Testing map

$$\text{map} : \forall \alpha. \forall \beta. (\alpha \rightarrow \beta) \rightarrow \text{list}(\alpha) \rightarrow \text{list}(\beta)$$
Testing map

\[ \text{map} : \forall \alpha. \forall \beta. (\alpha \to \beta) \to \text{list}(\alpha) \to \text{list}(\beta) \]

\[ \alpha^* = \beta^* = \mathbb{N} \]

identity function and \([0, 1, \ldots, n-1] \)
Testing exists

exists : $\forall \alpha. (\alpha \rightarrow 2) \rightarrow \text{list}(\alpha) \rightarrow 2$

$\alpha^* = 2$

identity function and all lists
Testing pick

pick : ∀α. α × α → α

α* = 2

(true, false)
Theorem [Bernardy et al.]

\[ p : \forall \alpha. (F(\alpha) \to \alpha) \times (G(\alpha) \to K) \to H(\alpha) \]

\[ \alpha^* = \mu F \text{ if it exists} \]

\[ \rho^* : F(\alpha) \to \alpha \text{ catamorphism \text{ } \text{\textquotedblleft}roll\text{\textquotedblright}} \]
Theorem [Bernardy et al.]

\[ p : \forall \alpha. (F(\alpha) \rightarrow \alpha) \times (G(\alpha) \rightarrow K) \rightarrow H(\alpha) \]

\[ p : \forall \alpha. P(\alpha) \rightarrow H(\alpha) \]
My Problems with Embedding/Projection

Type $\mu F$ buried under layers of existence theorems.

Error-prone: e.g., wrongly assuming $\text{list}(\alpha) \subseteq \mathbb{N} \times (\mathbb{N} \rightarrow \alpha)$. 
p: ∀α. P(α) → H(α)
Logarithmic Conjecture

\[ \mu \alpha. \log_\alpha P(\alpha) \text{ if it exists} \]
Logarithmic Conjecture

\[ p : \forall \alpha. P(\alpha) \rightarrow H(\alpha) \]

\[ \alpha^* = \mu \alpha. \log_\alpha P(\alpha) \quad \text{if it exists} \]

Goal: simple and direct calculation
Testing pick

\[ \text{pick: } \forall \alpha. \alpha \times \alpha \rightarrow \alpha \]

\[ \alpha^* = \log_\alpha (\alpha \times \alpha) = \log_\alpha (\alpha^2) = 2 \]

“all locations of \( \alpha \)”
“all ways to generate an \( \alpha \)-element”
\log_\alpha \alpha = 1 \quad \log_\alpha K = 0
\[ \log_\alpha \alpha = 1 \quad \log_\alpha K = 0 \]

\[ \log_\alpha (A \times B) = \log_\alpha A + \log_\alpha B \]
\[ \log_\alpha (A + B) = \log_\alpha A \cup \log_\alpha B \]

\[ A \cup B \approx "\text{max}(A,B)" \]
can be \( A+B \), ideally more optimized
\[ \log_{\alpha} \alpha = 1 \quad \log_{\alpha} K = 0 \]

\[ \log_{\alpha} (A \times B) = \log_{\alpha} A + \log_{\alpha} B \]
\[ \log_{\alpha} (A + B) = \log_{\alpha} A \cup \log_{\alpha} B \]

\( A \cup B \approx \text{“max}(A, B)\)''

can be A+B, ideally more optimized

\[ \log_{\alpha} (A \cup B) = \log_{\alpha} A \cup \log_{\alpha} B \quad \text{optional} \]
\[
\begin{align*}
\log_\alpha \alpha &= 1 & \log_\alpha K &= 0 \\
\log_\alpha (A \times B) &= \log_\alpha A + \log_\alpha B \\
\log_\alpha (A+B) &= \log_\alpha A \cup \log_\alpha B \\
A \cup B &\approx \text{“max}(A,B)” \\
\text{can be } A+B, \text{ ideally more optimized} \\
\log_\alpha (A \cup B) &= \log_\alpha A \cup \log_\alpha B \text{ optional} \\
\log_\alpha (A \rightarrow B) &= A \times \log_\alpha B \\
\text{“}B^A\text{”}
\end{align*}
\]
Testing map

\[ \text{map} : \forall \alpha. \forall \beta. (\alpha \rightarrow \beta) \rightarrow \text{list}(\alpha) \rightarrow \text{list}(\beta) \]
Testing map

\[ \text{map} : \forall \alpha. \forall \beta. (\alpha \rightarrow \beta) \rightarrow \text{list}(\alpha) \rightarrow \text{list}(\beta) \]

\[ P(\alpha) = (\alpha \rightarrow \beta) \times \text{list}(\alpha) \]
Testing map

\[
\text{map} : \forall \alpha. \forall \beta. (\alpha \rightarrow \beta) \rightarrow \text{list}(\alpha) \rightarrow \text{list}(\beta)
\]

\[
P(\alpha) = (\alpha \rightarrow \beta) \times \text{list}(\alpha)
\]

\[
\log\alpha P(\alpha) = \log\alpha (\alpha \rightarrow \beta) + \log\alpha \text{list}(\alpha)
\]
Testing map

