Theorems *for* Free

Robert Atkey
bob.atkey@gmail.com

6th November 2012

For any program M with the type:

$$\forall\alpha.\ \mathsf{List}\ \alpha \to \mathsf{List}\ \alpha$$

For any program M with the type:

$$\forall\alpha.\ \mathsf{List}\ \alpha \to \mathsf{List}\ \alpha$$

Deduce the following "Free Theorem":

for all types X, Y,
for all functions $f : X \to Y$,
for all lists $l : \mathsf{List}\ X$,
$M\,(\mathrm{map}\ f\ l) = \mathrm{map}\ f\,(M\ l)$.

For any program M with the type:

$$\forall\alpha.\ \mathsf{List}\ \alpha \to \mathsf{List}\ \alpha$$

Deduce the following "Free Theorem":

for all types X, Y,
for all functions $f : X \to Y$,
for all lists $l : \mathsf{List}\ X$,
$M\ (\mathrm{map}\ f\ l) = \mathrm{map}\ f\ (M\ l)$.

Without looking at the implementation of M.

Polymorphism

Ad-hoc Polymorphism

Ad-hoc Polymorphism

$$\left.\begin{array}{l}\text{add} : \mathsf{Integer} \rightarrow \mathsf{Integer} \rightarrow \mathsf{Integer} \\ \text{add} : \mathsf{Float} \rightarrow \mathsf{Float} \rightarrow \mathsf{Float}\end{array}\right\} \textit{multiple implementations}$$

implementation chosen at either compile-time or run-time

Ad-hoc Polymorphism

$$\left.\begin{array}{l}\text{add} : \mathsf{Integer} \rightarrow \mathsf{Integer} \rightarrow \mathsf{Integer} \\ \text{add} : \mathsf{Float} \rightarrow \mathsf{Float} \rightarrow \mathsf{Float}\end{array}\right\} \textit{multiple implementations}$$

implementation chosen at either compile-time or run-time

Parametric Polymorphism

Ad-hoc Polymorphism

$$\left.\begin{array}{l}\mathrm{add} : \mathsf{Integer} \to \mathsf{Integer} \to \mathsf{Integer} \\ \mathrm{add} : \mathsf{Float} \to \mathsf{Float} \to \mathsf{Float}\end{array}\right\} \textit{multiple implementations}$$

implementation chosen at either compile-time or run-time

Parametric Polymorphism

$$\forall \alpha.\ \mathsf{List}\ \alpha \to \mathsf{List}\ \alpha \ \Big\} \textit{single implementation}$$

single implementation works for all types

John Reynolds' Idea

John Reynolds' Idea (1983)

If we have a single implementation that works for all types, then it must be uniform in some way.

John Reynolds' Idea (1983)

If we have a single implementation that works for all types, then it must be uniform in some way. We can capture uniformity through *preservation of relations*.

John Reynolds' Idea (1983)

If we have a single implementation that works for all types, then it must be uniform in some way. We can capture uniformity through *preservation of relations*.

Preservation of Relations

$$M : \forall\alpha.\ \mathsf{List}\ \alpha \to \mathsf{List}\ \alpha$$

John Reynolds' Idea (1983)

If we have a single implementation that works for all types, then it must be uniform in some way. We can capture uniformity through *preservation of relations*.

Preservation of Relations

$$M : \forall\alpha.\ \mathsf{List}\ \alpha \to \mathsf{List}\ \alpha$$

Assume two types X and Y and a relation $R : X \leftrightarrow Y$

John Reynolds' Idea (1983)

If we have a single implementation that works for all types, then it must be uniform in some way. We can capture uniformity through *preservation of relations*.

Preservation of Relations

$$M : \forall\alpha.\ \mathsf{List}\ \alpha \to \mathsf{List}\ \alpha$$

Assume two types X and Y and a relation $R : X \leftrightarrow Y$

let $l_X = [x_1, x_2, x_3]$ be a list of Xs

John Reynolds' Idea (1983)

If we have a single implementation that works for all types, then it must be uniform in some way. We can capture uniformity through *preservation of relations*.

