mathsf{Nil} & = & f_1 \\ \mathsf{list\_rec}\ f_1\ f_2\ (\mathsf{Cons}\ x\ xs) & = & f_2\ x\ xs\ (\mathsf{list\_rec}\ f_1\ f_2\ xs) \end{array}$$

- case - Kombinator

$$\begin{array}{lcl} \mathsf{list\_case}\ f_1\ f_2\ \mathsf{Nil} & = & f_1 \\ \mathsf{list\_case}\ f_1\ f_2\ (\mathsf{Cons}\ x\ xs) & = & f_2\ x\ xs \end{array}$$

- size - Funktion

$$\begin{array}{lcl} \mathsf{size}\ \mathsf{Nil} & = & 0 \\ \mathsf{size}\ (\mathsf{Cons}\ x\ xs) & = & 1 + \mathsf{size}\ xs \end{array}$$

- ...

Konstruierbare Klassen von Datentypen

Allgemeine Gestalt von Datentypdefinitionen:

$$
\begin{array}{llcl}
\textsf{datatype} & (\alpha_1, \ldots, \alpha_h) t_1 & = & C_1^1 \; \tau_{1,1}^1 \; \ldots \; \tau_{1,m_1^1}^1 \;\mid\; \ldots \;\mid\; C_{k_1}^1 \; \tau_{k_1,1}^1 \; \ldots \; \tau_{k_1,m_{k_1}^1}^1 \\
 & & \vdots & \\
\textsf{and} & (\alpha_1, \ldots, \alpha_h) t_n & = & C_1^n \; \tau_{1,1}^n \; \ldots \; \tau_{1,m_1^n}^n \;\mid\; \ldots \;\mid\; C_{k_n}^n \; \tau_{k_n,1}^n \; \ldots \; \tau_{k_n,m_{k_n}^n}^n
\end{array}
$$

Die Typen $\tau_{i,i'}^j$ müssen *zulässig* sein.

Ein Typ τ heißt zulässig, falls gilt

- τ ist nicht rekursiv, d.h. τ enthält keinen der neu definierten Typkonstruktoren $t_1, \ldots, t_n$, oder
- $\tau = (\alpha_1, \ldots, \alpha_h) t_{j'}$, wobei $1 \leq j' \leq n$, oder
- $\tau = (\tau_1', \ldots, \tau_{h'}') t'$, wobei t' der Typkonstruktor eines existierenden Datentyps ist und $\tau_1', \ldots, \tau_{h'}'$ zulässige Typen sind („verschachtelte Rekursion“)

Grenzen der Konstruierbarkeit

datatype t $=$ C (t $\to$ bool) $\mid$ D

Theorem (Cantor): Es gibt keine injektive Funktion mit dem Typ

C $::$ (t $\to$ bool) $\to$ t

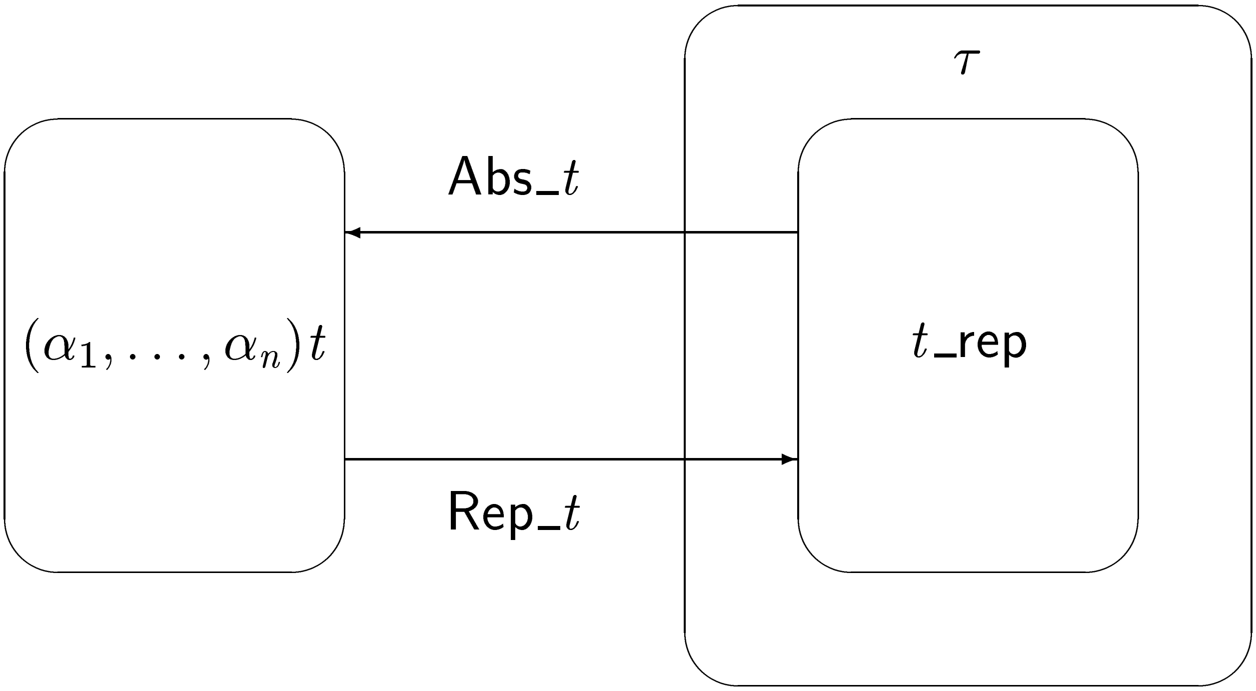
Aber:

- Konstruktion möglich, falls rekursiver Typ nur *strikt positiv*, d.h. rechts von $\to$ vorkommt:

$$\begin{array}{lcl} \text{datatype} \quad (\alpha, \beta, \gamma)\text{tree} & = & \text{Atom } \alpha \\ & \mid & \text{Branch } \beta \; (\gamma \to (\alpha, \beta, \gamma)\text{tree}) \end{array}$$

- Falls $\to$ nicht (wie in HOL) den *vollen Funktionenraum*, sondern den *stetigen Funktionenraum* („Scott Domains“) bezeichnet, gilt die obige Einschränkung nicht mehr.

Definition neuer Typen in Isabelle/HOL



- Teilmenge t_rep des existierenden Typs τ ist repräsentierenden Menge des neuen Typs $(\alpha_1, \ldots, \alpha_n)t$
- Isomorphie zwischen t_rep und $(\alpha_1, \ldots, \alpha_n)t$

$$\begin{array}{l} \mathsf{Abs\_}t\ (\mathsf{Rep\_}t\ x) = x \\ y \in t\mathsf{\_rep} \Longrightarrow \mathsf{Rep\_}t\ (\mathsf{Abs\_}t\ y) = y \\ \mathsf{Rep\_}t\ x \in t\mathsf{\_rep} \end{array}$$

- Eigenschaften von $(\alpha_1, \ldots, \alpha_n)t$ ergeben sich aus Eigenschaften von t_rep

Geeignete Wahl von τ?

Konstruktion von t_rep?

Ein Universum für rekursive Typen

Idee: Datentypen können als Bäume (Typ α dtree) aufgefaßt werden

Verwendete Konstruktionselemente:

- Einbettung: für nichtrekursive Vorkommen

  $$\mathsf{Leaf} \ :: \ \alpha \to \alpha \ \mathsf{dtree}$$

- Summe: Modellierung verschiedener Konstruktoren

  $$\mathsf{In0},\ \mathsf{In1} \ :: \ \alpha \ \mathsf{dtree} \to \alpha \ \mathsf{dtree}$$

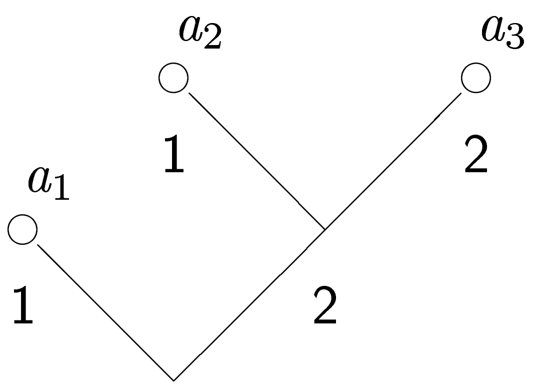
- Produkt: Darstellung von Konstruktoren mit mehreren Argumenten

  $$\mathsf{Pair} \ :: \ \alpha \ \mathsf{dtree} \to \alpha \ \mathsf{dtree} \to \alpha \ \mathsf{dtree}$$

Wichtige Eigenschaft von α dtree:
Abgeschlossenheit unter Summen- und Produktbildung.

