Generic Programming

Primitive (Co)Recursion and Course-of-Value (Co)Iteration, Categorically

Tarmo Uustalu and Varmo Vene

João Alpuim

Utrecht University

October 18, 2012

(Co-)Datatypes

Datatypes:

Natural numbers:
Algebra (initial): $(\mu \mathrm{N}, \mathrm{in}_N)$
Functor N with
$\mathrm{N}\ \mathrm{X} = 1 + \mathrm{X}$ and
$\mathrm{N}\ \mathrm{f} = \mathrm{id} + \mathrm{f}$

Co-datatypes:

Streams :
Co-algebra (terminal): $(\nu \mathrm{S}_A, \mathrm{out}_{S_A})$
Functor S_A with
$\mathrm{S}_A\ \mathrm{X} = \mathrm{A} \times \mathrm{X}$
$\mathrm{S}_A\ \mathrm{f} = \mathrm{id}_A \times \mathrm{f}$

(Co-)Datatypes in Haskell

```
data Fix f = In { out :: f (Fix f) }

data N r = Z | S r
type Nat = Fix N

instance Functor N where
   fmap f Z     = Z
   fmap f (S r) = S (f r)

zeroN = In Z
succN n = In (S n)
```

(Co-)Datatypes in Haskell (cont.)

```haskell
data S a x = St a x

instance Functor (S a) where
   fmap f (St x xs) = St x (f xs)

type Stream a = Fix (S a)

headS :: Stream a -> a
headS xs = case out xs of
                St x _ -> x

tailS :: Stream a -> Stream a
tailS xs = case out xs of
                St _ xs' -> xs'
```

Iteration / Catamorphisms

As you have seen earlier:
$f = (\!| \varphi |\!)_F \Leftrightarrow f \cdot in_F = \varphi \cdot F\ f$

In Haskell:

```
cata :: Functor f => (f c -> c) -> Fix f -> c
cata phi = phi . fmap (cata phi) . out
```

Recall: We have seen this under the name *fold*.

An interesting property: $out_F = (\!| F\ in_F |\!)_F$

Co-iteration / Anamorphisms

As you have seen earlier:
$\mathrm{out}_F \cdot f = F\ f \cdot \varphi \Leftrightarrow f = [(\ \varphi\ )]_F$

In Haskell:

```
ana :: Functor f => (c -> f c) -> c -> Fix f
ana phi = In . fmap (ana phi) . phi
```

Paramorphisms

Paramorphisms are defined by:
$\triangleleft\ \varphi\ \triangleright_F = \mathsf{fst} \cdot (\!|\ < \varphi, \mathsf{in}_F \cdot \mathsf{F}\ \mathsf{snd} >\ |\!)_F$

Which have the following universal property:
$\mathsf{f} = \triangleleft\ \varphi\ \triangleright \Leftrightarrow \mathsf{f} \cdot \mathsf{in}_F = \varphi \cdot \mathsf{F} < \mathsf{f}\ , \mathsf{id}_{\mu F} >$

In Haskell:

```
para :: Functor f => (f (c, Fix f) -> c) -> Fix f -> c
para phi = fst . cata (fork phi (In . fmap snd))
```

Paramorphisms (cont.)

- Paramorphisms generalize catamorphisms:
- $(\!|\ \varphi\ |\!)_F = \lhd\ \varphi \cdot \mathsf{F}\ \mathsf{fst}\ \rhd$
- In fact, paramorphisms generalise any function with domain equal to the carrier of an initial algebra:
- $\mathsf{f} = \lhd\ \mathsf{f} \cdot \mathsf{in}_F \cdot \mathsf{F}\ \mathsf{snd}\ \rhd_F$
- From this rule derives:
- $\mathsf{out}_F = \lhd\ \mathsf{F}\ \mathsf{snd}\ \rhd_F$

Apomorphisms

We may dualise the theory from paramorphisms:
$[\langle\ \varphi\ \rangle]_F = [(\ [\ \varphi\ ,\ \mathsf{F}\ \mathsf{right} \cdot \mathsf{out}_F\ ]\ )]_F \cdot \mathsf{inl}$

Apomorphisms are characterised by the following universal property:
$\mathsf{f} = [\langle\ \varphi\ \rangle]_F \Leftrightarrow \mathsf{out}_F \cdot \mathsf{f} = \mathsf{F}\ [\ \mathsf{f}\ ,\ \mathsf{id}_{\nu F}\ ]$

In Haskell:

```
apo :: Functor f => (c -> f (Either c (Fix f))) -> c -> Fix f
apo phi = ana (join phi (fmap Right . out)) . Left
```

Histomorphisms

- Now, consider we want to define the fibonacci function.
- We could define it as:
  $\mathsf{fibo} = \mathsf{fst} \cdot (\!| < [\ \mathsf{one}\ ,\ \mathsf{add}\ ]\ ,\ [\ \mathsf{zero}\ ,\ \mathsf{fst}\ ] > |\!)_N$
- However, we want to capture the function's natural definition;
- For any function requiring any subparts of any depth...
- Solution: generalise the current primitive iteration to a course-of-value iteration.

Histomorphisms (cont.)

Let F_A^X be the (bi)functor in which:
$F_A^X\ X = A \times F\ X$ and $F_A^X\ h = id_A \times F\ h$

An histomorphism would have the following definition:
$\{\!|\ \varphi\ |\!\}_F = fst \cdot out_F^{\times} \cdot (\!|\ in_F^{\times} \cdot <\varphi\ ,\ id>\ |\!)_F$

With the following universal property:
$f = \{\!|\ \varphi\ |\!\}_F \Leftrightarrow f \cdot in_F = \varphi \cdot F\ [(\ <f\ ,\ out_F>\ )]_F^{\times}$

This would lead to a possible fib re-definition:
$\{\!|\ [\ one\ ,\ [\ one \cdot snd\ ,\ add \cdot (id \times (fst \cdot out_{N\times}))]\ \cdot distl \cdot out_{N\times}\ |\!\}_N$

Futumorphisms

Let F^+_A be the (bi)functor in which:
$\mathsf{F}^+_A\ \mathsf{X} = \mathsf{A} + \mathsf{F}\ \mathsf{X}$ and $\mathsf{F}_A + X\ \mathsf{h} = \mathsf{id}_A + \mathsf{F}\ \mathsf{h}$

A futumorphism would have the following definition:
$[\{\ \varphi\ \}]_F = [(\ [\ \varphi\ ,\ \mathsf{id}\ ] \cdot \mathsf{out}_{F^+}\ )]^+_F \cdot \mathsf{in}_{F^+} \cdot \mathsf{left}$

With the following universal property:
$\mathsf{f} = [\{\ \varphi\ \}]_F \Leftrightarrow \mathsf{out}_F \cdot \mathsf{f} = \mathsf{F}\ (\!|\ [\ \mathsf{f}\ ,\ \mathsf{in}_F\ ]\ |\!)_{F^+} \cdot \varphi$

Conclusion

- Good attempt to generalise;
- Hard to reason about;
- Efficient implementation (histomorphism).

Generic Programming

Primitive (Co)Recursion and Course-of-Value (Co)Iteration, Categorically

Tarmo Uustalu and Varmo Vene

João Alpuim

Utrecht University

October 18, 2012