# On Typed Lambda Definability and Normalization by Evaluation

Andreas Abel[1]

[1]Department of Computer Science and Engineering
Chalmers and Gothenburg University, Sweden

IFIP 1.3 Meeting 2018
Royal Holloway University, London, UK
6 July 2018

# Strong Typing

- Typing allows correct instruction selection (compilation).
- Typing prevents basic runtime errors.

  *Well-typed programs don't go wrong.*
  *(Robin Milner, 1978)*

- But there is more...

# Free Theorems from Typing

- Reynolds 1974, Wadler 1989: Theorems for free!
  Polymorphic functions preserve all relations.
- Free theorems from linearity:

**Theorem**

*Every function* $\mathsf{List}\,A \multimap \mathsf{List}\,A$ *is a permutation.*
*(For an abstract type* $A$*.)*

- Free theorems proven by logical relations.
- What about free theorems from simple typing?

**Theorem**

*In STLC with the only constants* $\mathsf{true}, \mathsf{false} : \mathsf{Bool}$, *boolean negation is not definable.*

# STLC-Definability

- *Given a function $f$ of some type, is it definable in STLC?*
  (Replace simply-typed lambda calculus (STLC) by your favorite type theory.)
- Extended question: Can we decide whether $f$ is STLC-definable?
- Trivial answer to original question:

$$f \text{ STLC-definable} \iff \exists t.\ (\!|t|\!) = f$$

- Modified question: Can we characterize the STLC-definable functions without referencing STLC-syntax?

# A Universe of Types

- To talk about typed functions, we need a language of types.

$$\begin{array}{llllll} \iota & : & \mathsf{Base} & & & \text{base type} \\ S, T, U & : & \mathsf{Ty} & ::= & \iota \mid U \Rightarrow T & \text{simple type hierarchy} \end{array}$$

- Interpretation $(\!|\_|\!) : \mathsf{Ty} \to \mathsf{Set}$.

$$\begin{array}{llll} (\!|\iota|\!) & = & \mathit{parameter} & \\ (\!|U \Rightarrow T|\!) & = & (\!|U|\!) \to (\!|T|\!) & \text{full (meta-theoretic) function space} \end{array}$$

- Constants $c : T_c$ need to satisfy $(\!|c|\!) : (\!|T_c|\!)$.

# Contexts

- Types of argument lists (contexts).

$$\begin{array}{llll} \Gamma, \Delta : \mathsf{Cxt} & ::= & \emptyset & \text{empty context} \\ & \mid & \Gamma.U & \text{context extension} \end{array}$$

- Interpretation $(\!|\_|\!) : \mathsf{Cxt} \to \mathsf{Set}$.

$$\begin{array}{lcll} (\!|\emptyset|\!) & = & 1 & \text{unit set} \\ (\!|\Gamma.U|\!) & = & (\!|\Gamma|\!) \times (\!|U|\!) & \text{cartesian product} \end{array}$$

# Contexts as Worlds

- Context thinning $\Gamma \twoheadrightarrow \Delta$.

$$\frac{}{\mathrm{id}_\Gamma : \Gamma \twoheadrightarrow \Gamma} \qquad \frac{\tau : \Gamma \twoheadrightarrow \Delta}{\mathrm{weak}_U\, \tau : \Gamma.U \twoheadrightarrow \Delta} \qquad \frac{\tau : \Gamma \twoheadrightarrow \Delta}{\mathrm{lift}_U\, \tau : \Gamma.U \twoheadrightarrow \Delta.U}$$

- Makes the *category of contexts and order-preserving embeddings*.
- Interpretation $(\!|\_|\!) : (\Gamma \twoheadrightarrow \Delta) \to (\!|\Gamma|\!) \to (\!|\Delta|\!)$ as sublist projection.

$$\begin{array}{lcll}
(\!|\mathrm{id}_\Gamma|\!) & = & \mathrm{id}_{(\!|\Gamma|\!)} & : & (\!|\Gamma|\!) \to (\!|\Gamma|\!) \\
(\!|\mathrm{weak}_U\, \tau|\!) & = & (\!|\tau|\!) \circ \pi_1 & : & (\!|\Gamma.U|\!) \to (\!|\Delta|\!) \\
(\!|\mathrm{lift}_U\, \tau|\!) & = & (\!|\tau|\!) \times \mathrm{id}_{(\!|U|\!)} & : & (\!|\Gamma.U|\!) \to (\!|\Delta.U|\!)
\end{array}$$

