System F_i

a Higher-Order Polymorphic λ-Calculus with Erasable Term-Indices

Ki Yung Ahn Tim Sheard

Portland State University

Marcelo Fiore Andrew Pitts

University of Cambridge

TLCA 2013

26.VI.2013

Motivation

Studying and designing programming languages
with indexed datatypes.

- Type indexing.

Haskell `data Powl a = PCons a (Powl(a,a)) | PNil`

System F_ω $\lambda\, A^*.\, \forall\, X^{*\to *}.\, \big(A \to X\,(A \times A) \to X\,A\big) \to X\,A \to X\,A$

Motivation

Studying and designing programming languages
with indexed datatypes.

- Type indexing.

  Haskell `data Powl a = PCons a (Powl(a,a)) | PNil`

  System F_ω $\lambda\, A^*.\, \forall\, X^{*\to *}.\, \big(A \to X\,(A \times A) \to X\,A\big) \to X\,A \to X\,A$

- Term indexing.

  Language?

```
data Vec :  * -> Nat -> * where
  VCons :  a -> Vec a {i} -> Vec {S i}
  VNil  :  Vec a {Z}
```

  System?

Design Space

Functional Programming Language	Constructive Proving Assistant
ML, Haskell	Coq, Agda
System F	MLTT, ICC
Programming	Proof
Polymorphism	Dependent types
Impredicativity	Universes
Type inference	Type checking
Recursive types	Inductive types
Equality types	Identity types

Type-Theoretic Foundations

- What is a basic type theory for aiding the study and design of light-weight, and logically sound, programming languages with indexed datatypes?
- What induction and/or recursion principles are available?

System F_i

- Minimal extension (avoiding dependent types) of Curry-style System F_ω with type-indexed kinds.
- Index-erasure property:
  Type indices are irrelevant for computation.
- Strongly normalising.
- Logically consistent.
- Supports functional encodings of recursive indexed datatypes.

System F_i
Syntax

Kinds

$$\kappa \quad ::= \quad * \mid \kappa \to \kappa \mid A \to \kappa$$

System $\mathbf{F}_i$
Syntax

Kinds

$$\kappa \quad ::= \quad * \mid \kappa \to \kappa \mid A \to \kappa$$

Type Constructors

$$A, B, F, G \quad ::= \quad X \mid A \to B \mid \lambda X^{\kappa}.\, F \mid F\, G \mid \forall X^{\kappa}.\, B$$
$$\mid \lambda i^{A}.\, F \mid F\{s\} \mid \forall i^{A}.\, B$$

Terms

$$r, s, t \quad ::= \quad x \mid \lambda x.\, t \mid r\ s$$

Typing Contexts

$$\Delta \quad ::= \quad \cdot \mid \Delta, X^{\kappa} \mid \Delta, i^{A}$$
$$\Gamma \quad ::= \quad \cdot \mid \Gamma, x : A$$

System $\mathbf{F}_i$
Typing Rules

$$\frac{\vdash \Delta \qquad \cdot \vdash A : *}{\vdash \Delta, i^A} \; (i \notin \mathsf{dom}(\Delta))$$

$$(\mathsf{R}i) \; \frac{\cdot \vdash A : * \qquad \vdash \kappa : \Box}{\vdash A \to \kappa : \Box}$$

$$(\lambda i)\ \frac{\cdot \vdash A : * \qquad \Delta, i^A \vdash F : \kappa}{\Delta \vdash \lambda i^A.F : A \to \kappa}$$

$$(@i)\ \frac{\Delta \vdash F : A \to \kappa \qquad \Delta; \cdot \vdash s : A}{\Delta \vdash F\{s\} : \kappa}$$

$$(\forall i)\ \frac{\cdot \vdash A : * \qquad \Delta, i^A \vdash B : *}{\Delta \vdash \forall i^A.B : *}$$

$$(\mathsf{Conv})\ \frac{\Delta \vdash A : \kappa \qquad \Delta \vdash \kappa = \kappa' : \Box}{\Delta \vdash A : \kappa'}$$

