“Theorems for free”: a (calculational) introduction

J.N. Oliveira

Dept. Informática,
Universidade do Minho
Braga, Portugal

2003 (last update: 2013)

Parametric polymorphism by example

Function

$$countBits : \mathbb{N}_0 \leftarrow Bool^\star$$
$$countBits\ [\,] = 0$$
$$countBits(b{:}bs) = 1 + countBits\ bs$$

and

$$countNats : \mathbb{N}_0 \leftarrow \mathbb{N}^\star$$
$$countNats\ [\,] = 0$$
$$countNats(b{:}bs) = 1 + countNats\ bs$$

are both subsumed by **generic** (parametric):

$$count : (\forall a)\ \mathbb{N}_0 \leftarrow a^\star$$
$$count\ [\,] = 0$$
$$count(a{:}as) = 1 + count\ as$$

Parametric polymorphism: why?

- Less code (**specific** solution = **generic** solution + **customization**)
- Intellectual reward
- Last but not least, quotation from *Theorems for free!*, by Philip Wadler [4]:

  > *From the type of a polymorphic function we can derive a theorem that it satisfies.* (...) *How useful are the theorems so generated? Only time and experience will tell* (...)

- No doubt: free theorems are **very** useful!

Polymorphic type signatures

Polymorphic function signature:

$$f \quad : \quad t$$

where t is a functional type, according to the following "grammar" of types:

$$
\begin{array}{lll}
t & ::= & t' \leftarrow t'' \\
t & ::= & \mathsf{F}(t_1, \ldots, t_n) \qquad \text{type constructor F} \\
t & ::= & v \qquad \text{type } \textit{variables } v\text{, cf. } \textit{polymorphism}
\end{array}
$$

What does it mean for f to be **parametrically** polymorphic?

Free theorem of type t

Let

- V be the set of type variables involved in type t
- $\{R_v\}_{v \in V}$ be a V-indexed family of relations (f_v in case all such R_v are functions).
- R_t be a relation defined inductively as follows:

$$R_{t:=v} = R_v \tag{182}$$

$$R_{t:=\mathsf{F}(t_1,\ldots,t_n)} = \mathsf{F}(R_{t_1}, \ldots, R_{t_n}) \tag{183}$$

$$R_{t:=t' \leftarrow t''} = R_{t'} \leftarrow R_{t''} \tag{184}$$

Questions: What does F in the RHS of (183) mean? What kind of relation is $R_{t'} \leftarrow R_{t''}$? See next slides.

Background: relators

Parametric datatype G is said to be a **relator** [2] wherever, given a relation from A to B, $\mathsf{G}\, R$ extends R to G-structures: it is a relation

$$\begin{tikzcd} A \arrow[r, dotted, no head] \arrow[d, "R"'] & \mathsf{G}\, A \arrow[d, "\mathsf{G}\, R"] \\ B \arrow[r, dotted, no head] & \mathsf{G}\, B \end{tikzcd} \tag{185}$$

from $\mathsf{G}\, A$ to $\mathsf{G}\, B$ which obeys the following properties:

$$\mathsf{G}\, id = id \tag{186}$$

$$\mathsf{G}\,(R \cdot S) = (\mathsf{G}\, R) \cdot (\mathsf{G}\, S) \tag{187}$$

$$\mathsf{G}\,(R^\circ) = (\mathsf{G}\, R)^\circ \tag{188}$$

and is monotonic:

$$R \subseteq S \quad \Rightarrow \quad \mathsf{G}\, R \subseteq \mathsf{G}\, S \tag{189}$$

Relators: "*Maybe*" example

$$\begin{tikzcd} A \arrow[d, "R"'] \arrow[r, dotted, no head] & \mathsf{G}\,A = 1 + A \arrow[d, "\mathsf{G}\,R = id + R"] \\ B \arrow[r, dotted, no head] & \mathsf{G}\,B = 1 + B \end{tikzcd}$$

(Read $1 + A$ as "maybe A")

Unfolding $\mathsf{G}\,R = id + R$:

$$y(id + R)x$$

$\Leftrightarrow$ { unfolding the sum, cf. $id + R = [i_1 \cdot id\ , i_2 \cdot R]$ }

$$y(i_1 \cdot i_1^\circ \cup i_2 \cdot R \cdot i_2^\circ)x$$

$\Leftrightarrow$ { relational union (68); image }

$$y(\mathrm{img}\, i_1)x \vee y(i_2 \cdot R \cdot i_2^\circ)x$$

$\Leftrightarrow$ { let *NIL* be <u>the</u> inhabitant of the singleton type }

$$y = x = i_1 NIL \vee \langle \exists\ b, a\ :\ y = i_2\ b \wedge x = i_2\ a :\ b\ R\ a \rangle$$

Relators: example

Take $\mathsf{F}X = X^\star$.