# Kripke predicates in the world of contexts

- Predicate $f \in [\![T]\!]_\Gamma$ on $f : (\!|\Gamma|\!) \to (\!|T|\!)$ needs to satisfy
  1. (Monotonicity:) If $f \in [\![T]\!]_\Gamma$ and $\tau : \Delta \twoheadrightarrow \Gamma$ then $f \circ (\!|\tau|\!) \in [\![T]\!]_\Delta$.
  2. (Kripke function space:) $f \in [\![U \to T]\!]_\Gamma$ iff $f \overset{\tau}{\cdot} d \in [\![T]\!]_\Delta$ for all $\tau : \Delta \twoheadrightarrow \Gamma$ and $d \in [\![U]\!]_\Delta$.
     Herein: $(f \overset{\tau}{\cdot} d)\,\delta = f\,((\!|\tau|\!)\,\delta)\,(d\,\delta)$.
  3. $(\!|c|\!) \in [\![T_c]\!]_\emptyset$ for all constants $c$.
- Base case $[\![\iota]\!]_\Gamma$ is parameter (must be monotone!).

**Theorem**

*A function $f : (\!|\Gamma|\!) \to (\!|T|\!)$ is STLC-definable iff it satisfies all Kripke predicates, i.e., $f \in [\![T]\!]_\Gamma$ no matter how $[\![\iota]\!]$ is chosen.*

$\Rightarrow$ If $t : (\Gamma \vdash T)$ then $(\!|t|\!) \in [\![T]\!]_\Gamma$ (fundamental theorem of LR).
$\Leftarrow$ $\Sigma\, t : (\Gamma \vdash T).\ (\!|t|\!) = f$ is a Kripke predicate $f \in [\![T]\!]_\Gamma$ (term model).

# Application: Refuting STLC-definability

**Theorem**

*Boolean negation is not definable in STLC equipped with* $\mathsf{true}, \mathsf{false} : \mathsf{Bool}$.

- Proof 1: Look at possible normal forms of type $\mathsf{Bool} \to \mathsf{Bool}$.
- Proof 2: Construct a Kripke countermodel.
  - Let $f \in [\![\mathsf{Bool}]\!]_\Gamma$ iff $f$ is constant $\mathsf{true}/\mathsf{false}$ or a projection from $(\!|\Gamma|\!)$.
  - This is a Kripke model for STLC with $\mathsf{true}, \mathsf{false} : \mathsf{Bool}$.
  - Negation is neither constant nor a projection.
- By the connection between STLC-definability and normalization, these two proofs are somewhat "the same".

**Theorem (Peirce not inhabited)**

*There is not closed STLC-term of type* $((A \Rightarrow B) \Rightarrow A) \Rightarrow A$ *for some types* $A, B$.

Proof: Exercise!

# Syntax and Interpretation of STLC

- Variables: index $x : \mathsf{Var}\ \Gamma\ T$ into the context.

$$\frac{}{\mathsf{v}_0 : \mathsf{Var}\ \Gamma.T\ T} \qquad \frac{\mathsf{v}_i : \mathsf{Var}\ \Gamma\ T}{\mathsf{v}_{i+1} : \mathsf{Var}\ \Gamma.U\ T}$$

- Interpretation $(\!|\_|\!) : \mathsf{Var}\ \Gamma\ T \to (\!|\Gamma|\!) \to (\!|T|\!)$ as projections.

$$\begin{array}{rcl} (\!|\mathsf{v}_0|\!) & = & \pi_2 \\ (\!|\mathsf{v}_{i+1}|\!) & = & (\!|\mathsf{v}_i|\!) \circ \pi_1 \end{array}$$

- Terms $t : \Gamma \vdash T$.

$$\frac{x : \mathsf{Var}\ \Gamma\ T}{x : \Gamma \vdash T} \qquad \frac{t : \Gamma.U \vdash T}{\lambda t : \Gamma \vdash U \Rightarrow T} \qquad \frac{t : \Gamma \vdash U \Rightarrow T \qquad u : \Gamma \vdash U}{t\,u : \Gamma \vdash T}$$

- Interpretation $(\!|\_|\!) : (\Gamma \vdash T) \to (\!|\Gamma|\!) \to (\!|T|\!)$.

$$\begin{array}{rclcl} (\!|\lambda t|\!) & = & \mathsf{curry}\ (\!|t|\!) & \qquad & \mathsf{curry}\ f\ (\gamma, d) = f\ \gamma\ d \\ (\!|t\ u|\!) & = & \mathsf{S}\ (\!|t|\!)\ (\!|u|\!) & \qquad & \mathsf{S}\ f\ g\ \gamma \qquad = f\ \gamma\ (g\ \gamma) \end{array}$$

# Fundamental theorem

- Extension to environments: $\rho \in [\![\Gamma]\!]_\Delta$ for $\rho : (\!|\Delta|\!) \to (\!|\Gamma|\!)$.

$$\begin{array}{lcl} \rho \in [\![\emptyset]\!]_\Delta & \iff & \text{true} \\ \rho \in [\![\Gamma.U]\!]_\Delta & \iff & \pi_1 \circ \rho \in [\![\Gamma]\!]_\Delta \text{ and } \pi_2 \circ \rho \in [\![U]\!]_\Delta \end{array}$$

Monotonicity: If $\rho \in [\![\Gamma]\!]_\Delta$ and $\tau : \Delta' \twoheadrightarrow \Delta$ then $\rho \circ (\!|\tau|\!) \in [\![\Gamma]\!]_{\Delta'}$.