Modellierung von Bäumen in HOL

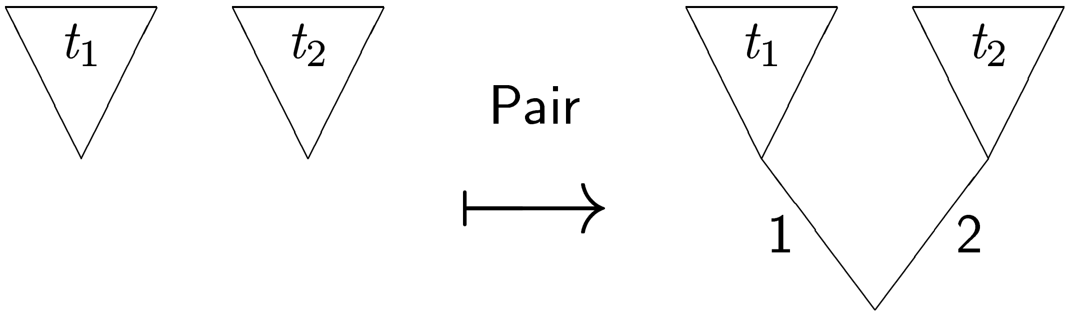
- Baum: Menge von Knoten
- Knoten über „Pfad“ erreichbar
- Pfad: Funktion vom Typ nat $\rightarrow$ nat



$$T = \{(f_1, a_1), (f_2, a_2), (f_3, a_3)\}$$

wobei

$$\begin{aligned} f_1 &= \langle 1, 0, 0, \ldots \rangle \\ f_2 &= \langle 2, 1, 0, 0, \ldots \rangle \\ f_3 &= \langle 2, 2, 0, 0, \ldots \rangle \end{aligned}$$



Modellierung von Bäumen in HOL

$$
\begin{array}{lcl}
\alpha\ \mathsf{node} & = & (\mathsf{nat} \to \mathsf{nat}) \times (\alpha + \mathsf{nat}) \\
\alpha\ \mathsf{dtree} & = & (\alpha\ \mathsf{node})\ \mathsf{set} \\[1ex]
\mathsf{push} & :: & \mathsf{nat} \to \alpha\ \mathsf{node} \to \alpha\ \mathsf{node} \\
\mathsf{push}\ k\ (f, a) & = & (\lambda i.\ \mathsf{case}\ i\ \mathsf{of}\ 0 \Rightarrow k \mid \mathsf{Suc}\ j \Rightarrow f\ j,\ a) \\[1ex]
\mathsf{Pair} & :: & \alpha\ \mathsf{dtree} \to \alpha\ \mathsf{dtree} \to \alpha\ \mathsf{dtree} \\
\mathsf{Pair}\ t_1\ t_2 & = & (\mathsf{push}\ 1\ \text{``}\ t_1) \cup (\mathsf{push}\ 2\ \text{``}\ t_2) \\[1ex]
\mathsf{Leaf}\ a & :: & \alpha \to \alpha\ \mathsf{dtree} \\
\mathsf{Leaf}\ a & = & \{(\lambda x.\ 0, \mathsf{Inl}\ a)\} \\[1ex]
\mathsf{Numb} & :: & \mathsf{nat} \to \alpha\ \mathsf{dtree} \\
\mathsf{Numb}\ k & = & \{(\lambda x.\ 0, \mathsf{Inr}\ k)\} \\[1ex]
\mathsf{In0} & :: & \alpha\ \mathsf{dtree} \to \alpha\ \mathsf{dtree} \\
\mathsf{In0}\ t & = & \mathsf{Pair}\ (\mathsf{Numb}\ 0)\ t \\[1ex]
\mathsf{In1} & :: & \alpha\ \mathsf{dtree} \to \alpha\ \mathsf{dtree} \\
\mathsf{In1}\ t & = & \mathsf{Pair}\ (\mathsf{Numb}\ 1)\ t
\end{array}
$$

Konstruktion der repräsentierenden Menge

Induktive Definition von list_rep :: (α dtree) set

$$\begin{array}{lcl} \text{Nil\_rep} & \equiv & \text{In0 dummy} \\ \text{Cons\_rep}\ y\ ys & \equiv & \text{In1 (Pair (Leaf}\ y)\ ys) \end{array}$$

$$\frac{}{\text{Nil\_rep} \in \text{list\_rep}} \qquad \frac{ys \in \text{list\_rep}}{\text{Cons\_rep}\ y\ \ ys \in \text{list\_rep}}$$