Then, for some $B \xleftarrow{R} A$, relator $B^\star \xleftarrow{R^\star} A^\star$ is the relation

$$
\begin{aligned}
s'(R^\star)s \quad \Leftrightarrow \quad & \mathit{inds}\ s' = \mathit{inds}\ s \wedge \\
& \langle \forall\ i\ :\ i \in \mathit{inds}\ s :\ (s\ i)R(s'\ i) \rangle
\end{aligned}
\tag{190}
$$

Exercise 79: Check properties (186) and (188) for the list relator defined above.

□

Background: "Reynolds arrow" operator

Define

$$f(R \leftarrow S)g \quad \Leftrightarrow \quad f \cdot S \subseteq R \cdot g \qquad \begin{tikzcd} A & B \arrow[l, "S"'] \arrow[d, "g"] \\ C \arrow[from=u, "f"'] & D \arrow[l, "R"] \end{tikzcd} \tag{191}$$

That is to say,

$$\frac{A \xleftarrow{S} B \qquad C \xleftarrow{R} D}{C^A \xleftarrow{R \leftarrow S} D^B}$$

For instance, $f(id \leftarrow id)g \quad \Leftrightarrow \quad f = g$ that is, $id \leftarrow id \quad = \quad id$

Free theorem (FT) of type t

The *free theorem* (FT) of type t is the following (remarkable) result due to J. Reynolds [3], advertised by P. Wadler [4] and re-written by Backhouse [1] in the pointfree style:

> *Given any function $\theta : t$, and V as above, then $\theta \; R_t \; \theta$ holds, for any relational instantiation of type variables in V.*

Note that this theorem

- is a result about t
- holds **independently** of the actual definition of θ.
- holds about any polymorphic function of type t

First example (id)

The target function:

$$\theta = id : a \leftarrow a$$

Calculation of $R_{t=a\leftarrow a}$:

$$\begin{array}{cl} & R_{a\leftarrow a} \\ \Leftrightarrow & \quad \{ \text{ rule } \quad R_{t=t'\leftarrow t''} \quad = \quad R_{t'} \leftarrow R_{t''} \quad \} \\ & R_a \leftarrow R_a \end{array}$$

Calculation of FT (R_a abbreviated to R):

$$\begin{array}{cl} & id(R \leftarrow R)id \\ \Leftrightarrow & \quad \{ \ (191) \ \} \\ & id \cdot R \subseteq R \cdot id \end{array}$$

First example (id)

In case R is a function f, the FT theorem boils down to id's **natural** property:

$$id \cdot f \;=\; f \cdot id$$

cf.

$$\begin{tikzcd} a \arrow[d, "f"'] & a \arrow[l, "id"'] \arrow[d, "f"] \\ b & b \arrow[l, "id"] \end{tikzcd}$$

which can be read alternatively as stating that id is the **unit** of composition.

Second example (*invl*)

The target function: $\theta = invl : a^\star \leftarrow a^\star$.

Calculation of $R_{t=a^\star \leftarrow a^\star}$:

$$
\begin{array}{ll}
 & R_{a^\star \leftarrow a^\star} \\
\Leftrightarrow & \qquad \{ \text{ rule } \; R_{t=t' \leftarrow t''} \; = \; R_{t'} \leftarrow R_{t''} \quad \} \\
 & R_{a^\star} \leftarrow R_{a^\star} \\
\Leftrightarrow & \qquad \{ \text{ rule } \; R_{t=\mathsf{F}(t_1,\ldots,t_n)} \; = \; \mathsf{F}(R_{t_1},\ldots,R_{t_n}) \quad \} \\
 & {R_a}^\star \leftarrow {R_a}^\star
\end{array}
$$

where $s \; R^\star s'$ is given by (190). The calculation of FT follows.

