Slide 1

The Curry-Howard Isomorphism: Remarks on Recursive Types[a]

Paweł Urzyczyn[b]
Warsaw University
Edinburgh, April 13 ;-) 1999

[a] Corrected version
[b] This lecture is based on a joint work with Zdzisław Spławski

Slide 2

Example: integer lists

$$< list >::= \texttt{nil} \mid \ < number >::< list >$$

This is a countable fixpoint: type **list** is the union of a sequence

$$\{\},\ \mathcal{F}(\{\}),\ \mathcal{F}^2(\{\}),\ \mathcal{F}^3(\{\}),\ldots$$

where $\mathcal{F}(\alpha) = \{a :: \ell \mid a \in \mathbf{int} \text{ and } \ell \in \alpha\}$.

Induction axiom (scheme):

$$\forall R(R(\texttt{nil}) \to \forall x(R(x) \to \forall a^{\mathbf{int}} R(a :: x)) \to \forall x^{\mathbf{list}} R(x))$$

Slide 3

Formulas-as-(dependent) types

formulas	$\sim$	types
proofs	$\sim$	terms
universe of individuals	$\sim$	type constant $\top$
specific domains (predicates)	$\sim$	type constants
algebraic terms	$\sim$	terms of ground types
predicate letters	$\sim$	type constructors
atomic formula $R(t_1, \ldots, t_n)$	$\sim$	dependent type $Rt_1 \ldots t_n$
implication	$\sim$	function type
universal formula	$\sim$	product type

Slide 4

Individuals and constructors:

int, **list**	$\sim$	nullary predicates
$s : \mathbf{int} \to \mathbf{int}$	$\sim$	a proof constant
$\underline{0}$, ::	$\sim$	proof constants
$\underline{3}$	$\sim$	a proof of predicate **int**
$\underline{3} :: \underline{7} :: \texttt{nil}$	$\sim$	a proof of predicate **list**
any individual	$\sim$	a proof of $\top$

Proof terms:

Slide 5

implication introduction	$\sim$	abstraction
		$(\lambda X : \psi . M^{\varphi}) : \psi \to \varphi$
proof by generalization	$\sim$	abstraction
		$(\lambda x : \top . M^{\varphi}) : (\forall x : \top)\varphi$
second-order generalization	$\sim$	abstraction
		$\lambda R . M^{\varphi}[R] : \forall R\, \varphi[R]$
proof by *modus ponens*	$\sim$	application $M^{\varphi \to \psi} N^{\varphi} : \psi$
first-order instantiation	$\sim$	application $M^{\forall x : \top\, \varphi(x)} t^{\top} : \varphi(t)$
second-order instantiation	$\sim$	type application
		$M^{\forall R \varphi[R]} \tau : \varphi[\tau]$

Dependency-erasing map:

Slide 6

predicate letter	$\Rightarrow$	type variable
individual terms	$\Rightarrow$	nothing
specific domain, e.g. **int**	$\Rightarrow$	type constant
bounded quantifier	$\Rightarrow$	implication
$\kappa[(\forall x : \mathbf{int})\varphi(x)]$	$:=$	$\mathbf{int} \to \kappa[\varphi]$
unbounded quantifier	$\Rightarrow$	nothing
$\kappa[(\forall x : \top . \varphi(x)]$	$:=$	$\kappa[\varphi]$

Slide 7

Proof reduction: integers

An induction scheme:

$$\forall x(\varphi(x) \to \varphi(s(x))) \to \varphi(\underline{0}) \to \forall x^{\mathbf{int}}\varphi(x)$$

To prove $\varphi(\underline{3})$ from $A : \forall x(\varphi(x) \to \varphi(s(x)))$ and $B : \varphi(\underline{0})$ one can derive $\forall x^{\mathbf{int}}\varphi(x)$ by induction, and instantiate x to $\underline{3}$,

or

directly apply A three times to B:

$$A\underline{2}(A\underline{1}(A\underline{0}B)) : \varphi(\underline{3})$$

Slide 8

An induction scheme:

$$\forall x(\varphi(x) \to \varphi(s(x))) \to \varphi(\underline{0}) \to \forall x^{\mathbf{int}}\varphi(x)$$

Proof of $\varphi(\underline{k+1})$ by induction reduces to an application of $A : \forall x(\varphi(x) \to \varphi(s(x)))$ to $\underline{k} : \top$ and a proof of $\varphi(\underline{k})$.

