Theorems for free! (Philip Wadler)

Afonso Arriaga

HASLab - INESC TEC, University of Minho

June 22, 2012

1. Introduction
2. Basic concepts
3. Examples of theorems from types
4. Generalization of theorems for free
5. Conclusions

Introduction

"Based on the concept of relational parametricity (Reynolds 1983), Wadler (1989) established so-called 'free theorems', a method for obtaining proofs of program properties from parametrically polymorphic types in purely functional languages."

(Daniel Seidel and Janis Voigtlander, 2011)

Basic concepts

What is a parametrically polymorphic function?

It's a function whose type signature allows various arguments to take on arbitrary types, but the types must be related to each other in some way.

Basic concepts

Why parametric polymorphism?

Because...

- it allows code to be reused.
- we can derive **useful** theorems.
- type-agnostic reasoning is better!

Example 1 - Reverse

Let r be a function of type:
$r :: \mathit{forall}\ x.\ [x] \rightarrow [x]$

Take any total function $f :: a \rightarrow b$ for concrete types a and b.
For example, suppose $f = \mathit{ord}$.
$\mathit{ord} :: \mathit{Char} \rightarrow \mathit{Int}$
$f = \mathit{ord}$

It is possible to conclude that:
$\mathit{map}\ f\ .\ r_{\mathit{char}} \equiv r_{\mathit{int}}\ .\ \mathit{map}\ f$

Example 2 - Head

Let h be a function of type:
$h :: forall\ x.\ [x] \rightarrow x$

Take any total function $f :: a \rightarrow b$ for concrete types a and b.

It is possible to conclude that:
$map\ f\ .\ h_a \equiv h_b\ .\ map\ f$

Example 3 - Zip

Let z be a function of type:
$z :: \mathit{forall}\ x, y.\ [x] \to [y] \to [(x, y)]$

Take any total function $f :: a \to a'$ and $g :: b \to b'$ for concrete types a, a', b and b'.

Commutative diagram:

$$\begin{tikzcd}
{[AxB]} \arrow[d, "map(f\ x\ g)"] & {[A]x[B]} \arrow[l, "zip_{AB}"'] \arrow[d, "(map\ f)\ x\ (map\ g)"] \\
{[A'xB']} & {[A']x[B']} \arrow[l, "zip_{A'B'}"']
\end{tikzcd}$$

Generalization of theorems for free

- **F** and **G** are functors
- ϕ is a natural transformation from **F** to **G**.

$$\begin{array}{ccc}
\mathbf{G}\ A & \xleftarrow{\ \phi\ } & \mathbf{F}\ A \\
\downarrow{\scriptstyle \mathbf{G}\ f} & & \downarrow{\scriptstyle \mathbf{F}\ f} \\
\mathbf{G}\ B & \xleftarrow{\ \phi\ } & \mathbf{F}\ B
\end{array}$$

Conclusions

For every function $f :: \mathbf{F}a \rightarrow \mathbf{F}a$ and for every choice of $g :: x \rightarrow y$, with x and y concrete types, it holds that:

$$f_y.\ Fmap\ g \equiv Fmap\ g.\ f_x$$

Reading types as relations is the key to derive other theorems from types, such as for higher-order functions **sort** and **fold**.

Automatic generation of free theorems

- http://www-ps.iai.uni-bonn.de/cgi-bin/free-theorems-webui.cgi

Thank you!