Second example (*invl*)

The FT itself will predict (R_a abbreviated to R):

$$invl(R^\star \leftarrow R^\star)invl$$

$$\Leftrightarrow \quad \{ \text{ definition } \ f(R \leftarrow S)g \ \Leftrightarrow \ f \cdot S \subseteq R \cdot g \ \}$$

$$invl \cdot R^\star \subseteq R^\star \cdot invl$$

In case R is a function r, the FT theorem boils down to $invl$'s **natural** property:

$$invl \cdot r^\star \ = \ r^\star \cdot invl$$

that is,

$$invl \ [\ r \ a \mid a \leftarrow l\ ] \ = \ [\ r \ b \mid b \leftarrow invl \ l\ ]$$

Second example (*invl*)

Further calculation (back to R):

$$
\begin{array}{ll}
 & invl \cdot R^{\star} \subseteq R^{\star} \cdot invl \\
\Leftrightarrow & \quad \{ \text{ shunting rule (54) } \} \\
 & R^{\star} \subseteq invl^{\circ} \cdot R^{\star} \cdot invl \\
\Leftrightarrow & \quad \{ \text{ going pointwise (39, 47) } \} \\
 & \langle \forall\ s, r \ :: \ s\ R^{\star} r \Rightarrow (invl\ \ s) R^{\star} (invl\ \ r) \rangle
\end{array}
$$

An instance of this pointwise version of *invl*-FT will state that, for example, *invl* will respect element-wise orderings ($R :=<$):

Second example (*invl*)

$$length\ s = length\ r \wedge \langle \forall\ i\ :\ i \in inds\ s :\ (s\ i) < (r\ i) \rangle$$
$$\Downarrow$$
$$length(invl\ s) = length(inv\ r)$$
$$\wedge$$
$$\langle \forall\ j\ :\ j \in inds\ s :\ (invl\ s)j < (invl\ r)j \rangle$$

(Guess other instances.)

Third example: FT of *sort*

Our next example calculates the FT of

$$sort : a^\star \leftarrow a^\star \leftarrow (Bool \leftarrow (a \times a))$$

where the first parameter stands for the chosen ordering relation, expressed by a binary predicate:

$$\begin{array}{cl}
& sort(R_{(a^\star \leftarrow a^\star) \leftarrow (Bool \leftarrow (a \times a))})sort \\
\Leftrightarrow & \qquad \{ \ (183,\ 182,\ 184);\ \text{abbreviate } R_a := R \ \} \\
& sort((R^\star \leftarrow R^\star) \leftarrow (R_{Bool} \leftarrow (R \times R)))sort \\
\Leftrightarrow & \qquad \{ \ R_{t:=Bool} = id \ \text{(constant relator) — cf. exercise 90} \ \} \\
& sort((R^\star \leftarrow R^\star) \leftarrow (id \leftarrow (R \times R)))sort
\end{array}$$

Third example: FT of *sort*

$$sort((R^\star \leftarrow R^\star) \leftarrow (id \leftarrow (R \times R)))sort$$

$\Leftrightarrow$ { (191) }

$$sort \cdot (id \leftarrow (R \times R)) \quad \subseteq \quad (R^\star \leftarrow R^\star) \cdot sort$$

$\Leftrightarrow$ { shunting (54) }

$$(id \leftarrow (R \times R)) \quad \subseteq \quad sort^\circ \cdot (R^\star \leftarrow R^\star) \cdot sort$$

$\Leftrightarrow$ { introduce variables f and g (39, 47) }

$$f(id \leftarrow (R \times R))g \quad \Rightarrow \quad (sort\ f)(R^\star \leftarrow R^\star)(sort\ g)$$

$\Leftrightarrow$ { (191) twice }

$$f \cdot (R \times R) \subseteq g \quad \Rightarrow \quad (sort\ f) \cdot R^\star \subseteq R^\star \cdot (sort\ g)$$