Definition der Konstruktoren:

$$\begin{array}{rcl} \text{Nil} & \equiv & \text{Abs\_list Nil\_rep} \\ \text{Cons}\ x\ xs & \equiv & \text{Abs\_list (Cons\_rep}\ x\ (\text{Rep\_list}\ xs)) \end{array}$$

wobei

$$\begin{array}{lcl} \text{Abs\_list} & :: & \alpha\ \text{dtree} \to \alpha\ \text{list} \\ \text{Rep\_list} & :: & \alpha\ \text{list} \to \alpha\ \text{dtree} \end{array}$$

Fixpunkte und induktive Definitionen

$$\mathsf{lfp}\ F = \bigcap\{x \mid F\ x \subseteq x\}$$

Theorem (Knaster-Tarski): Sei F monoton. Dann gilt:

$$\mathsf{lfp}\ F = F\ (\mathsf{lfp}\ F) \qquad\qquad \frac{F\ y \subseteq y}{\mathsf{lfp}\ F \subseteq y}$$

$$\frac{F\ \{x \mid P\ x\} \subseteq \{x \mid P\ x\}}{\mathsf{lfp}\ F \subseteq \{x \mid P\ x\}}\ (\mathsf{weakInd}) \qquad \frac{F\ (\mathsf{lfp}\ F \cap \{x \mid P\ x\}) \subseteq \{x \mid P\ x\}}{\mathsf{lfp}\ F \subseteq \{x \mid P\ x\}}\ (\mathsf{strongInd})$$

Darstellung von list_rep als kleinster Fixpunkt:

$$\begin{aligned} F\ L &= \{x \mid x = \mathsf{Nil\_rep}\ \vee\ \exists y\ ys.\ x = \mathsf{Cons\_rep}\ y\ ys \wedge ys \in L\} \\ \mathsf{list\_rep} &= \mathsf{lfp}\ F \end{aligned}$$

$$\begin{aligned} \mathsf{datatype}\ \alpha\ \mathsf{list} &= \mathsf{Nil} \\ &\mid\ \mathsf{Cons}\ \alpha\ (\alpha\ \mathsf{list}) \end{aligned} \quad \rightsquigarrow \quad \mathsf{list\_rep} = F\ \mathsf{list\_rep}$$

Strukturelle Induktion

Konkretes Induktionstheorem für list_rep (aus strongInd):

$$\frac{P\ \mathsf{Nil\_rep} \quad \forall y\ ys.\ P\ ys \wedge ys \in \mathsf{list\_rep} \Longrightarrow P\ (\mathsf{Cons\_rep}\ y\ ys)}{ys' \in \mathsf{list\_rep} \Longrightarrow P\ ys'}$$

Abstraktes Induktionstheorem:

$$\frac{Q\ \mathsf{Nil} \quad \forall x\ xs.\ Q\ xs \Longrightarrow Q\ (\mathsf{Cons}\ x\ xs)}{Q\ xs'}$$

Herleitung aus konkretem Induktionstheorem:

$$\cfrac{\cfrac{\ldots \quad \cfrac{\cfrac{\ldots \Longrightarrow Q\ (\mathsf{Cons}\ y\ (\mathsf{Abs\_list}\ ys))}{\ldots \Longrightarrow Q\ (\mathsf{Abs\_list}\ (\mathsf{Cons\_rep}\ y\ (\mathsf{Rep\_list}\ (\mathsf{Abs\_list}\ ys))))}}{\forall y\ ys.\ Q\ (\mathsf{Abs\_list}\ ys) \wedge ys \in \mathsf{list\_rep} \Longrightarrow Q\ (\mathsf{Abs\_list}\ (\mathsf{Cons\_rep}\ y\ ys))}}{\mathsf{Rep\_list}\ xs' \in \mathsf{list\_rep} \Longrightarrow Q\ (\mathsf{Abs\_list}\ (\mathsf{Rep\_list}\ xs'))}}{Q\ xs'}$$

Hierbei: $P = Q \circ \mathsf{Abs\_list}$, $ys' = \mathsf{Rep\_list}\ xs'$

Primitive Rekursion

Charakteristische Gleichungen für list_rec:

$$\begin{array}{lcl} \text{list\_rec}\ f_1\ f_2\ \text{Nil} & = & f_1 \\ \text{list\_rec}\ f_1\ f_2\ (\text{Cons}\ x\ xs) & = & f_2\ x\ xs\ (\text{list\_rec}\ f_1\ f_2\ xs) \end{array}$$

