

“Theorems for free”: a (calculational) introduction

J.N. Oliveira

Dept. Informática,
Universidade do Minho
Braga, Portugal

2003 (updated 2008, 2009, 2011)

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:

$$f : t$$

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

$$t ::= t' \leftarrow t''$$

$$t ::= F(t_1, \dots, t_n)$$

$$t ::= v \quad \text{type variables } v, \text{ cf. } \textit{polymorphism}$$

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:

$$R_{t:=v} = R_v \quad (1)$$

$$R_{t:=F(t_1, \dots, t_n)} = F(R_{t_1}, \dots, R_{t_n}) \quad (2)$$

$$R_{t:=t' \leftarrow t''} = R_{t'} \leftarrow R_{t''} \quad (3)$$

Questions: What does F mean on the RHS of (2)? 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] whenever, given a relation from A to B , GR extends R to G -structures: it is a relation from GA to GB

$$\begin{array}{ccc} A & \dots\dots\dots & GA \\ \downarrow R & & \downarrow GR \\ B & \dots\dots\dots & GB \end{array} \quad (4)$$

which obeys the following properties:

$$G id = id \quad (5)$$

$$G(R \cdot S) = (GR) \cdot (GS) \quad (6)$$

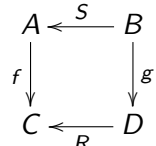
$$G(R^\circ) = (GR)^\circ \quad (7)$$

and is monotonic:

$$R \subseteq S \Rightarrow GR \subseteq GS \quad (8)$$

Background: “Reynolds arrow” operator

Define

$$f(R \leftarrow S)g \Leftrightarrow f \cdot S \subseteq R \cdot g \quad (9)$$


A commutative diagram with nodes A, B, C, D. A horizontal arrow labeled S points from B to A. A horizontal arrow labeled R points from D to C. A vertical arrow labeled f points from A to C. A vertical arrow labeled g points from B to D.

That is to say,

$$\frac{\begin{array}{ccc} A & \xleftarrow{S} & B \\ & R & \\ C & \xleftarrow{R} & D \end{array}}{C^A \xleftarrow{R \leftarrow S} D^B}$$

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

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 function of type t

First example (id)

- The target function: $\theta = id : a \leftarrow a$.
- Calculation of $R_{t=a \leftarrow a}$:

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

- Calculation of FT (R_a abbreviated to R):

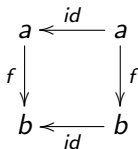
$$\begin{aligned} & id(R \leftarrow R)id \\ \Leftrightarrow & \quad \{ (9) \} \\ & id \cdot R \subseteq R \cdot id \end{aligned}$$

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.



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

Second example (*invl*)

- The target function: $\theta = \text{invl} : a^* \leftarrow a^*$.
- Calculation of $R_{t=a^* \leftarrow a^*}$:

$$\begin{aligned} & R_{a^* \leftarrow a^*} \\ \Leftrightarrow & \quad \left\{ \text{rule } R_{t=t' \leftarrow t''} = R_{t' \leftarrow R_{t''}} \right\} \\ & R_{a^* \leftarrow R_{a^*}} \\ \Leftrightarrow & \quad \left\{ \text{rule } R_{t=F(t_1, \dots, t_n)} = F(R_{t_1}, \dots, R_{t_n}) \right\} \\ & R_{a^* \leftarrow R_{a^*}} \end{aligned}$$

where

$$s R^* s' \stackrel{\text{def}}{=} \text{length } s = \text{length } s' \wedge \langle \forall i : i \in \text{inds } s : (s \ i) R(s' \ i) \rangle$$

Calculation of FT follows.