Third example: FT of *sort*

Case $R := r$:

$$f \cdot (r \times r) = g \quad \Rightarrow \quad (sort\ f) \cdot r^\star = r^\star \cdot (sort\ g)$$

$$\Leftrightarrow \qquad \{ \text{ introduce variables } \}$$

$$\left\langle \begin{array}{c} \forall\ a, b\ :: \\ f(r\ a, r\ b) = g(a, b) \end{array} \right\rangle \Rightarrow \left\langle \begin{array}{c} \forall\ l\ :: \\ (sort\ f)(r^\star\ l) = r^\star(sort\ g\ l) \end{array} \right\rangle$$

Denoting predicates f, g by infix orderings $\leq, \preceq$:

$$\left\langle \begin{array}{c} \forall\ a, b\ :: \\ r\ a \leq r\ b \Leftrightarrow a \preceq b \end{array} \right\rangle \Rightarrow \left\langle \begin{array}{c} \forall\ l\ :: \\ sort\ (\leq)(r^\star\ l) = r^\star(sort\ (\preceq)\ l) \end{array} \right\rangle$$

That is, for r monotonic and injective,

$$sort\ (\leq)\ [\ r\ a \mid a \leftarrow l\ ]$$

is always the same list a

$$[\ r\ a \mid a \leftarrow sort\ (\preceq)\ l\ ]$$

Exercises

Exercise 80: Let C be a nonempty data domain and let and $c \in C$. Let $\underline{c}$ be the *"everywhere c"* function:

$$\begin{array}{rcl} \underline{c} & : & A \longrightarrow C \\ \underline{c}\, a & \triangleq & c \end{array} \qquad (192)$$

Show that the free theorem of $\underline{c}$ reduces to

$$\langle \forall\ R\ ::\ R \subseteq \top \rangle \qquad (193)$$

□

Exercise 81: Calculate the free theorem associated with the projections $A \xleftarrow{\pi_1} A \times B \xrightarrow{\pi_2} B$ and instantiate it to (a) functions; (b) coreflexives. Introduce variables and derive the corresponding pointwise expressions.

□

Exercises

Exercise 82: Consider higher order function `const: a -> b -> a` such that, given any x of type a, produces the constant function $const\ x$. Show that the equalities

$$
\begin{aligned}
const(f\ x) &= f \cdot (const\ x) && (194)\\
(const\ x) \cdot f &= const\ x && (195)\\
(const\ x)^{\circ} \cdot (const\ x) &= \top && (196)
\end{aligned}
$$

arise as corollaries of the *free theorem* of $const$.

□

Exercises

Exercise 83: The following is a well-known Haskell function

```
filter :: forall a. (a -> Bool) -> [a] -> [a]
```

Calculate the free theorem associated with its type

$$filter : a^\star \leftarrow a^\star \leftarrow (Bool \leftarrow a)$$

and instantiate it to the case where all relations are functions.

□

Exercise 84: In many sorting problems, data are sorted according to a given *ranking* function which computes each datum's numeric rank (eg. students marks, credits, etc). In this context one may parameterize sorting with an extra parameter f ranking data into a fixed numeric datatype, eg. the integers: $serial : (a \to \mathbb{N}) \to a^\star \to a^\star$.
Calculate the FT of $serial$.

□

Exercises

Exercise 85: Consider the following function from Haskell's Prelude:

```
findIndices ::  (a -> Bool) -> [a] -> [Int]
findIndices p xs = [ i | (x,i) <- zip xs [0..], p
x ]
```

which yields the indices of elements in a sequence `xs` which satisfy `p`. For instance, $\mathit{findIndices}\ (< 0)\ [1, -2, 3, 0, -5] = [1, 4]$. Calculate the FT of this function.

□

Exercise 86: Choose arbitrary functions from Haskell's Prelude and calculate their FT.

□

Exercises

Exercise 87: Wherever two equally typed functions f, g such that $f\ a \leq g\ a$, for all a, we say that f is *pointwise at most* g and write $f \mathrel{\dot{\leq}} g$. In symbols:

$$f \mathrel{\dot{\leq}} g \quad \triangleq \quad f \subseteq (\leq) \cdot g \qquad \text{cf. diagram} \qquad \begin{tikzcd} & A \arrow[dl, "f"'] \arrow[dr, "g"] \arrow[d, phantom, "\subseteq"] & \\ B & & B \arrow[ll, "\leq"] \end{tikzcd} \tag{197}$$

Show that implication

$$f \mathrel{\dot{\leq}} g \quad \Rightarrow \quad (map\ f) \mathrel{\dot{\leq}^{\star}} (map\ g) \tag{198}$$

follows from the *FT* of the function $map : (a \rightarrow b) \rightarrow a^{\star} \rightarrow b^{\star}$.

