Indexed Containers

based on joint work with Neil Ghani, Peter Hancock, Conor McBride, Peter Morris

Thorsten Altenkirch

School of Computer Science and IT
University of Nottingham

April 26, 2007

Inductive types

- Example: λ terms ala de Bruijn: $\mathrm{Lam} \in \textbf{Set}$:

$$\begin{array}{rcl} \mathrm{var} & \in & \mathbb{N} \to \mathrm{Lam} \\ \mathrm{app} & \in & \mathrm{Lam} \to \mathrm{Lam} \to \mathrm{Lam} \\ \mathrm{lam} & \in & \mathrm{Lam} \to \mathrm{Lam} \end{array}$$

- In Epigram syntax:

$$\frac{i \in \mathbb{N}}{\mathrm{var}\ i \in \mathrm{Lam}} \qquad \frac{t, u \in \mathrm{Lam}}{\mathrm{app}\ t\ u \in \mathrm{Lam}} \qquad \frac{t \in \mathrm{Lam}}{\mathrm{lam}\ t \in \mathrm{Lam}}$$

- Natural numbers (ala Peano):

$$\frac{}{0 \in \mathbb{N}} \qquad \frac{n \in \mathbb{N}}{\mathrm{succ}\ n \in \mathbb{N}}$$

- Initial algebra semantics:

$$\begin{array}{rcl} F_{\mathbb{N}}\, X & = & 1 + X \\ F_{\mathrm{Lam}}\, X & = & \mathbb{N} + X \times X + X \end{array}$$

Indexed inductive types

- λ-terms indexed by the number of free variables: $\mathrm{Lam} \in \mathbb{N} \to \mathbf{Set}$

$$\frac{i \in \mathrm{Fin}\, n}{\mathrm{var}\, i \in \mathrm{Lam}\, n} \qquad \frac{t, u \in \mathrm{Lam}\, n}{\mathrm{app}\, t\, u \in \mathrm{Lam}\, n} \qquad \frac{t \in \mathrm{Lam}\, (n+1)}{\mathrm{lam}\, t \in \mathrm{Lam}\, n}$$

- Finite sets with n elements: $\mathrm{Fin} \in \mathbb{N} \to \mathbf{Set}$:

$$\frac{n \in \mathbb{N}}{\mathrm{fzero} \in \mathrm{Fin}\, (n+1)} \qquad \frac{i \in \mathrm{Fin}\, n}{\mathrm{fsucc}\, i \in \mathrm{Fin}\, (n+1)}$$

- Initial algebra sematics:

$$\begin{aligned} F_{\mathrm{Lam}}, F_{\mathrm{Fin}} &\in (\mathbb{N} \to \mathbf{Set}) \to \mathbb{N} \to \mathbf{Set} \\ F_{\mathrm{Lam}}\, X\, n &= \mathrm{Fin}\, n + X\, n \times X\, n + X\, (n+1) \\ F_{\mathrm{Fin}}\, X\, n &= \Sigma m \in \mathbb{N}. m + 1 = n \times (1 + X\, m) \end{aligned}$$

Typed λ-terms

$$\begin{array}{rcl} \mathrm{Ty} & \in & \mathbf{Set} \\ \mathrm{Var}, \mathrm{Lam} & \in & [\mathrm{Ty}] \to \mathrm{Ty} \to \mathbf{Set} \end{array}$$

$$\frac{}{\mathrm{ty} \in \mathrm{Ty}} \qquad \frac{\sigma, \tau \in \mathrm{Ty}}{\mathrm{arr}\,\sigma\,\tau \in \mathrm{Ty}}$$

$$\frac{}{\mathrm{vzero} \in \mathrm{Var}\,(\sigma : \Gamma)\,\sigma} \qquad \frac{x \in \mathrm{Var}\,\Gamma\,\sigma}{\mathrm{vsucc}\,x \in \mathrm{Var}\,(\tau : \Gamma)\,\sigma}$$

$$\frac{x \in \mathrm{Var}\,\Gamma\,\sigma}{\mathrm{var}\,x \in \mathrm{Lam}\,\Gamma\,\sigma} \qquad \frac{t \in \mathrm{Lam}\,\Gamma\,(\mathrm{arr}\,\sigma\,\tau) \quad u \in \mathrm{Lam}\,\Gamma\,\sigma}{\mathrm{app}\,t\,u \in \mathrm{Lam}\,\Gamma\,\tau}$$

$$\frac{t \in \mathrm{Lam}\,(\sigma : \Gamma)\,\tau}{\mathrm{lam}\,t \in \mathrm{Lam}\,\Gamma\,(\mathrm{arr}\,\sigma\,\tau)}$$

Why container?