Induktive Definition des Funktionsgraphen list_rec_rel:

$$\frac{}{(\text{Nil},\ f_1) \in \text{list\_rec\_rel}\ f_1\ f_2} \qquad \frac{(xs,\ y) \in \text{list\_rec\_rel}\ f_1\ f_2}{(\text{Cons}\ x\ xs,\ f_2\ x\ xs\ y) \in \text{list\_rec\_rel}\ f_1\ f_2}$$

Definition von list_rec mit Hilfe von list_rec_rel und Auswahloperator ε:

$$\text{list\_rec}\ f_1\ f_2\ xs \ \equiv\ \varepsilon y.\ (xs,\ y) \in \text{list\_rec\_rel}\ f_1\ f_2$$

Zu zeigen: list_rec_rel ist $linkstotal$ und $rechtseindeutig$, d.h.

$$\exists_1 y.\ (xs,\ y) \in \text{list\_rec\_rel}\ f_1\ f_2$$

Konstruktion verschachtelt rekursiver Typen

„Auffalten" der Datentypdefinition:

$$\begin{array}{lllll}
\text{datatype} & (\alpha, \beta)\text{term} & = & \text{Var } \alpha \\
& & \mid & \text{App } \beta \ ((\alpha, \beta)\text{term_list}) \\
\text{and} & (\alpha, \beta)\text{term_list} & = & \text{Nil'} \\
& & \mid & \text{Cons' } ((\alpha, \beta)\text{term}) \ ((\alpha, \beta)\text{term_list})
\end{array}$$

Vorsicht: Typ (α, β)term_list und Konstruktoren Nil' und Cons' nicht wirklich einführen!

Konstruktion der repräsentierenden Menge:

$$\frac{}{\text{In0 (Leaf (Inl } a)) \in \text{term\_rep}} \qquad \frac{xs \in \text{term\_list\_rep}}{\text{In1 (Pair (Leaf (Inr } b)) \ xs) \in \text{term\_rep}}$$

$$\frac{}{\text{In0 dummy} \in \text{term\_list\_rep}} \qquad \frac{x \in \text{term\_rep} \quad xs \in \text{term\_list\_rep}}{\text{In1 (Pair } x \ xs) \in \text{term\_list\_rep}}$$

Konstruktion verschachtelt rekursiver Typen

Bijektion zwischen Elementen von term_list_rep und $((\alpha, \beta)$term)list:

$$\begin{array}{lcl} \text{Rep\_term\_list Nil} & = & \text{In0 dummy} \\ \text{Rep\_term\_list (Cons } x\ xs) & = & \text{In1 (Pair (Rep\_term } x\text{) (Rep\_term\_list } xs)) \end{array}$$

Damit:

$$\text{Abs\_term\_list (Rep\_term\_list } xs) = xs$$
$$ys \in \text{term\_list\_rep} \Longrightarrow \text{Rep\_term\_list (Abs\_term\_list } ys) = ys$$
$$\text{Rep\_term\_list } xs \in \text{term\_list\_rep}$$

wobei

$$\text{Abs\_term\_list} \equiv \text{inv Rep\_term\_list}$$

Definition der Konstruktoren:

$$\begin{array}{rcl} \text{Var } a & \equiv & \text{Abs\_term (In0 (Leaf (Inl } a))) \\ \text{App } b\ ts & \equiv & \text{Abs\_term (In1 (Pair (Leaf (Inr } b)) \text{ (Rep\_term\_list } ts))) \end{array}$$

Theoreme für verschachtelt rekursive Typen

- Strukturelle Induktion

$$\frac{\begin{array}{lcl} \forall a. & & P_1\ (\mathsf{Var}\ a) \\ \forall b\ ts.\ P_2\ ts & \Longrightarrow & P_1\ (\mathsf{App}\ b\ ts) \\ & & P_2\ \mathsf{Nil} \\ \forall t\ ts.\ P_1\ t \wedge P_2\ ts & \Longrightarrow & P_2\ (\mathsf{Cons}\ t\ ts) \end{array}}{P_1\ t \wedge P_2\ ts}$$

- Primitive Rekursion

$$\begin{array}{lcl} \mathsf{term\_rec}\ f_1\ \ldots\ f_4\ (\mathsf{Var}\ a) & = & f_1\ a \\ \mathsf{term\_rec}\ f_1\ \ldots\ f_4\ (\mathsf{App}\ b\ ts) & = & f_2\ b\ ts\ (\mathsf{term\_list\_rec}\ f_1\ \ldots\ f_4\ ts) \\ \mathsf{term\_list\_rec}\ f_1\ \ldots\ f_4\ \mathsf{Nil} & = & f_3 \\ \mathsf{term\_list\_rec}\ f_1\ \ldots\ f_4\ (\mathsf{Cons}\ t\ ts) & = & f_4\ t\ ts\ (\mathsf{term\_rec}\ f_1\ \ldots\ f_4\ t) \\ & & \qquad (\mathsf{term\_list\_rec}\ f_1\ \ldots\ f_4\ ts) \end{array}$$