$\square$

Automatic generation of free theorems (Haskell)

See the interesting site in Janis Voigtlaender's home page:

http://www-ps.iai.uni-bonn.de/ft

Relators in our calculational style are implemented in this automatic generator by structural *lifting*.

Exercise 88: Infer the FT of the following function, written in Haskell syntax,

```
while ::  (a -> Bool) -> (a -> a) -> (a -> b) -> a -> b
while p f g x = if not(p x) then g x else while p f g
(f x)
```

which implements a generic `while`-loop. Derive its corollary for functions and compare your result with that produced by the tool above.

□

Fourth example: FT of $(\!|\_|\!)$

Recall the catamorphism (fold) combinator:

$$\begin{tikzcd} \mathsf{F}\,a \arrow[d, "(\!|g|\!)"'] & \mathsf{B}\,(a, \mathsf{F}\,a) \arrow[l, "in_{\mathsf{F}\,a}"'] \arrow[d, "\mathsf{B}\,(id,(\!|g|\!))"] \\ b & \mathsf{B}\,(a, b) \arrow[l, "g"] \end{tikzcd}$$

So $(\!|\_|\!)$ has generic type

$$(\!|\_|\!) : b \leftarrow \mathsf{F}\,a \leftarrow (b \leftarrow \mathsf{B}\,(a, b))$$

where $\mathsf{F}\,a \cong \mathsf{B}\,(a, \mathsf{F}\,a)$. Then $(\!|\_|\!)$-FT is

$$(\!|\_|\!) \cdot (R_b \leftarrow \mathsf{B}\,(R_a, R_b)) \quad \subseteq \quad (R_b \leftarrow \mathsf{F}\,R_a) \cdot (\!|\_|\!)$$

Fourth example: FT of $(\!|\_|\!)$

This unfolds into (R_a, R_b abbreviated to R, S):

$$
\begin{array}{cl}
& (\!|\_|\!) \cdot (S \leftarrow \mathsf{B}\,(R, S)) \;\subseteq\; (S \leftarrow \mathsf{F}\,R) \cdot (\!|\_|\!) \\
\Leftrightarrow & \qquad \{ \text{ shunting (54) } \} \\
& (S \leftarrow \mathsf{B}\,(R, S)) \;\subseteq\; (\!|\_|\!)^{\circ}(S \leftarrow \mathsf{F}\,R) \cdot (\!|\_|\!) \\
\Leftrightarrow & \qquad \{ \text{ introduce variables } f \text{ and } g \text{ (39, 47) } \} \\
& f(S \leftarrow \mathsf{B}\,(R, S))g \;\Rightarrow\; (\!|f|\!)(S \leftarrow \mathsf{F}\,R)(\!|g|\!) \\
\Leftrightarrow & \qquad \{ \text{ definition } \; f(R \leftarrow S)g \;\Leftrightarrow\; f \cdot S \subseteq R \cdot g \; \} \\
& f \cdot \mathsf{B}\,(R, S) \subseteq S \cdot g \;\Rightarrow\; (\!|f|\!) \cdot \mathsf{F}\,R \subseteq S \cdot (\!|g|\!)
\end{array}
$$

$(\!|\_|\!)$-FT corollaries

From

$$f \cdot \mathsf{B}\,(R, S) \subseteq S \cdot g \quad \Rightarrow \quad (\!|f|\!) \cdot \mathsf{F}\,R \subseteq S \cdot (\!|g|\!) \tag{199}$$

we can infer:

- $(\!|\_|\!)$-*fusion* $(R, S := id, s)$:

$$f \cdot \mathsf{B}\,(id, s) = s \cdot g \quad \Rightarrow \quad (\!|f|\!) = s \cdot (\!|g|\!) \tag{200}$$