Preservation of Relations

$$M : \forall \alpha.\ \mathsf{List}\ \alpha \to \mathsf{List}\ \alpha$$

Assume two types X and Y and a relation $R : X \leftrightarrow Y$

let $l_X = [x_1, x_2, x_3]$ be a list of Xs

let $l_Y = [y_1, y_2, y_3]$ be a list of Ys

John Reynolds' Idea (1983)

If we have a single implementation that works for all types, then it must be uniform in some way. We can capture uniformity through *preservation of relations*.

Preservation of Relations

$$M : \forall\alpha.\ \mathsf{List}\ \alpha \to \mathsf{List}\ \alpha$$

Assume two types X and Y and a relation $R : X \leftrightarrow Y$

let $l_X = [x_1, x_2, x_3]$ be a list of Xs

let $l_Y = [y_1, y_2, y_3]$ be a list of Ys

such that $x_1 R y_1$, $x_2 R y_2$, $x_3 R y_3$

John Reynolds' Idea (1983)

If we have a single implementation that works for all types, then it must be uniform in some way. We can capture uniformity through *preservation of relations.*

Preservation of Relations

$$M : \forall \alpha.\ \mathsf{List}\ \alpha \rightarrow \mathsf{List}\ \alpha$$

Assume two types X and Y and a relation $R : X \leftrightarrow Y$

let $l_X = [x_1, x_2, x_3]$ be a list of Xs

let $l_Y = [y_1, y_2, y_3]$ be a list of Ys

such that $x_1 R y_1$, $x_2 R y_2$, $x_3 R y_3$

then:

$M\ l_X = [x'_1, ..., x'_n]$, $M\ l_Y = [y'_1, ..., y'_n]$ and for all i, $x'_i R y'_i$

From Types to Relations

If A is a type with free variables $\alpha_1, \ldots, \alpha_n$
and we have types $X_1, \ldots, X_n$, and $Y_1, \ldots, Y_n$
and relations $R_1 : X_1 \leftrightarrow Y_1, \ldots, R_n : X_n \leftrightarrow Y_n$

then $\lfloor A \rfloor : (A[X_1/\alpha_1, \ldots, X_n/\alpha_n]) \leftarrow (A[Y_1/\alpha_1, \ldots, Y_n/\alpha_n])$

From Types to Relations

If A is a type with free variables $\alpha_1, \ldots, \alpha_n$
and we have types $X_1, \ldots, X_n$, and $Y_1, \ldots, Y_n$
and relations $R_1 : X_1 \leftrightarrow Y_1, \ldots, R_n : X_n \leftrightarrow Y_n$
then $\lfloor A \rfloor : (A[X_1/\alpha_1, \ldots, X_n/\alpha_n]) \leftarrow (A[Y_1/\alpha_1, \ldots, Y_n/\alpha_n])$

Definition

$$
\begin{array}{lcl}
\lfloor \alpha_i \rfloor & = & R_i \\
\lfloor \forall \alpha . A \rfloor & = & \{(x, x') \mid \forall X, Y, R : X \leftrightarrow Y.\ x \lfloor A \rfloor x'\} \\
\lfloor A \to B \rfloor & = & \{(f, g) \mid \forall (x, x') \in \lfloor A \rfloor .\ (f\,x, g\,x') \in \lfloor B \rfloor\} \\
\lfloor \mathsf{List}\ A \rfloor & = & \{(l, l') \mid |l| = |l'| \text{ and for all } i,\ l_i \lfloor A \rfloor l'_i\}
\end{array}
$$

Reynolds' Abstraction Theorem

If M has type A, then $M \lfloor A \rfloor M$.

Reynolds' Abstraction Theorem

If M has type A, then $M\lfloor A\rfloor M$.

