Theorems for Free: an Introduction

DI/UM, 2003

José N. Oliveira

Dept. Informatica

Universidade do Minho, 4700 Braga, Portugal

jno@di.uminho.pt

Parametric polymorphism: why?

- Less code (**specific** solution = **generic** solution + **costumization**)
- Intelectual reward
- Last but not least, quotation (from Theorems for free!, by Philip Wadler [Wad89]):

  From the type of a polymorphic function we can derive a theorem that is 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:

$$\begin{array}{lcl} t & ::= & t' \leftarrow t'' \\ t & ::= & \mathsf{F}(t_1, \ldots, t_n) \\ t & ::= & v \end{array}$$

type variables v, cf. polymorphism

What does it mean that f is **parametrically** polymorphic?

Free theorem of type t

Let

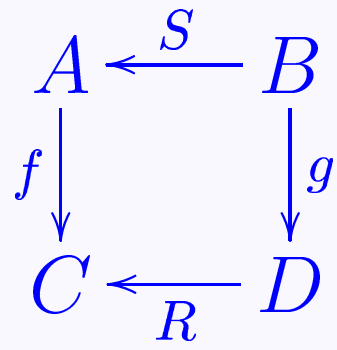
- V be the set of type variables involved in type t
- $\{R_v\}_{v \in V}$ be a V-indexed family of relations (f_v in case all such R_v are functions).
- R_t be a relation defined inductively as follows:

$$\begin{aligned} R_{t=\mathsf{F}(t_1,\ldots,t_n)} &= \mathsf{F}(R_{t_1}, \ldots, R_{t_n}) \\ R_{t=v} &= R_v \\ R_{t=t' \leftarrow t''} &= R_{t'} \leftarrow R_{t''} \end{aligned}$$

- What kind of relation is $R_{t'} \leftarrow R_{t''}$?

Reynolds arrow operator

$$f(R \leftarrow S)g \;\equiv\; f \cdot S \subseteq R \cdot g$$



That is to say,

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

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

Free theorem (FT) of type t

The following (remarkable) theorem — due to J. Reynolds and advertised by P. Wadler — holds:

> *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

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

$$
\begin{aligned}
& R_{a^\star \leftarrow a^\star} \\
\equiv & \quad \{ \text{ rule } R_{t=t' \leftarrow t''} = R_{t'} \leftarrow R_{t''} \} \\
& R_{a^\star} \leftarrow R_{a^\star} \\
\equiv & \quad \{ \text{ rule } R_{t=\mathsf{F}(t_1,\ldots,t_n)} = \mathsf{F}(R_{t_1}, \ldots, R_{t_n}) \} \\
& R_a{}^\star \leftarrow R_a{}^\star
\end{aligned}
$$

- Calculation of FT $invl\ (R_{a^\star \leftarrow a^\star})\ invl$ follows

$invl$ FT calculation

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

$$
\begin{array}{rl}
 & invl(R^\star \leftarrow R^\star)invl \\
\equiv & \quad \{ \text{ definition } f(R \leftarrow S)g \equiv f \cdot S \subseteq R \cdot g \} \\
 & invl \cdot R^\star \subseteq (R^\star) \cdot invl
\end{array}
$$

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

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

Further calculation (back to R):

Pointwise version of FT

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

$\equiv$ { shunting rule }

$$R^\star \subseteq invl^\circ \cdot (R^\star) \cdot invl$$

$\equiv$ { going pointwise }

$$l(R^\star)r \;\Rightarrow\; (invl\ l)(R^\star)(invl\ r)$$

$\equiv$ { }

$$\forall i \in inds\ l.(l\ i)R(r\ i) \;\Rightarrow\; (invl\ l)(R^\star)(invl\ r)$$

Pointwise version of FT

For example, $invl$ will respect orderings:

$$\begin{aligned} & \forall i \in inds\ l.(l\ i) < (r\ i) \\ \Rightarrow\ & \forall j \in inds(invl\ l).(invl\ l)j < (invl\ r)j \end{aligned}$$

Exercise: calculate the FT of

$$sort : a^\star \leftarrow a^\star \leftarrow (2 \leftarrow (a \times a))$$

(the first parameter stands for the ordering relation.)