**Theorem (Fundamental theorem of logical relations)**

*If $t : (\Gamma \vdash T)$ and $\rho \in [\![\Gamma]\!]_\Delta$ then $(\!|t|\!) \circ \rho \in [\![T]\!]_\Delta$.*

- Prove this first for $x : \mathsf{Var}\ \Gamma\ T$ (easy).
- Then prove by induction on $t : \Gamma \vdash T$.
- Case $\lambda t : \Gamma \vdash U \Rightarrow T$: Show $\mathsf{curry}(\!|t|\!) \circ \rho \in [\![U \Rightarrow T]\!]_\Delta$.
  (Needs monotonicity!)
- Case $t\ u : \Gamma \vdash T$: Show $(\mathsf{S}\ (\!|t|\!)\ (\!|u|\!)) \circ \rho \in [\![T]\!]_\Delta$.

# Term model

- Define $f \in [\![\iota]\!]_\Gamma$ as $\Sigma t : (\Gamma \vdash \iota).\ (\!|t|\!) = f$.

**Theorem (Reflect/reify)**

1. *If* $t : \Gamma \vdash T$ *then* $(\!|t|\!) \in [\![T]\!]_\Gamma$ *(reflect).*
2. *If* $f \in [\![T]\!]_\Gamma$ *then* $(\!|t|\!) = f$ *for some* $t : \Gamma \vdash T$ *(reify).*

- Prove simulateneously by induction on $T$.
- Discovery: does not introduce $\beta$-redexes!

# Normal forms

- Define simultaneously $t : \mathsf{Ne}\ \Gamma\ T$ (neutral) and $t : \mathsf{Nf}\ \Gamma\ T$ (normal).

$$\frac{x : \mathsf{Var}\ \Gamma\ T}{x : \mathsf{Ne}\ \Gamma\ T} \qquad \frac{t : \mathsf{Ne}\ \Gamma\ (U \Rightarrow T) \qquad u : \mathsf{Nf}\ \Gamma\ U}{t\, u : \mathsf{Ne}\ \Gamma\ T}$$

$$\frac{t : \mathsf{Ne}\ \Gamma\ T}{t : \mathsf{Nf}\ \Gamma\ T} \qquad \frac{t : \mathsf{Nf}\ \Gamma.U\ T}{\lambda t : \mathsf{Nf}\ \Gamma\ (U \Rightarrow T)}$$

- Define $f \in [\![\iota]\!]_\Gamma$ as $\Sigma(t : \mathsf{Ne}\ \Gamma\ \iota).\ (\!|t|\!) = f$.

**Theorem (Reflect/reify)**

1. *If* $t : \mathsf{Ne}\ \Gamma\ T$ *then* $(\!|t|\!) \in [\![T]\!]_\Gamma$ *(reflect).*
2. *If* $f \in [\![T]\!]_\Gamma$ *then* $(\!|t|\!) = f$ *for some* $t : \mathsf{Nf}\ \Gamma\ T$ *(reify).*

# Normalization by Evaluation

- Show $id_{(\!|\Gamma|\!)} \in [\![\Gamma]\!]_\Gamma$ (reflection!).
- Assume $t : \Gamma \vdash T$.
- By the fundamental theorem, $(\!|t|\!) \circ id : [\![T]\!]_\Gamma$.
- By reification, $(\!|t|\!) = (\!|v|\!)$ for some $v : \mathrm{Nf}\ \Gamma\ T$.

# Conclusions

- Proof-relevant version of completeness proof of IPL.
- Implemented in Agda with a tiny bit of `--rewriting`.
  https://github.com/andreasabel/lambda-definability/tree/master/src-stlc
- Extension to sum types:
  - Use Beth models to incorporate case trees.
  - Need lots of `--rewriting`.
- Aspired future work: Extension to dependent types.

# Related Work

- A. (habil. 2013): "Type-assignment NbE" for dependent and polymorphic types.
- What I presented here are classic results:
    - Friedman / Plotkin (1970s/80s): Logical relations.
    - Catarina Coquand (1993): NbE for STLC$\sigma$ using Kripke model
    - Jung, Tiuryn (TLCA 1993): More or less this formulation.
- Fiore, Simpson (TLCA 1999); Altenkirch, Dybjer, Hofmann, Scott (LICS 2001): Extension to disjoint sum types.
- Altenkirch Kaposi 2016: Extension to $\Pi$-types.