- Functorial semantics is too general:
  not every funtors has an initial algebra, e.g.

$$F\,X = (X \to \mathrm{Bool}) \to \mathrm{Bool}$$

- Strictly positive functors?
  Too syntactic.
- Theory of Containers:
  semantic counterpart of strictly positive
- Useful for generic programming (e.g. derivatives of datatypes)
- **Now:** extend containers to indexed datatypes.

Unary container

- A unary container $S \lhd P \in \mathrm{Cont}$ is given by:
  $S \in \mathbf{Set}$ a set of shapes
  $P \in S \to \mathbf{Set}$ a family of positions
- Extension of a container as a functor:

$$\frac{S \lhd P \in \mathrm{Cont}}{[\![S \lhd P]\!] \in \mathbf{Set} \to \mathbf{Set}} \qquad [\![S \lhd P]\!]\, X = \Sigma s \in S.P\, s \to X$$

- Example Lists: $[\![\mathbb{N} \lhd \mathrm{Fin}]\!]\, X = [X]$.
- Obs:

$$[\![S \lhd P]\!]\, \{*\} = S$$

Container morphisms

- $f \lhd u \in \mathrm{Cont}\,(S \lhd P)\,(T \lhd Q)$ is given by:

$$\begin{aligned} f &\in S \to T \\ u &\in \Pi s \in S.Q\,s \to P\,s \end{aligned}$$

- Every morphism gives rise to a natural transformation:

$$\frac{f \lhd u \in \mathrm{Cont}\,(S \lhd P)\,(T \lhd Q)}{[\![f \lhd u]\!] \in \Pi X \in \mathbf{Set}.[\![S \lhd P]\!]\,X \to [\![T \lhd Q]\!]\,X}$$

$$[\![f \lhd u]\!]\,X\,(s, p) = (f\,s, p \circ u\,s)$$

- Example:

$$\begin{aligned} \mathrm{rev} &\in \Pi X \in \mathbf{Set}.\mathrm{List}\,X \to \mathrm{List}\,X \\ \mathrm{rev} &= [\![\lambda n.n \lhd \lambda n, i.n - i]\!] \end{aligned}$$

- Theorem (Abbott, A., Ghani): The extension functor $[\![-]\!]$ is full and faithful.

Constructions on containers

Given container $S \lhd P, T \lhd Q$:

Coproduct

$$(S \lhd P) + (T \lhd Q) = S + T \lhd \lambda \begin{array}{lcl} \mathrm{Left}\, s & . & P\, s \\ \mathrm{Right}\, t & . & Q\, t \end{array}$$

Product

$$(S \lhd P) \times (T \lhd Q) = S \times T \lhd \lambda(s, t).P\, s + Q\, t$$

Constant exponentiation

$$K \to (S \lhd P) = K \to S \lhd \lambda f.\Sigma k \in K.P\,(f\, k)$$

n-ary container

- An n-ary container $S \lhd P \in \mathrm{Cont}\, n$ is given by:

$S \in \mathbf{Set}$	a set of shapes
$P \in S \to \mathrm{Fin}\, n \to \mathbf{Set}$	n families of positions

- Extension of a container as a functor:

$$\frac{S \lhd P \in \mathrm{Cont}\, n}{[\![S \lhd P]\!] \in (\mathrm{Fin}\, n \to \mathbf{Set}) \to \mathbf{Set}}$$

$$[\![S \lhd P]\!]\, X = \Sigma s \in S.\Pi i \in \mathrm{Fin}\, n.P\, s\, i \to X\, i$$