Second example (*invl*)

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

$$\begin{aligned} & \text{invl}(R^* \leftarrow R^*)\text{invl} \\ \Leftrightarrow & \quad \{ \text{definition } f(R \leftarrow S)g \Leftrightarrow f \cdot S \subseteq R \cdot g \} \\ & \text{invl} \cdot R^* \subseteq R^* \cdot \text{invl} \end{aligned}$$

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

$$\text{invl} \cdot r^* = r^* \cdot \text{invl}$$

that is,

$$\text{invl} [r a \mid a \leftarrow I] = [r b \mid b \leftarrow \text{invl} I]$$

Second example (*invl*)

Further calculation (back to R):

$$\begin{aligned} & \textit{invl} \cdot R^* \subseteq R^* \cdot \textit{invl} \\ \Leftrightarrow & \quad \{ \text{shunting rule (18)} \} \\ & R^* \subseteq \textit{invl}^\circ \cdot R^* \cdot \textit{invl} \\ \Leftrightarrow & \quad \{ \text{going pointwise (16, 17)} \} \\ & \langle \forall s, r :: s R^* r \Rightarrow (\textit{invl} s) R^* (\textit{invl} r) \rangle \end{aligned}$$

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

Second example (*invl*)

$$\text{length } s = \text{length } r \wedge \langle \forall i : i \in \text{inds } s : (s \ i) < (r \ i) \rangle$$

\Downarrow

$$\text{length}(\text{invl } s) = \text{length}(\text{inv } r)$$

\wedge

$$\langle \forall j : j \in \text{inds } s : (\text{invl } s)j < (\text{invl } r)j \rangle$$

(Guess other instances.)

Third example: FT of *sort*

Our next example calculates the FT of

$$\text{sort} : a^* \leftarrow a^* \leftarrow (\text{Bool} \leftarrow (a \times a))$$

(the first parameter stands for the chosen ordering relation):

$$\text{sort}(R_{(a^* \leftarrow a^*) \leftarrow (\text{Bool} \leftarrow (a \times a))}) \text{sort}$$

$$\Leftrightarrow \{ (2, 1, 3); \text{abbreviate } R_a := R \}$$

$$\text{sort}((R^* \leftarrow R^*) \leftarrow (R_{\text{Bool}} \leftarrow (R \times R))) \text{sort}$$

$$\Leftrightarrow \{ R_{t:=\text{Bool}} = \text{id} \text{ (constant relator) — cf. exercise 8} \}$$

$$\text{sort}((R^* \leftarrow R^*) \leftarrow (\text{id} \leftarrow (R \times R))) \text{sort}$$

Third example: FT of *sort*

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

\Leftrightarrow { (9) }

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

\Leftrightarrow { shunting (18) }

$(id \leftarrow (R \times R)) \subseteq sort^{\circ} \cdot (R^* \leftarrow R^*) \cdot sort$

\Leftrightarrow { introduce variables f and g (16, 17) }

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

\Leftrightarrow { (9) twice }

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

Third example: FT of *sort*

Case $R := r$:

$$f \cdot (r \times r) = g \Rightarrow (\text{sort } f) \cdot r^* = r^* \cdot (\text{sort } g)$$

\Leftrightarrow { introduce variables }

$$\left\langle \forall a, b :: f(r\ a, r\ b) = g(a, b) \right\rangle \Rightarrow \left\langle \forall l :: (\text{sort } f)(r^* l) = r^*(\text{sort } g\ l) \right\rangle$$

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

$$\left\langle \forall a, b :: r\ a \leq r\ b \Leftrightarrow a \preceq b \right\rangle \Rightarrow \left\langle \text{sort } (\leq)(r^* l) = r^*(\text{sort } (\preceq) l) \right\rangle$$

That is, for r monotonic and injective,

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

is always the same list a

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

Exercises

Exercise 1: Let C be a nonempty data domain and let $c \in C$. Let \underline{c} be the “everywhere c ” function:

$$\begin{array}{l} \underline{c} \quad : \quad A \longrightarrow C \\ \underline{c} a \quad \triangleq \quad c \end{array} \quad (10)$$

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

$$\langle \forall R :: R \subseteq T \rangle \quad (11)$$

□

Exercise 2: 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 3: The following is a well-known Haskell function

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

Calculate the free theorem associated with its type

$$\text{filter} : a^* \leftarrow a^* \leftarrow (\text{Bool} \leftarrow a)$$

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



Exercise 4: 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: $\text{serial} : (a \rightarrow \mathbb{N}) \rightarrow a^* \rightarrow a^*$.

Calculate the FT of *serial*.