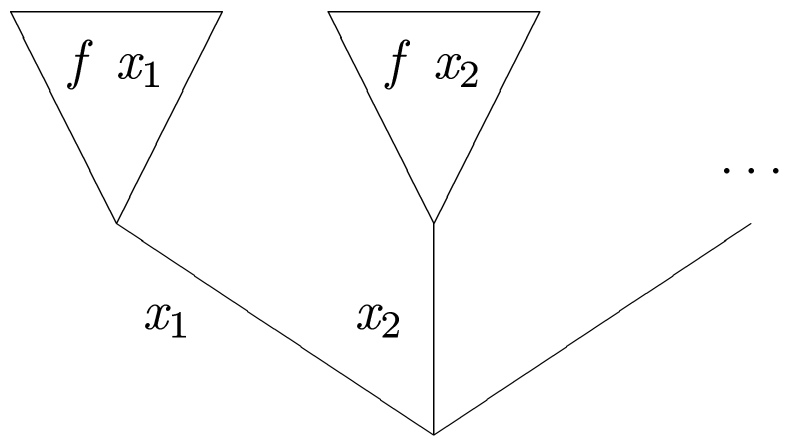
Verschachtelte Rekursion mit Funktionstypen

Erweiterung des Universums:

$$
\begin{aligned}
(\alpha, \beta)\mathsf{node} &= (\mathsf{nat} \to (\beta + \mathsf{nat})) \times (\alpha + \mathsf{nat}) \\
(\alpha, \beta)\mathsf{dtree} &= ((\alpha, \beta)\mathsf{node})\mathsf{set}
\end{aligned}
$$

β: Verzweigungstyp

$$
\begin{aligned}
\mathsf{Lim} \quad &:: \quad (\beta \to (\alpha, \beta)\mathsf{dtree}) \to (\alpha, \beta)\mathsf{dtree} \\
\mathsf{Lim}\ f \quad &\equiv \quad \bigcup\{z \mid \exists x.\ z = \mathsf{push}\ (\mathsf{Inl}\ x)\ \text{“}\ (f\ \ x)\}
\end{aligned}
$$



Verschachtelte Rekursion mit Funktionstypen

Repräsentierende Menge von (α, β, γ)tree:

$$\text{Funns } S \equiv \{g \mid \text{range } g \subseteq S\}$$

$$\frac{}{\text{In0 (Leaf (Inl } a)) \in \text{tree\_rep}} \qquad \frac{g \in \text{Funs tree\_rep}}{\text{In1 (Pair (Leaf (Inr } b)) \text{ (Lim } g)) \in \text{tree\_rep}}$$

Konstruktoren:

$$\begin{array}{lcl} \text{Atom } a & \equiv & \text{Abs\_tree (In0 (Leaf (Inl } a))) \\ \text{Branch } b\ f & \equiv & \text{Abs\_tree (In1 (Pair (Leaf (Inr } b)) \text{ (Lim (Rep\_tree} \circ f)))) \end{array}$$

Theoreme für tree

- Strukturelle Induktion

$$\frac{\forall a.\ P\ (\mathsf{Atom}\ a) \quad \forall b\ f.\ (\forall x.\ P\ (f\ x)) \Longrightarrow P\ (\mathsf{Branch}\ b\ f)}{P\ t}$$

- Primitive Rekursion

$$\begin{array}{lcl} \mathsf{tree\_rec}\ f_1\ f_2\ (\mathsf{Atom}\ a) & = & f_1\ a \\ \mathsf{tree\_rec}\ f_1\ f_2\ (\mathsf{Branch}\ b\ f) & = & f_2\ b\ f\ ((\mathsf{tree\_rec}\ f_1\ f_2) \circ f) \end{array}$$

Vergleich mit anderen Arbeiten

System	verschränkt	verschachtelt	Funktionstypen	definitorisch
Isabelle/HOL	ja	ja	bald :-)	ja
HOL98	ja	ja	nein	ja
HOL Light	ja	ja	nein	ja
PVS	nein	ja	ja	**nein :-(**