- $(\!|\_|\!)$-*absorption* $(R, S := r, id)$:

$$f \cdot \mathsf{B}\,(r, id) = g \quad \Rightarrow \quad (\!|f|\!) \cdot \mathsf{F}\,r = (\!|g|\!) \tag{201}$$

Substituting $g := f \cdot \mathsf{B}\,(r, id)$:

$$(\!|f|\!) \cdot \mathsf{F}\,r = (\!|f \cdot \mathsf{B}\,(r, id)|\!) \tag{202}$$

Exercises

Exercise 89: Let $iprod = (\![\underline{1}\ , (\times)]\!)$ be the function which multiplies all natural numbers in a given list; $even$ be the predicate which tests natural numbers for evenness; and $exists = (\![\underline{\textsc{False}}\ , (\vee)]\!)$.
From (199) infer

$$even \cdot iprod \quad = \quad exists \cdot even^{\star}$$

meaning that product $n_1 \times n_2 \times \ldots \times n_m$ is even iff some n_i is so.
$\square$

Exercises

Exercise 90: Show that the *identity* relator Id, which is such that $\mathsf{Id}\ R \ = \ R$ and the *constant* relator K (for a given data type K) which is such that $\mathsf{K}\ R \ = \ id_K$ are indeed relators.

$\square$

Exercise 91: Show that product

$$\begin{tikzcd}
A \arrow[d, "R"'] & C \arrow[d, "S"'] \arrow[r, dotted, no head] & \mathsf{G}(A, C) = A \times C \arrow[d, "\mathsf{G}(R,S) = R \times S"] \\
B & D \arrow[r, dotted, no head] & \mathsf{G}(B, D) = B \times D
\end{tikzcd}$$

is a (binary) relator.

$\square$

Last but not least

"Free contracts" in **DbC**:

- Many functional **contracts** arise naturally as corollaries of **free theorems**.
- This has the advantage of **saving** us **from proving** such contracts explicitly.
- The following exercises provide ample evidence of this.

Exercise 92: The type of functional composition $(\cdot)$ is

```
(.)  ::  (b -> c) -> (a -> b) -> a -> c
```

Show that **contract composition** (151) is a corollary of the free theorem (FT) of this type.

□

Exercises

Exercise 93: Show that contract $\Psi^\star \xleftarrow{map\ f} \Phi^\star$ holds provided contract $\Psi \xleftarrow{f} \Phi$ holds.

$\square$

Exercise 94: Suppose a functional programmer wishes to prove the following property of lists:

$$\left\langle \begin{array}{c} \forall\ a, s \\ (\phi\ a) \wedge \langle \forall\ a' \ :\ a' \in elems\ s :\ \phi\ a' \rangle : \\ \langle \forall\ a'' \ :\ a'' \in elems(a : s) :\ \phi\ a'' \rangle \end{array} \right\rangle$$

Show that this property is a contract arising (for free) from the polymorphic type of operation $(\_ : \_)$ on lists.

$\square$

Background

Going pointwise (39):

$$R \subseteq S \quad \Leftrightarrow \quad \langle \forall\ b, a\ ::\ b\ R\ a \Rightarrow b\ S\ a \rangle$$

Function converses (47):

$$(f\ b) R (g\ a) \quad \Leftrightarrow \quad b (f^{\circ} \cdot R \cdot g) a$$

Shunting rule (54):

$$f \cdot R \subseteq S \quad \Leftrightarrow \quad R \subseteq f^{\circ} \cdot S$$

Shunting rule (55):

$$R \cdot f^{\circ} \subseteq S \quad \Leftrightarrow \quad R \subseteq S \cdot f$$

K. Backhouse and R.C. Backhouse.
Safety of abstract interpretations for free, via logical relations and Galois connections.
SCP, 15(1–2):153–196, 2004.

R.C. Backhouse, P. de Bruin, P. Hoogendijk, G. Malcolm, T.S. Voermans, and J. van der Woude.
Polynomial relators.
In *AMAST'91*, pages 303–362. Springer, 1992.

J.C. Reynolds.
Types, abstraction and parametric polymorphism.
Information Processing 83, pages 513–523, 1983.

P.L. Wadler.
Theorems for free!
In *4th International Symposium on Functional Programming Languages and Computer Architecture*, pages 347–359, London, Sep. 1989. ACM.