Type of iterator:

$$\mathbf{It}_\varphi : (\varphi \to \varphi) \to \varphi \to \mathbf{int} \to \varphi$$

Reduction rules for iterator:

$$\mathbf{It}_\varphi AB\underline{0} \Rightarrow B;$$

$$\mathbf{It}_\varphi AB\underline{k+1} \Rightarrow A(\mathbf{It}_\varphi AB\underline{k}).$$

Slide 9

Another variant of the induction scheme:

$$\forall x^{\mathbf{int}}(\varphi(x) \to \varphi(s(x))) \to \varphi(\underline{0}) \to \forall x^{\mathbf{int}}\varphi(x)$$

Proof of $\varphi(\underline{k+1})$ by induction reduces to an application of $A : \forall x^{\mathbf{int}}(\varphi(x) \to \varphi(s(x)))$ to $\underline{k} : \mathbf{int}$ and a proof of $\varphi(\underline{k})$.

Type of recursor:

$$\mathbf{R}_\varphi : (\mathbf{int} \to \varphi \to \varphi) \to \varphi \to \mathbf{int} \to \varphi$$

Reduction rules for recursor:

$$\mathbf{R}_\varphi AB\underline{0} \Rightarrow B;$$

$$\mathbf{R}_\varphi AB\underline{k+1} \Rightarrow A\underline{k}(\mathbf{R}_\varphi AB\underline{k}).$$

Slide 10

Three variants of the induction axiom:

a) $\forall R(\forall x(R(x) \to R(s(x))) \to R(\underline{0}) \to \forall x^{\mathbf{int}}R(x))$

b) $\forall R(\forall x^{\mathbf{int}}(R(x) \to R(s(x))) \to R(\underline{0}) \to \forall x^{\mathbf{int}}R(x))$

c) $\forall R(\forall x(R(x) \to R(s(x))) \to R(\underline{0}) \to \forall x R(x))$

Three different erasures:

a) $\mathbf{It}_\varphi : (\varphi \to \varphi) \to \varphi \to \mathbf{int} \to \varphi$

b) $\mathbf{R}_\varphi : (\mathbf{int} \to \varphi \to \varphi) \to \varphi \to \mathbf{int} \to \varphi$

c) $\lambda fx.f^k(x) : (\varphi \to \varphi) \to \varphi \to \varphi$

Slide 11

A generalization

New clause in the definition of types: If τ is a type and α does not occur negatively in τ then $\mu\alpha\tau$ is a type.

(Intention: Least fixpoint obtained as union of $\tau^k(\perp)$.)
Abbreviation μ for $\mu\alpha\tau$.

One approach: postulate equation $\mu = \tau(\mu)$;

Another approach: interprete members of $\tau(\mu)$ as members of μ, via a new constant:

$$\mathbf{in} : \tau(\mu) \to \mu$$

Note: this constant represents a family of embeddings

$$\mathbf{in}^k : \tau(\mu^k) \to \mu^{k+1}.$$

Slide 12

Function definitions by induction

Induction base: Just one trivial function $\perp : \perp \to \sigma$.

Induction step: To define $f^{k+1} : \mu^{k+1} \to \sigma$ from $f^k : \mu^k \to \sigma$ one needs a transformation $A\mu^k : (\mu^k \to \sigma) \to \tau(\mu^k) \to \sigma$. This normally comes as a generic operator

$$A : \forall\alpha((\alpha \to \sigma) \to \tau(\alpha) \to \sigma).$$

System $\boldsymbol{\lambda}$2I

Type of eliminator:

$$\mathbb{I} : \forall\beta(\forall\alpha((\alpha \to \beta) \to (\tau(\alpha) \to \beta)) \to \mu \to \beta)$$

Reduction scheme:

$$\mathbb{I}\sigma A(\mathbf{in}\, x) \Rightarrow A\mu(\mathbb{I}\sigma A)x$$

Explanation: $(\mathbb{I}\sigma A)^{k+1}(\mathbf{in}\, x) \Rightarrow A\mu^k(\mathbb{I}\sigma A)^k x$, where $x \in \tau(\mu^k)$.