$$(:i)\ \frac{i^A \in \Delta \qquad \Delta \vdash \Gamma}{\Delta;\Gamma \vdash i : A}$$

$$(\forall \mathsf{I} i)\ \frac{\cdot \vdash A : * \qquad \Delta, i^A;\Gamma \vdash t : B}{\Delta;\Gamma \vdash t : \forall\, i^A.\, B}\ \left(i \notin \mathrm{FV}(t) \cup \mathrm{FV}(\Gamma)\right)$$

$$(\forall \mathsf{E} i)\ \frac{\Delta;\Gamma \vdash t : \forall\, i^A.\, B \qquad \Delta;\cdot \vdash s : A}{\Delta;\Gamma \vdash t : B[s/i]}$$

Index Erasure

Definition:

$$(A \to \kappa)^\circ = \kappa^\circ \qquad (\lambda i^A.\, F)^\circ = F^\circ \qquad (F\{s\})^\circ = F^\circ \qquad (\forall i^A.\, B)^\circ = B^\circ$$

$$(\Delta, i^A)^\circ = \Delta^\circ$$

Theorem:

$$\frac{\Delta \vdash \Gamma \qquad \Delta;\Gamma \vdash t : A}{\Delta^\circ;\Gamma^\circ \vdash t : A^\circ}\ \big(\mathsf{dom}(\Delta) \cap \mathrm{FV}(t) = \emptyset\big)$$

Corollary:

Strong normalisation and logical consistency.

Mendler-style Recursive Types

- Notation:

  For $F, G : A \to *$,

  $$[F \to G] \triangleq \forall i^A.\, F\{i\} \to G\{i\} : *$$

- Recursive types:

  $$\begin{aligned} \mu_{A\to *} \quad &: \quad \big((A \to *) \to A \to *\big) \to A \to * \\ &\triangleq \quad \lambda F^{((A\to *)\to A\to *)}.\, \lambda i^A.\, \forall X^{(A\to *)}. \\ &\qquad \big(\forall R^{(A\to *)}.\, [R \to X] \to [F\,R \to X]\big) \to X\{i\} \end{aligned}$$

- Iterator and constructor:

$$
\begin{aligned}
\mathtt{mit} \quad &: \quad \forall F^{((A\to *)\to A\to *)}.\ \forall X^{(A\to *)}. \\
&\qquad \left(\forall R^{(A\to *)}.\ \left[R \to X\right] \to \left[F\,R \to X\right]\right) \to \left[\mu_{A\to *} F \to X\right] \\
&\triangleq \quad \lambda f.\,\lambda x.\,x\,f \\
\mathtt{cons} \quad &: \quad \forall F^{((A\to *)\to A\to *)}.\ \left[F\left(\mu_{A\to *}\,F\right) \to \mu_{A\to *}\,F\right] \\
&\triangleq \quad \lambda x.\,\lambda f.\,f\left(\mathtt{mit}\,f\right)x
\end{aligned}
$$

NB: $\mathtt{mit}\ f\ (\mathtt{cons}\ x) \rightsquigarrow f\ (\mathtt{mit}\ f)\ x$

Leibniz Equalities

- $|\kappa| \quad : \quad \kappa \to \kappa \to *$

  $\triangleq \quad \lambda\, X^{\kappa}.\, \lambda\, Y^{\kappa}.\, \forall\, P^{\kappa \to *}.\, P\, X \to P\, Y$

- $|A| \quad : \quad A \to A \to *$

  $\triangleq \quad \lambda\, i^{A}.\, \lambda\, j^{A}.\, \forall\, P^{A \to *}.\, P\, \{i\} \to P\, \{j\}$

Syntactic Kan Extensions

- Notation:

  For $h : A \to B, G : B \to *$, $G \circ h \triangleq \lambda i^A.\, G\{h\, i\}$.

Syntactic Kan Extensions

- Notation:

  For $h : A \to B, G : B \to *$, $\quad G \circ h \triangleq \lambda\, i^{A}.\, G\,\{h\, i\}$.