Caveats

Depends heavily on the language!

general recursion	: (some) relations must be *admissible*
effects	: (some) relations must be $\top\top$-closed
seq	: (some) relations must be *bottom-reflecting*
typecase	: things get weird

$$M : \forall\alpha.\ \mathsf{List}\ \alpha \to \mathsf{List}\ \alpha$$

and we have types X, Y and a function $f : X \to Y$

$$M : \forall\alpha.\ \mathsf{List}\ \alpha \to \mathsf{List}\ \alpha$$

and we have types X, Y and a function $f : X \to Y$

By Reynolds' Abstraction Theorem

$M \lfloor \forall\alpha.\ \mathsf{List}\ \alpha \to \mathsf{List}\ \alpha \rfloor M$

$$M : \forall\alpha.\ \mathsf{List}\ \alpha \to \mathsf{List}\ \alpha$$

and we have types X, Y and a function $f : X \to Y$

By Reynolds' Abstraction Theorem

$M \lfloor \forall\alpha.\ \mathsf{List}\ \alpha \to \mathsf{List}\ \alpha \rfloor M$

Which means

For all types X, Y and relations $R : X \leftrightarrow Y$,

for all l_X and l_Y such that $l_X \lfloor \mathsf{List}\ \alpha \rfloor l_Y$,

$(M\ l_X) \lfloor \mathsf{List}\ \alpha \rfloor (M\ l_Y)$

$$M : \forall\alpha.\ \mathsf{List}\ \alpha \to \mathsf{List}\ \alpha$$

and we have types X, Y and a function $f : X \to Y$

By Reynolds' Abstraction Theorem

$M \lfloor \forall\alpha.\ \mathsf{List}\ \alpha \to \mathsf{List}\ \alpha \rfloor M$

Which means

For all types X, Y and relations $R : X \leftrightarrow Y$,

for all l_X and l_Y such that $l_X \lfloor \mathsf{List}\ \alpha \rfloor l_Y$,

$(M\ l_X) \lfloor \mathsf{List}\ \alpha \rfloor (M\ l_Y)$

Instantiating

Set $R = \{(x, y) \mid fx = y\}$

then $l_X \lfloor \mathsf{List}\ \alpha \rfloor l_Y \Leftrightarrow l_Y = \mathrm{map}\ f\ l_X$

so for all l, $M\ (\mathrm{map}\ f\ l) = \mathrm{map}\ f\ (M\ l)$

Representing Datatypes

Representing Lists

$$\forall \alpha.\, \alpha \to (A \to \alpha \to \alpha) \to \alpha$$
$$\cong$$
$$\mathsf{List}(A)$$

Representing Lists

$$\forall\alpha.\ \alpha \to (A \to \alpha \to \alpha) \to \alpha$$
$$\cong$$
$$\mathsf{List}(A)$$

$$\lambda\text{nil}.\ \lambda\text{cons}.\ \text{cons}\ a_1\ (\text{cons}\ a_2\ \text{nil}) \approx [a_1, a_2]$$

Representing Lists

$$\forall\alpha.\ \alpha \to (A \to \alpha \to \alpha) \to \alpha$$
$$\cong$$
$$\mathsf{List}(A)$$

$$\lambda\text{nil}.\ \lambda\text{cons}.\ \text{cons}\ a_1\ (\text{cons}\ a_2\ \text{nil}) \approx [a_1, a_2]$$

Representing Syntax

$$\forall\alpha.\ ((\alpha \to \alpha) \to \alpha) \to (\alpha \to \alpha \to \alpha) \to \alpha$$
$$\cong$$

Terms of the untyped λ-calculus with no free type variables

Representing Lists

$$\forall\alpha.\ \alpha \to (A \to \alpha \to \alpha) \to \alpha$$
$$\cong$$
$$\mathsf{List}(A)$$

$$\lambda\text{nil}.\ \lambda\text{cons}.\ \text{cons}\ a_1\ (\text{cons}\ a_2\ \text{nil}) \approx [a_1, a_2]$$

Representing Syntax

$$\forall\alpha.\ ((\alpha \to \alpha) \to \alpha) \to (\alpha \to \alpha \to \alpha) \to \alpha$$
$$\cong$$
Terms of the untyped λ-calculus with no free type variables

$$\lambda\text{lam}.\ \lambda\text{app}.\ \text{lam}\ (\lambda x.\ \text{lam}\ (\lambda y.\ \text{app}\ x\ y)) \approx \lambda xy.xy$$