Exercises

Exercise 5: 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, *findIndices* (*< 0*) [1, -2, 3, 0, -5] = [1, 4]. Calculate the FT of this function.



Exercise 6: Choose arbitrary functions from Haskell's Prelude and calculate their FT.



Fourth example: FT of $(\lfloor _)$

Recall the catamorphism (fold) combinator:

$$\begin{array}{ccc} F a & \xleftarrow{\text{in}_{F a}} & B(a, F a) \\ \downarrow (\lfloor g) & & \downarrow B(\text{id}, (\lfloor g)) \\ b & \xleftarrow{g} & B(a, b) \end{array}$$

So $(\lfloor _)$ has generic type

$$(\lfloor _): b \leftarrow F a \leftarrow (b \leftarrow B(a, b))$$

where $F a \cong B(a, F a)$. Then $(\lfloor _)$ -FT is

$$(\lfloor _) \cdot (R_b \leftarrow B(R_a, R_b)) \subseteq (R_b \leftarrow F R_a) \cdot (\lfloor _)$$

Fourth example: FT of $(\lfloor - \rfloor)$

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

$$(\lfloor - \rfloor) \cdot (S \leftarrow B(R, S)) \subseteq (S \leftarrow F R) \cdot (\lfloor - \rfloor)$$

$$\Leftrightarrow \{ \text{shunting (18)} \}$$

$$(S \leftarrow B(R, S)) \subseteq (\lfloor - \rfloor)^\circ (S \leftarrow F R) \cdot (\lfloor - \rfloor)$$

$$\Leftrightarrow \{ \text{introduce variables } f \text{ and } g \text{ (16, 17)} \}$$

$$f(S \leftarrow B(R, S))g \Rightarrow (\lfloor f \rfloor)(S \leftarrow F R)(\lfloor g \rfloor)$$

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

$$f \cdot B(R, S) \subseteq S \cdot g \Rightarrow (\lfloor f \rfloor) \cdot F R \subseteq S \cdot (\lfloor g \rfloor)$$

$(_)$ -FT corollaries

From

$$f \cdot B(R, S) \subseteq S \cdot g \Rightarrow (f) \cdot FR \subseteq S \cdot (g) \quad (12)$$

we can infer:

- $(_)$ -fusion $(R, S := id, s)$:

$$f \cdot B(id, s) = s \cdot g \Rightarrow (f) = s \cdot (g) \quad (13)$$

- $(_)$ -absorption $(R, S := r, id)$:

$$f \cdot B(r, id) = g \Rightarrow (f) \cdot Fr = (g) \quad (14)$$

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

$$(f) \cdot Fr = (f \cdot B(r, id)) \quad (15)$$

Exercises

Exercise 7: Let $iprod = ([1, (\times)])$ be the function which multiplies all natural numbers in a given list; $even$ be the predicate which tests natural numbers for evenness; and $exists = ([\underline{FALSE}], (\vee))$.
From (12) infer

$$even \cdot iprod = exists \cdot even^*$$

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

□

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*.

Exercises

Exercise 8: Show that the *identity* relator Id , which is such that $Id R = R$ and the *constant* relator K (for a given data type K) which is such that $K R = id_K$ are indeed relators.

□

Exercise 9: Show that product

$$\begin{array}{ccc} A & C & \dots\dots\dots G(A, C) = A \times C \\ R \downarrow & S \downarrow & \downarrow G(R,S)=R \times S \\ B & D & \dots\dots\dots G(B, D) = B \times D \end{array}$$

is a (binary) relator.

□

Background

Going pointwise:

$$R \subseteq S \Leftrightarrow \langle \forall b, a :: b R a \Rightarrow b S a \rangle \quad (16)$$

Function converses:

$$(f \cdot b)R(g \cdot a) \Leftrightarrow b(f^\circ \cdot R \cdot g)a \quad (17)$$

Shunting rules:

$$f \cdot R \subseteq S \Leftrightarrow R \subseteq f^\circ \cdot S \quad (18)$$

$$R \cdot f^\circ \subseteq S \Leftrightarrow R \subseteq S \cdot f \quad (19)$$



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.