Slide 13

Types of operators:

$$\mathbf{in} : \tau(\mu) \to \mu \qquad \mathbb{I} : \forall\beta(\forall\alpha((\alpha \to \beta) \to (\tau(\alpha) \to \beta)) \to \mu \to \beta)$$

Read it again this way:

$$(\mathbf{in})\ \tau(\mu) \subseteq \mu \qquad (\mathbb{I})\ \frac{\forall\alpha(\alpha \subseteq \beta \to \tau(\alpha) \subseteq \beta)}{\mu \subseteq \beta}$$

Second-order interpretation:

The following second-order axioms assert that L is the least fixpoint of $\Phi[R]$ seen as a monotone operator on R:

$$\forall x(\Phi[L](x) \to L(x));$$

$$\forall R(\forall P(P \subseteq R \to \Phi[P] \subseteq R) \to L \subseteq R).$$

Slide 14

Induction to iteration again

Induction axiom (for $L = \mathrm{lfp}\Phi$):

$$\forall R(\forall P(\forall x(P(x) \to R(x)) \to \forall x(\Phi[P](x) \to R(x)) \to \forall x(L(x) \to R(x))$$

Eliminator:

$$\mathbb{I} : \forall\beta(\forall\alpha((\alpha \to \beta) \to (\tau(\alpha) \to \beta)) \to \mu \to \beta)$$

Read the reduction scheme $\mathbb{I}\sigma A(\mathbf{in}\,x) \Rightarrow A\mu(\mathbb{I}\sigma A)x$ as a proof normalization rule.

A simplification:

Slide 15

$$(\mathbb{I})\ \frac{\forall\alpha(\alpha \subseteq \beta \to \tau(\alpha) \subseteq \beta)}{\mu \subseteq \beta} \quad \Longrightarrow \quad (\mathbb{J})\ \frac{\tau(\beta) \subseteq \beta}{\mu \subseteq \beta}$$

Note: Conditions $\tau(\beta) \subseteq \beta$ and $\forall\alpha(\alpha \subseteq \beta \to \tau(\alpha) \subseteq \beta)$ are equivalent but monotonicity is needed to prove left-to-right direction.

System $\boldsymbol{\lambda}$2J

Slide 16

Type of eliminator:

$$\mathbb{J} : \forall\beta((\tau(\beta) \to \beta) \to \mu \to \beta)$$

Reduction scheme:

$$\begin{tikzcd}
\mu \arrow[r, "(\mathbb{J}\sigma A)"] & \sigma \\
\tau(\mu) \arrow[u, "\mathbf{in}"] \arrow[r, "Lift(\mathbb{J}\sigma A)"'] & \tau(\sigma) \arrow[u, "A"']
\end{tikzcd}$$

Slide 17

Explanation for the reduction scheme of $\boldsymbol{\lambda}$**2J**

$$\begin{tikzcd} \mu^{k+1} \arrow[r, "(\mathbb{J}\sigma A)^{k+1}"] & \sigma \\ \tau(\mu^k) \arrow[u, "\mathbf{in}"] \arrow[r, "Lift(\mathbb{J}\sigma A)^k"'] & \tau(\sigma) \arrow[u, "A"'] \end{tikzcd}$$

Slide 18

Second-order definition of fixpoint:

$$L(x) := \forall R((\Phi[R] \subseteq R) \to R(x))$$

Translation of $\boldsymbol{\lambda}$2J to $\boldsymbol{\lambda}$2

$$\mu_J = \forall\beta((\tau(\beta) \to \beta) \to \beta)$$

$$\mathbb{J} = \Lambda\alpha\lambda y^{\tau(\alpha)\to\alpha}\lambda z^{\mu_J}.z\alpha y$$
$$\mathbf{in} = \lambda x^{\tau(\mu_J)}\Lambda\beta\lambda y^{\tau(\beta)\to\beta}.y(Lift(\mathbb{J}\beta y)x)$$

This is correct:

$$\mathbb{J}\,\sigma\,M\,(\mathbf{in}\,N) \twoheadrightarrow_\beta M\,(Lift(\mathbb{J}\,\sigma\,M)\,N)$$