FT of $(\!|\_|\!)$

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

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

  where $\mathsf{F}\, a \cong \mathsf{B}\,(a, \mathsf{F}\, a)$.

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

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

- FT-$(\!|\_|\!)$ calculation follows (R_a, R_b abbreviated to R, S):

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

$$(\!|\_|\!) \cdot (S \leftarrow \mathsf{B}\,(R,S)) \;\subseteq\; (S \leftarrow \mathsf{F}\,R) \cdot (\!|\_|\!)$$

$\equiv$ { definition $f(R \leftarrow S)g \;\equiv\; f \cdot S \subseteq R \cdot g$ }

$$f(S \leftarrow \mathsf{B}\,(R,S))g \;\Rightarrow\; (\!|f|\!)(S \leftarrow \mathsf{F}\,R)(\!|g|\!)$$

$\equiv$ { idem }

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

At this point, we can infer ...

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

From this, infer

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

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

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

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

$$\equiv \qquad \{\ \text{replacement of } g\ \}$$

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

Mutual recursion

Consider mutually-dependent f and g as follows:

```
f: nat -> nat
f(n) == if n = 0 then 0 else g(n - 1)

g: nat -> nat
g(n) == if n = 0 then 1
        else f(n - 1) + g(n - 1);
```

How do we reason about mutually-dependent functions?

Mutual-recursion law

The situation is handled by the so-called mutual-recursion law, also called “Fokkinga law”:

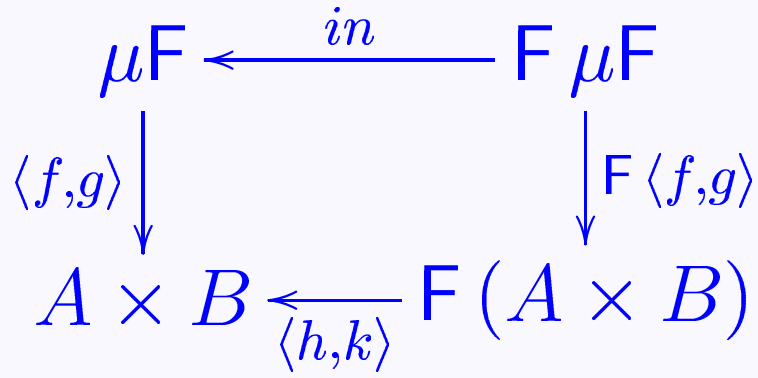
$$\begin{array}{l} f \cdot in = h \cdot \mathsf{F}\,\langle f, g\rangle \\ \qquad\qquad \wedge \qquad\qquad \Rightarrow \quad \langle f, g\rangle = (\!|\langle h, k\rangle|\!) \\ g \cdot in = k \cdot \mathsf{F}\,\langle f, g\rangle \end{array}$$

In terms of diagrams: from

$$\begin{array}{ccc} \mathsf{T} & \xleftarrow{in} & \mathsf{F}\,\mathsf{T} \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle \mathsf{F}\,\langle f,g\rangle} \\ A & \xleftarrow[h]{} & \mathsf{F}\,(A \times B) \end{array} \qquad \begin{array}{ccc} \mathsf{T} & \xleftarrow{in} & \mathsf{F}\,\mathsf{T} \\ {\scriptstyle g}\downarrow & & \downarrow{\scriptstyle \mathsf{F}\,\langle f,g\rangle} \\ B & \xleftarrow[k]{} & \mathsf{F}\,(A \times B) \end{array}$$

Mutual-recursion law

. . . we get



Proof:

$$\langle f, g\rangle \cdot in = \langle h, k\rangle \cdot \mathsf{F}\,\langle f, g\rangle$$

$\equiv$ { by $\times$-fusion }

$$\langle f, g\rangle \cdot in = \langle h \cdot \mathsf{F}\,\langle f, g\rangle, k \cdot \mathsf{F}\,\langle f, g\rangle\rangle$$

$\equiv$ { by hypothesis}

Proof

$$\langle f, g\rangle \cdot in = \langle f \cdot in, g \cdot in\rangle$$

$\equiv$ { by (reverse) $\times$-fusion }

$$\langle f, g\rangle \cdot in = \langle f, g\rangle \cdot in$$

$\equiv$ { equality is reflexive}

T

Applying this to the above pair of f and g:

$$\begin{aligned} f \cdot [\underline{0}, suc] &= [\underline{0}, g] \\ g \cdot [\underline{0}, suc] &= [\underline{1}, + \cdot \langle f, g\rangle] \end{aligned}$$