- Right Kan extension:

$$F : A \to *, h : A \to B$$

$$\vdash \underbrace{\lambda\, j^{B}.\, \forall\, i^{A}.\, |B|\,\{j\}\,\{h i\} \to F\{i\}}_{\mathrm{Ran}_{A,B}\,\{h\}\,F} \;:\; B \to *$$

- Ran property:

$$F : A \to *, G : B \to *, h : A \to B$$

$$\vdash \big[\,(G \circ h) \to F\,\big] \lhd \big[\,G \to \mathrm{Ran}_{A,B}\,\{h\}\,F\,\big]$$

- Application:

  For $\mathtt{Vec}\ A = \mu_{\mathtt{Nat}\to *}(V\ A)$, define a safe tail function of type

  $$\left[\ (\mathtt{Vec}\ A) \circ \mathtt{S} \to \mathtt{Vec}\ A\ \right]$$

- Application:

  For $\mathtt{Vec}\ A = \mu_{\mathtt{Nat}\to *}(V\ A)$, define a safe tail function of type

  $$\left[\ (\mathtt{Vec}\ A) \circ \mathtt{S} \to \mathtt{Vec}\ A\ \right]$$

  by defining a function of type

  $$\left[\ \mathtt{Vec}\ A \to \mathrm{Ran}_{\mathtt{Nat},\mathtt{Nat}}\,\{\mathtt{S}\}\,(\mathtt{Vec}\ A)\ \right]$$

  by iteration and retracting it to one of the desired type.

$$
\begin{aligned}
V \quad &: \quad * \longrightarrow (\mathtt{Nat} \longrightarrow *) \longrightarrow \mathtt{Nat} \longrightarrow * \\
&\triangleq \quad \lambda\, A^{*}.\, \lambda\, X^{\mathtt{Nat}\to *}.\, \lambda\, \mathtt{j}^{\mathtt{Nat}}. \\
&\qquad \left(\exists\, \mathtt{i}^{\mathtt{Nat}}.\, |\mathtt{Nat}|\{\mathtt{j}\}\{\mathtt{S}\,\mathtt{i}\} \times A \times X\{\mathtt{i}\}\right) \;+\; |\mathtt{Nat}|\{\mathtt{j}\}\{\mathtt{Z}\}
\end{aligned}
$$

$$
\begin{aligned}
V \quad &: \quad * \to (\mathtt{Nat} \to *) \to \mathtt{Nat} \to * \\
&\triangleq \quad \lambda\, A^{*}.\, \lambda\, X^{\mathtt{Nat}\to *}.\, \lambda\, j^{\mathtt{Nat}}. \\
&\qquad \underbrace{\left(\exists\, i^{\mathtt{Nat}}.\, |\mathtt{Nat}|\,\{j\}\{\mathtt{S}\, i\} \times A \times X\{i\}\right)}_{\mathrm{Lan}_{\mathtt{Nat},\mathtt{Nat}}\,\{\mathtt{S}\}\,\left(\lambda\, i^{\mathtt{Nat}}.\, A \times X\{i\}\right)} \;+\; |\mathtt{Nat}|\,\{j\}\{\mathtt{Z}\}
\end{aligned}
$$

- Left Kan extension:

$$
\begin{aligned}
&F : A \to *,\, h : A \to B \\
&\quad \vdash \underbrace{\lambda\, j^{B}.\, \exists\, i^{A}.\, |B|\,\{j\}\,\{h\, i\} \times F\{i\}}_{\mathrm{Lan}_{A,B}\,\{h\}\,F} \;:\; B \to *
\end{aligned}
$$

- Lan property:

$$
\begin{aligned}
&F : A \to *,\, G : B \to *,\, h : A \to B \\
&\quad \vdash \left[\, F \to (G \circ h)\,\right] \lhd \left[\, \mathrm{Lan}_{A,B}\,\{h\}\,F \to G\,\right]
\end{aligned}
$$

System F_i provides foundations for the design of the programming language

Nax

- Supports programming with recursion schemes.
- Logical consistency for *verification*.
- Light-weighted *indexed programming*.
- *Type inference* with minimal annotation.