Slide 19

Translation of $\boldsymbol{\lambda}$2I to $\boldsymbol{\lambda}$2

$$\mu_I = \forall\beta(\forall\alpha((\alpha \to \beta) \to \tau(\alpha) \to \beta) \to \beta)$$

$$\mathbb{I} = \Lambda\beta\lambda y^{\forall\alpha((\alpha\to\beta)\to\tau(\alpha)\to\beta)}\lambda z^{\mu_I}.z\beta y,$$
$$\mathbf{in} = \lambda x^{\tau(\mu_I)}\Lambda\beta\lambda y^{\forall\alpha((\alpha\to\beta)\to\tau(\alpha)\to\beta)}.y\mu(\mathbb{I}\ \beta\ y)x.$$

This is correct:

$$\mathbb{I}\,\sigma\,M\,(\mathbf{in}\,N) \twoheadrightarrow_\beta M\,\mu_I\,(\mathbb{I}\,\sigma\,M)\,N$$

Slide 20

Induction to recursion

Induction axioms (for $L = \mathrm{lfp}\,\Phi$):

a) $\forall R(\forall P(\forall x(P(x) \to R(x)) \to \forall x(\Phi[P](x) \to R(x)) \to$
$\to \forall x(L(x) \to R(x)));$

b) $\forall R(\forall x(\Phi[R](x) \to R(x)) \to \forall x(L(x) \to R(x)));$

c) $\forall R(\forall P(\forall x(P(x) \to L(x)) \to \forall x(P(x) \to R(x)) \to$
$\to \forall x(\Phi[P](x) \to R(x)) \to \forall x(L(x) \to R(x)));$

d) $\forall R(\forall x \in L(\Phi[R](x) \to R(x)) \to \forall x(L(x) \to R(x))).$

Axioms (a) and (b) erase to types of $\mathbb{I}$ and $\mathbb{J}$, respectively.

Axioms (c) and (d) erase to types of recursors.

Slide 21

System $\boldsymbol{\lambda}$2R

Type of recursor:

$$\mathbb{R} : \forall\beta(\forall\alpha((\alpha \to \mu) \to (\alpha \to \beta) \to \tau(\alpha) \to \beta) \to \mu \to \beta)$$

Reduction scheme:

$$\mathbb{R}\sigma A(\mathbf{in}\, x) \Rightarrow A\mu(\lambda z^{\mu}.z)(\mathbb{R}\sigma A)x$$

System $\boldsymbol{\lambda}$2Rec

Type of recursor:

$$\mathbf{Rec} : \forall\beta((\tau(\mu \times \beta) \to \beta) \to \mu \to \beta)$$

Reduction scheme:

$$\mathbf{Rec}\, \sigma A(\mathbf{in}\, x) \Rightarrow A(\mathit{Lift}(\mathbf{I}_{\mu} \times \mathbf{Rec}\, \sigma A)x)$$

Slide 22

System $\boldsymbol{\lambda}$2U

Embedding:

$$\mathbf{Fold} : \tau[\mu] \to \mu$$

Projection:

$$\mathbf{Unfold} : \mu \to \tau[\mu]$$

Reduction:

$$\mathbf{Unfold}(\mathbf{Fold}\, x) \Rightarrow x$$

Slide 23

Fact 1 (old): Systems $\boldsymbol{\lambda}$**2U**, $\boldsymbol{\lambda}$**2R** and $\boldsymbol{\lambda}$**2Rec** are definable within each other.

Fact 2 (new): System $\boldsymbol{\lambda}$**2U** (so the other two too) is *not* definable within $\boldsymbol{\lambda}$**2**.

Conclusion: Iterators cannot be simulated by recursors.

Slide 24

Hint for Fact 2:

Type embedding: Write $\rho \leq \tau$, iff there are
$F : \rho \to \tau$ and $G : \tau \to \rho$, such that $G \circ F =_\beta \mathbf{I}$.

Theorem: If $\rho \leq \tau$ then $|\rho| \leq |\tau|$.

Proof method: Geometry of Interaction.

Slide 25

This material is available from
`http://zls.mimuw.edu.pl/~urzy/ftp.html`