Mutual-recursion law

The mutual dependence can be made more explicit

$$\begin{aligned} f \cdot [\underline{0}, suc] &= [\underline{0}, \pi_2 \cdot \langle f, g \rangle] \\ g \cdot [\underline{0}, suc] &= [\underline{1}, + \cdot \langle f, g \rangle] \end{aligned}$$

The underlying inductive type is

$$\mathbb{N}_0 \cong \underbrace{1 + \mathbb{N}_0}_{\mathsf{F}\,\mathbb{N}_0} \tag{1}$$

which is such that $\mathsf{F}\, f = id + f$. So we can write

$$\begin{aligned} f \cdot in &= [\underline{0}, \pi_2] \cdot \mathsf{F}\, \langle f, g \rangle \\ g \cdot in &= [\underline{1}, +] \cdot \mathsf{F}\, \langle f, g \rangle \end{aligned}$$

Mutual-recursion law

So we identify $h = [\underline{0}, \pi_2]$ and $k = [\underline{1}, +]$ therefore obtaining

$$
\begin{aligned}
& \langle f, g \rangle \\
= & \quad \{ \text{Fokkinga law} \} \\
& (\!| \langle [\underline{0}, \pi_2], [\underline{1}, +] \rangle |\!) \\
= & \quad \{ \text{exchange law} \} \\
& (\!| [\langle \underline{0}, \underline{1} \rangle, \langle \pi_2, + \rangle] |\!)
\end{aligned}
$$

which is easily converted into VDM-SL as follows:

```
fg: nat -> (nat*nat)
fg(n) == if n = 0 then mk_(0,1)
         else let p=fg(n - 1)
              in mk_(p.#2,p.#1 + p.#2);
```

Corollary: “banana-split”

Let $h = i \cdot \mathsf{F}\,\pi_1$ and $k = j \cdot \mathsf{F}\,\pi_2$ in Fokkinga law. Then

$$
\begin{array}{ll}
 & f \cdot in = (i \cdot \mathsf{F}\,\pi_1) \cdot \mathsf{F}\,\langle f, g\rangle \\
\equiv & \quad \{ \text{ composition is associative and F is a functor} \} \\
 & f \cdot in = i \cdot \mathsf{F}\,(\pi_1 \cdot \langle f, g\rangle) \\
\equiv & \quad \{ \text{ by } \times\text{-cancellation } \} \\
 & f \cdot in = i \cdot \mathsf{F}\,f \\
\equiv & \quad \{ \text{ by cata-cancellation} \} \\
 & f = (\!|i|\!)
\end{array}
$$

Similarly, from $k = j \cdot \mathsf{F}\,\pi_2$ we get $g = (\!|j|\!)$.

Corollary: “banana-split”

Then we get

$$\langle (\!|i|\!), (\!|j|\!) \rangle = (\!|\langle i \cdot \mathsf{F}\,\pi_1, j \cdot \mathsf{F}\,\pi_2 \rangle|\!)$$

that is

$$\langle (\!|i|\!), (\!|j|\!) \rangle = (\!|(i \times j) \cdot \langle \mathsf{F}\,\pi_1, \mathsf{F}\,\pi_2 \rangle|\!) \tag{2}$$

by (reverse) $\times$-absorption.

Comment: This law provides us with a very useful tool for “parallel loop” inter-combination: “loops” $(\!|i|\!)$ and $(\!|j|\!)$ are fused together into a single “loop” $(\!|(i \times j) \cdot \langle \mathsf{F}\,\pi_1, \mathsf{F}\,\pi_2 \rangle|\!)$. The need for this kind of calculation arises very often.

Bibliography

[Wad89] P. Wadler. Theorems for free! In *4th International Symposium on Functional Programming Languages and Computer Architecture*, London, Sep. 1989. ACM.

References

[Wad89] P. Wadler. Theorems for free! In *4th International Symposium on Functional Programming Languages and Computer Architecture*, London, Sep. 1989. ACM.