Beyond Type Quantification

For any function *area* with the type:

$$area : \forall B{:}\mathsf{GL}_2, t{:}\mathsf{T}_2. \quad \texttt{vec}\langle B, t\rangle \to \texttt{vec}\langle B, t\rangle \to \texttt{vec}\langle B, t\rangle \to \texttt{real}\langle |\det B| \rangle$$

For any function *area* with the type:

$$area : \forall B{:}\mathsf{GL}_2, t{:}\mathsf{T}_2.\\ \quad \texttt{vec}\langle B, t\rangle \to \texttt{vec}\langle B, t\rangle \to \texttt{vec}\langle B, t\rangle \to \texttt{real}\langle |\det B| \rangle$$

Deduce translation invariance:

$$\forall \vec{t}, \vec{v_1}, \vec{v_2}, \vec{v_3}.\ area\ (v_1 + t)\ (v_2 + t)\ (v_3 + t) = area\ v_1\ v_2\ v_3$$

For any function *area* with the type:

$$area : \forall B{:}\mathsf{GL}_2, t{:}\mathsf{T}_2.\quad \texttt{vec}\langle B, t\rangle \to \texttt{vec}\langle B, t\rangle \to \texttt{vec}\langle B, t\rangle \to \texttt{real}\langle|\det B|\rangle$$

Deduce translation invariance:

$$\forall \vec{t}, \vec{v_1}, \vec{v_2}, \vec{v_3}.\ area\ (v_1 + t)\ (v_2 + t)\ (v_3 + t) = area\ v_1\ v_2\ v_3$$

Deduce orthogonal transformation invariance:

$$\forall O, \vec{v_1}, \vec{v_2}, \vec{v_3}.\ area\ (Ov_1)\ (Ov_2)\ (Ov_3) = area\ v_1\ v_2\ v_3$$

For any function *area* with the type:

$$area : \forall B{:}\mathsf{GL}_2, t{:}\mathsf{T}_2. \quad \texttt{vec}\langle B, t\rangle \to \texttt{vec}\langle B, t\rangle \to \texttt{vec}\langle B, t\rangle \to \texttt{real}\langle |\det B| \rangle$$

Deduce translation invariance:

$$\forall \vec{t}, \vec{v_1}, \vec{v_2}, \vec{v_3}.\ area\ (v_1 + t)\ (v_2 + t)\ (v_3 + t) = area\ v_1\ v_2\ v_3$$

Deduce orthogonal transformation invariance:

$$\forall O, \vec{v_1}, \vec{v_2}, \vec{v_3}.\ area\ (Ov_1)\ (Ov_2)\ (Ov_3) = area\ v_1\ v_2\ v_3$$

Deduce scaling variance:

$$\forall s, \vec{v_1}, \vec{v_2}, \vec{v_3}.\ area\ (s \cdot v_1)\ (s \cdot v_2)\ (s \cdot v_3) = s^2 \cdot area\ v_1\ v_2\ v_3$$

Types to Relations

$$
\begin{aligned}
\lfloor \forall B : \mathsf{GL}_2.A \rfloor &= \{(x, x') \mid \forall B : \mathrm{GL}_2.\ x\lfloor A \rfloor x'\} \\
\lfloor \forall t : \mathsf{T}_2.A \rfloor &= \{(x, x') \mid \forall t : \mathrm{T}_2.\ x\lfloor A \rfloor x'\} \\
\lfloor \forall s : \mathsf{GL}_1.A \rfloor &= \{(x, x') \mid \forall s : \mathrm{GL}_1.\ x\lfloor A \rfloor x'\} \\
\lfloor \mathtt{vec}\langle B, t\rangle \rfloor &= \{(\vec{v_1}, \vec{v_2}) \mid \vec{v_2} = B\vec{v_1} + t\} \\
\lfloor \mathtt{real}\langle s\rangle \rfloor &= \{(x_1, x_2) \mid x_2 = s x_1\}
\end{aligned}
$$