- Example: leaf/node labelled trees

Closure under μ

- $$\frac{S \lhd P, Q \in \mathrm{Cont}\,2}{\mu(S \lhd P, Q) \in \mathrm{Cont}\,1}$$
  $\mu(S \lhd P, Q) = \mathrm{W}\,S\,P \lhd \mathrm{Path}\,S\,P\,Q$
- W-types: Given $S \in \textbf{Set}, P \in S \to \textbf{Set}$: $\mathrm{W}\,S\,P \in \textbf{Set}$.
  $$\frac{s \in S \quad f \in P\,s \to \mathrm{W}\,S\,P}{\mathrm{sup}\,s\,f \in \mathrm{W}\,S\,P}$$
- Paths, additionally given $Q \in S \to \textbf{Set}$,
  $\mathrm{Path}\,S\,P\,Q \in \mathrm{W}\,S\,P \to \textbf{Set}$
  $$\frac{q \in Q\,s}{\mathrm{top}\,q \in \mathrm{Path}\,S\,P\,Q\,(\mathrm{sup}\,s\,f)} \qquad \frac{p \in P\,s \quad r \in \mathrm{Path}\,S\,P\,Q\,(f\,p)}{\mathrm{down}\,p\,r \in \mathrm{Path}\,S\,P\,Q\,(sup\,s\,f)}$$
- Lemma (Abbott,A.,Ghani):
  $\mathrm{Path}\,S\,P\,Q$ is definable using W-types.

Strictly Positive Types

$$\frac{n \in \mathbb{N}}{\mathrm{SPT}\, n \in \mathbf{Type}} \qquad \frac{i \in \mathrm{Fin}\, n}{\eta^{\mathrm{T}}\, i \in \mathrm{SPT}\, n} \qquad \frac{}{0^{\mathrm{T}}, 1^{\mathrm{T}} \in \mathrm{SPT}\, n}$$

$$\frac{S, T \in \mathrm{SPT}\, n}{S + T, S \times T \in \mathrm{SPT}\, n} \qquad \frac{S \in \mathrm{SPT}\, n \quad K \in \mathbf{Set}}{K \to S \in \mathrm{SPT}\, n}$$

$$\frac{S \in \mathrm{SPT}\,(n+1)}{\mu S \in \mathrm{SPT}\, n}$$

Indexed container

- Given $I \in \mathbf{Set}$, an I-indexed container $S \lhd P \in \mathrm{Cont}\, I$ is given by:

given by:	$S \in \mathbf{Set}$	a set of shapes
	$P \in S \to I \to \mathbf{Set}$	n families of positions

- Extension of a container as a functor:

$$\frac{S \lhd P \in \mathrm{Cont}\, n}{[\![S \lhd P]\!] \in (I \to \mathbf{Set}) \to \mathbf{Set}}$$

$$[\![S \lhd P]\!]\, X = \Sigma s \in S.\Pi i \in I.P\, s\, i \to X\, i$$

- Examples $\mathrm{Fin}, \mathrm{Lam}, \ldots$

Strictly positive indexed types

$$\frac{I \in \mathbf{Set}}{\mathrm{SPT}\, I \in \mathbf{Type}} \qquad \frac{i \in I}{\eta^{\mathrm{T}}\, i \in \mathrm{SPT}\, I} \qquad \frac{}{0^{\mathrm{T}}, 1^{\mathrm{T}} \in \mathrm{SPT}\, I}$$

$$\frac{f \in I \to J \quad F \in I \to \mathrm{SPT}\, O}{\begin{array}{c}\Pi^{\mathrm{T}}\, f\, F \in J \to \mathrm{SPT}\, O \\ \Sigma^{\mathrm{T}}\, f\, F \in J \to \mathrm{SPT}\, O\end{array}} \qquad \frac{F \in J \to \mathrm{SPT}\,(I + J)}{\mu\, F \in J \to \mathrm{SPT}\, I}$$

$$\begin{array}{rcl}
F_{\mathrm{Fin}}, F_{\mathrm{Lam}} & \in & \mathbb{N} \to \mathrm{SPT}\, \mathbb{N} \\
F_{\mathrm{Fin}} & = & \Sigma^{\mathrm{T}}(\lambda n.1 + n)\,(\lambda n.1 + \eta^{\mathrm{T}} n) \\
\mathrm{Fin}, \mathrm{Lam} & \in & \mathbb{N} \to \mathrm{SPT}\emptyset \\
\mathrm{Fin} & = & \mu F_{\mathrm{Fin}} \\
F_{\mathrm{Lam}}\, n & = & (\mathrm{Fin})^{+\mathbb{N}} + (\eta^{\mathrm{T}}\, n)^2 + \eta^{\mathrm{T}}\,(n + 1) \\
\mathrm{Lam} & = & \mu F_{\mathrm{Lam}}
\end{array}$$

Summary

- We can extend the results for ordinary containers to indexed containers.
- The type of indexed strictly positive types is itself an indexed strictly positive type.
- We still only need W-types!
- Interpret inductive schemes in a very small core theory. Application: Small trusted code base for proof assistants.
- Generic programming for dependent types.