\[
\text{map} : \forall \alpha. \forall \beta. (\alpha \rightarrow \beta) \rightarrow \text{list}(\alpha) \rightarrow \text{list}(\beta)
\]

\[
P(\alpha) = (\alpha \rightarrow \beta) \times \text{list}(\alpha)
\]

\[
\log_\alpha P(\alpha) = \log_\alpha (\alpha \rightarrow \beta) + \log_\alpha \text{list}(\alpha)
\]

\[
= \alpha \times \log_\alpha \beta
\]

\[
= \alpha \times 0
\]

\[
\cong 0
\]
Rules of Recursive Logarithm

\[ X \cong P(X) \]
\[ \log_\alpha X \cong \log_\alpha P(X) \]
Rules of *Recursive* Logarithm

\[ X \equiv P(X) \]
\[ \log_\alpha X \equiv \log_\alpha P(X) \]

\[ \log_\alpha (\mu X. P) = \mu X'. (\log_\alpha P)[\mu X. P/X] \]
\[ \log_\alpha X = X' \]
Rules of *Recursive Logarithm*

\[
\text{list}(\alpha) = \mu X. 1 + \alpha \times X \\
\log_\alpha(\text{list}(\alpha)) = \mu X'. 0 \cup (1 + X') \\
\approx \mu X'. 1 + X' \\
= \mathbb{N}
\]

\[
\log_\alpha(\mu X.P) = \mu X'. (\log_\alpha P)[\mu X.P/X] \\
\log_\alpha X = X'
\]

I will assume \(0 \cup A = A\) from now on
Testing map

\[ \text{map} : \forall \alpha. \forall \beta. (\alpha \to \beta) \to \text{list}(\alpha) \to \text{list}(\beta) \]

\[
P(\alpha) = (\alpha \to \beta) \times \text{list}(\alpha)
\]

\[
\log_\alpha P(\alpha) = \log_\alpha (\alpha \to \beta) + \log_\alpha \text{list}(\alpha)
\]

\[
\approx 0 \quad \text{and} \quad = \mathbb{N}
\]
Logarithmic Conjecture

\[ p : \forall \alpha. P(\alpha) \rightarrow H(\alpha) \]

\[ \alpha^* = \mu \alpha. \log_\alpha P(\alpha) \quad \text{if it exists} \]

Goal: simple and direct calculation
Non-regular Recursion and Change of Bases / Chain Rules

\[ N(\alpha) \cong 1 + \alpha \times N(\alpha \times \alpha) \]

\[ \alpha - \alpha^2 - \alpha^4 - \alpha^8 - 1 \]

example stolen from Categories of Containers [Abbott et al.]
Non-regular Recursion and Change of Bases / Chain Rules

\[ N(\alpha) \cong 1 + \alpha \times N(\alpha \times \alpha) \]

\[ \alpha - \alpha^2 - \alpha^4 - \alpha^8 - 1 \]

\[ \log_\alpha N(\alpha) \cong 1 + "\log_{\alpha \times \alpha} N(\alpha \times \alpha)" \times \log_\alpha (\alpha \times \alpha) \]

\[ N'(\alpha) \cong 1 + N'(\alpha \times \alpha) \times 2 \]

example stolen from Categories of Containers [Abbott et al.]
\[ \log_\alpha P(Q(\alpha)) = P'(Q(\alpha)) \times Q'(\alpha) \]
Non-regular Recursion and Change of Bases / Chain Rules

\[ \log_\alpha P(Q(\alpha)) = P'(Q(\alpha)) \times Q'(\alpha) \]

*Should* work for arbitrary \( P \) with more equations

\[ \log_\alpha ((\alpha^2)^3) = 3 \times 2 \]
\[ \log_\alpha (\alpha^2 \times \alpha^2 \times \alpha^2) = 2 + 2 + 2 \]
F(α) \cong 1 + \alpha + \text{list}(\alpha) \times F(\alpha^2 + \alpha^3) \times \text{list}(\alpha)

(finger trees)

\text{list}(\alpha) \text{ may be optimized to } \alpha + \alpha^2 + \alpha^3 + \alpha^4
Non-regular Recursion and Change of Bases / Chain Rules

\[ F(\alpha) \cong 1 + \alpha + \text{list}(\alpha) \times F(\alpha^2 + \alpha^3) \times \text{list}(\alpha) \]

(finger trees)

\[ F'(\alpha) \cong 1 \cup (\mathbb{N} + F'(\alpha^2 + \alpha^3) \times (2 \cup 3) + \mathbb{N}) \]