$$\textit{area} : \forall B{:}\mathsf{GL}_2, t{:}\mathsf{T}_2.$$
$$\qquad \texttt{vec}\langle B, t\rangle \to \texttt{vec}\langle B, t\rangle \to \texttt{vec}\langle B, t\rangle \to \texttt{real}\langle |\det B|\rangle$$

$$area : \forall B{:}\mathsf{GL}_2, t{:}\mathsf{T}_2.\quad \texttt{vec}\langle B, t\rangle \to \texttt{vec}\langle B, t\rangle \to \texttt{vec}\langle B, t\rangle \to \texttt{real}\langle |\det B| \rangle$$

Symmetry as a Guide:

$$area : \forall B{:}\mathsf{GL}_2, t{:}\mathsf{T}_2.\; \texttt{vec}\langle B, t\rangle \to \texttt{vec}\langle B, t\rangle \to \texttt{vec}\langle B, t\rangle \to \texttt{real}\langle |\det B| \rangle$$

Symmetry as a Guide:

We have $p_1 : \texttt{vec}\langle B, t\rangle$, $p_2 : \texttt{vec}\langle B, t\rangle$, $p_3 : \texttt{vec}\langle B, t\rangle$

$$area : \forall B{:}\mathsf{GL}_2, t{:}\mathsf{T}_2.$$
$$\texttt{vec}\langle B, t\rangle \to \texttt{vec}\langle B, t\rangle \to \texttt{vec}\langle B, t\rangle \to \texttt{real}\langle|\det B|\rangle$$

Symmetry as a Guide:

We have $p_1 : \texttt{vec}\langle B, t\rangle$, $p_2 : \texttt{vec}\langle B, t\rangle$, $p_3 : \texttt{vec}\langle B, t\rangle$

goal is $\texttt{real}\langle|\det B|\rangle$

$$area : \forall B{:}\mathsf{GL}_2,\, t{:}\mathsf{T}_2.$$
$$\texttt{vec}\langle B, t\rangle \to \texttt{vec}\langle B, t\rangle \to \texttt{vec}\langle B, t\rangle \to \texttt{real}\langle |\det B|\rangle$$

Symmetry as a Guide:

We have $p_1 : \texttt{vec}\langle B, t\rangle$, $p_2 : \texttt{vec}\langle B, t\rangle$, $p_3 : \texttt{vec}\langle B, t\rangle$

goal is $\texttt{real}\langle |\det B|\rangle$

select one to be the "origin":

$(p_2 - p_1) : \texttt{vec}\langle B, 0\rangle$ and $(p_3 - p_1) : \texttt{vec}\langle B, 0\rangle$

$$area : \forall B{:}\mathsf{GL}_2,\ t{:}\mathsf{T}_2.\quad \texttt{vec}\langle B, t\rangle \to \texttt{vec}\langle B, t\rangle \to \texttt{vec}\langle B, t\rangle \to \texttt{real}\langle |\det B|\rangle$$

Symmetry as a Guide:

We have $p_1 : \texttt{vec}\langle B, t\rangle$, $p_2 : \texttt{vec}\langle B, t\rangle$, $p_3 : \texttt{vec}\langle B, t\rangle$

goal is $\texttt{real}\langle |\det B|\rangle$

select one to be the "origin":

$(p_2 - p_1) : \texttt{vec}\langle B, 0\rangle$ and $(p_3 - p_1) : \texttt{vec}\langle B, 0\rangle$

remove rotational symmetry and get area of parallelogram:

$(p_2 - p_1) \times (p_3 - p_1) : \texttt{real}\langle \det B\rangle$

$$area : \forall B{:}\mathsf{GL}_2, t{:}\mathsf{T}_2. \quad \texttt{vec}\langle B, t\rangle \to \texttt{vec}\langle B, t\rangle \to \texttt{vec}\langle B, t\rangle \to \texttt{real}\langle |\det B|\rangle$$

Symmetry as a Guide:

We have $p_1 : \texttt{vec}\langle B, t\rangle$, $p_2 : \texttt{vec}\langle B, t\rangle$, $p_3 : \texttt{vec}\langle B, t\rangle$

goal is $\texttt{real}\langle |\det B|\rangle$

select one to be the "origin":

$(p_2 - p_1) : \texttt{vec}\langle B, 0\rangle$ and $(p_3 - p_1) : \texttt{vec}\langle B, 0\rangle$

remove rotational symmetry and get area of parallelogram:

$(p_2 - p_1) \times (p_3 - p_1) : \texttt{real}\langle \det B\rangle$

remove reflectional symmetry:

$|(p_2 - p_1) \times (p_3 - p_1)| : \texttt{real}\langle |\det B|\rangle$

$$area : \forall B{:}\mathsf{GL}_2, t{:}\mathsf{T}_2.$$
$$\texttt{vec}\langle B, t\rangle \to \texttt{vec}\langle B, t\rangle \to \texttt{vec}\langle B, t\rangle \to \texttt{real}\langle|\det B|\rangle$$

Symmetry as a Guide:

We have $p_1 : \texttt{vec}\langle B, t\rangle$, $p_2 : \texttt{vec}\langle B, t\rangle$, $p_3 : \texttt{vec}\langle B, t\rangle$

goal is $\texttt{real}\langle|\det B|\rangle$

select one to be the "origin":

$(p_2 - p_1) : \texttt{vec}\langle B, 0\rangle$ and $(p_3 - p_1) : \texttt{vec}\langle B, 0\rangle$

remove rotational symmetry and get area of parallelogram:

$(p_2 - p_1) \times (p_3 - p_1) : \texttt{real}\langle\det B\rangle$

remove reflectional symmetry:

$|(p_2 - p_1) \times (p_3 - p_1)| : \texttt{real}\langle|\det B|\rangle$

halve:

$\frac{1}{2} * |(p_2 - p_1) \times (p_3 - p_1)| : \texttt{real}\langle|\det B|\rangle$

$area : \forall B{:}\mathsf{GL}_2, t{:}\mathsf{T}_2.$

$$\texttt{vec}\langle B, t\rangle \to \texttt{vec}\langle B, t\rangle \to \texttt{vec}\langle B, t\rangle \to \texttt{real}\langle |\det B|\rangle$$

Symmetry as a Guide:

We have $p_1 : \texttt{vec}\langle B, t\rangle$, $p_2 : \texttt{vec}\langle B, t\rangle$, $p_3 : \texttt{vec}\langle B, t\rangle$

goal is $\texttt{real}\langle |\det B|\rangle$

select one to be the "origin":

$(p_2 - p_1) : \texttt{vec}\langle B, 0\rangle$ and $(p_3 - p_1) : \texttt{vec}\langle B, 0\rangle$

remove rotational symmetry and get area of parallelogram:

$(p_2 - p_1) \times (p_3 - p_1) : \texttt{real}\langle \det B\rangle$

remove reflectional symmetry:

$|(p_2 - p_1) \times (p_3 - p_1)| : \texttt{real}\langle |\det B|\rangle$

halve:

$\frac{1}{2} * |(p_2 - p_1) \times (p_3 - p_1)| : \texttt{real}\langle |\det B|\rangle$

The Message

Polymorphism means Uniformity

If a type has a $\forall$, then implementations are uniform under change:

$\forall \alpha.$	: uniform under change of data representation
$\forall B : \mathsf{GL}_2.$	: uniform under change of basis
$\forall t : \mathsf{T}_2.$	: uniform under change of origin
$\forall s : \mathsf{GL}_1.$	: uniform under change of scale

Uniformity allows one to deduce “free theorems”

Polymorphism means Uniformity

If a type has a ∀, then implementations are uniform under change:

$\forall \alpha.$	: uniform under change of data representation
$\forall B : \mathsf{GL}_2.$	: uniform under change of basis
$\forall t : \mathsf{T}_2.$	: uniform under change of origin
$\forall s : \mathsf{GL}_1.$	: uniform under change of scale

Uniformity allows one to deduce "free theorems"

What other sorts of uniformity are useful?