\text{list}(\alpha) \text{ may be optimized to } \alpha + \alpha^2 + \alpha^3 + \alpha^4
Corollary of Conjecture

\[ p : \forall \alpha. (F(\alpha) \rightarrow \alpha) \times (G(\alpha) \rightarrow K) \rightarrow H(\alpha) \]
Corollary of Conjecture

\[ p : \forall \alpha. \ (F(\alpha) \to \alpha) \times (G(\alpha) \to K) \to H(\alpha) \]

\[
\log_\alpha((F(\alpha) \to \alpha) \times (G(\alpha) \to K)) \\
= \log_\alpha(F(\alpha) \to \alpha) + \log_\alpha(G(\alpha) \to K) \\
= F(\alpha) + \log_\alpha(G(\alpha) \to K)
\]

\[ \approx 0 \]

\[ \alpha^* = \mu \alpha.(F(\alpha) + 0) \approx \mu F \]
Corollary of Conjecture

\[ p : \forall \alpha. \ (F(\alpha) \to \alpha) \times (G(\alpha) \to K) \to H(\alpha) \]

\[
\log_\alpha((F(\alpha) \to \alpha) \times (G(\alpha) \to K)) \\
= \log_\alpha(F(\alpha) \to \alpha) + \log_\alpha(G(\alpha) \to K) \\
= F(\alpha) \quad \approx 0
\]

\[ \alpha^* = \mu \alpha. (F(\alpha) + 0) \cong \mu F \]

How to plug in \( \rho^* \), the catamorphism ("roll")?
Upgraded Logarithmic Conjecture

\[ p : \forall \alpha. P(\alpha) \rightarrow H(\alpha) \]

\[ \alpha^* = \mu \alpha. P'(\alpha) \text{ if it exists} \]

\[ \rho^+ : P^-(\alpha^*) \rightarrow P(\alpha^*) \]

to utilize

"roll" \[ \rho^* : P'(\alpha^*) \rightarrow \alpha^* \]
\[ \rho^+ : P^- (\alpha^*) \rightarrow P(\alpha^*) \]

To utilize

"roll" \[ \rho^* : P'(\alpha^*) \rightarrow \alpha^* \]

\( \rho^* \) fixes arguments at strictly positive locations

\[ P^- (\alpha^*) : \text{the residual} \]
Killing the Strictly Positive

\[
\begin{align*}
\alpha^- &= 1 \quad K^- &= K \\
(A \rightarrow B)^- &= A \rightarrow B^-
\end{align*}
\]

\[
\begin{align*}
(A \times B)^- &= A^- \times B^- \\
(A \cup B)^- &= A^- \cup B^- \\
(A + B)^- &= A^- + B^- \\
(\mu \beta. A)^- &= \mu \beta. A^-
\end{align*}
\]

\[
\rho^+ : P^-(\alpha^*) \rightarrow P(\alpha^*)
\]

residual \quad full
Corollary of *Upgraded* Conjecture

\[
p : \forall \alpha. (F(\alpha) \to \alpha) \times (G(\alpha) \to K) \to H(\alpha)
\]

\[
((F(\alpha) \to \alpha) \times (G(\alpha) \to K))^{-1}
\]

\[
= (F(\alpha) \to 1) \times (G(\alpha) \to K)
\]

\[
\approx 1
\]

\[
\rho^+(_, g) = (\rho^*, g)
\]

*with appropriate rules (omitted)*

\[
\rho^+ \text{ fills in the } \rho^* \text{ for you}
\]
Problems: Suboptimal Types

exists : ∀α.(α→2)→list(α)→2

α* = ℕ, but α = 2 with id seems better
similarly “all”, “toString”, “filter”, etc.
Problems: Suboptimal Types

\[\text{exists} : \forall \alpha. (\alpha \to \mathbb{N}) \to \text{list}(\alpha) \to \mathbb{N}\]
\[\alpha^* = \mathbb{N}, \text{ but } \alpha = 2 \text{ with id seems better}\]
\[\text{similarly ”all”, ”toString”, ”filter”, etc.}\]

\[\text{length} : \forall \alpha. \text{list}(\alpha) \to \mathbb{N}\]
\[\alpha^* = \mathbb{N}, \text{ but } \alpha = 1 \text{ suffices}\]
Problems: Suboptimal Types

\[ \exists \alpha. (\alpha \rightarrow 2) \rightarrow \text{list}(\alpha) \rightarrow 2 \]

\[ \alpha^* = \mathbb{N}, \text{ but } \alpha = 2 \text{ with id seems better} \]

\[ \text{similarly "all", "toString", "filter", etc.} \]

\[ \text{length : } \forall \alpha. \text{list}(\alpha) \rightarrow \mathbb{N} \]

\[ \alpha^* = \mathbb{N}, \text{ but } \alpha = 1 \text{ suffices} \]

Current: full distinctiveness

Next: inspectability
Future Work
Future Work

Conjectures $\rightarrow$ Theorems
Future Work

Conjectures $\rightarrow$ Theorems

Greatest fixed points, etc.
Future Work

Conjectures $\rightarrow$ Theorems

Greatest fixed points, etc.

Unknown *datatypes*, not just functions

Checker for parts of CMU 15-210 in 2015

General theory under development
Some Related Work

Testing Polymorphic Properties
Jean-Philippe Bernardy, Patrik Jansson, and Koen Claessen

Categories of Containers
Michael Abbott, Thorsten Altenkirch, and Neil Ghani

A semantics for shape
C. Barry Jay

Species and Functors and Types, Oh My!
Brent A. Yorgey
Unlike derivatives, logarithms differentiate.

credit to Danny Gratzer