

Extending OCaml with copatterns and first-class labels for Free

YANN RÉGIS-GIANAS, PAUL LAFORGUE, Diderot University

ACM Reference format:

Yann Régis-Gianas, Paul Laforgue. 2017. Extending OCaml with copatterns and first-class labels for Free. *ACM J. Comput. Cult. Herit.* 9, 4, Article 00 (March 2017), 3 pages.

DOI: 0000001.0000001

1 SYNTAX

Here we describe the syntax of our extension. This one is conflictless with the actual syntax used in OCaml 4.04. In the following section :

- Programs with a yellow background stand for the original code, while blue ones represent generated code.
- Data constructors indexed with a small cross (such as **Stream^x**) are inaccessible for the programmer.
- Infix symbol (\triangleright) corresponds to the reverse-application operator such that $(x \triangleright f)$ is equivalent to $f(x)$.
- Type constructors `codata` and `query` are abstracts and automatically imported from module `Pervasives`.

1.1 Codata types

Codata types are introduced with the **type** keyword.

```
type  $\alpha$  !stream = {
  Head :  $\alpha$ ;
  Tail  :  $\alpha$  !stream;
}
```

```
type ( $\sigma, \alpha$ ) stream =
  | Streamx : {dispatch :  $\sigma.(\sigma \text{ query}, \alpha) \text{ stream} \rightarrow \sigma\} \rightarrow (\text{codata}, \alpha) \text{ stream}$ 
  | Head    : ( $\alpha \text{ query}, \alpha$ ) stream
  | Tail    : (((codata,  $\alpha$ ) stream) query,  $\alpha$ ) stream
let head = function Streamx {dispatch}  $\rightarrow$  dispatch Head
let tail  = function Streamx {dispatch}  $\rightarrow$  dispatch Tail
```

```
type ( $\alpha, \beta$ ) !product = {
  Fst :  $\alpha$ ;
  Snd :  $\beta$ ;
}
```

ACM acknowledges that this contribution was authored or co-authored by an employee, or contractor of the national government. As such, the Government retains a nonexclusive, royalty-free right to publish or reproduce this article, or to allow others to do so, for Government purposes only. Permission to make digital or hard copies for personal or classroom use is granted. Copies must bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. To copy otherwise, distribute, republish, or post, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2017 ACM. XXXX-XXXX/2017/3-ART00 \$15.00

DOI: 0000001.0000001

```

type ( $\sigma, \alpha, \beta$ ) product =
  | Productx : { dispatch :  $\sigma.(\sigma \text{ query}, \alpha, \beta)$  product  $\rightarrow \sigma$  }  $\rightarrow$  (codata,  $\alpha, \beta$ ) product
  | Fst      : ( $\alpha$  query,  $\alpha, \beta$ ) product
  | Snd      : ( $\beta$  query,  $\alpha, \beta$ ) product
let fst = function Productx { dispatch }  $\rightarrow$  dispatch Fst
let snd = function Productx { dispatch }  $\rightarrow$  dispatch Snd

```

1.2 Copattern matching

```

let zeros = comatch zs : int !stream with
  | zs#Head  $\rightarrow$  0
  | zs#Tail  $\rightarrow$  zs

```

```

let zeros =
  let rec zs : (codata, int) stream =
    let dispatch : type  $\sigma.(\sigma \text{ query}, \text{int}) \text{ stream} \rightarrow \sigma$  = function
      | Head  $\rightarrow$  0
      | Tail  $\rightarrow$  zs
    in Stream { dispatch }
  in zs

```

1.3 Unnesting copatterns

Here, we process more or less as in [?].

```

let fibonacci = comatch fib : int !stream with
  | fib#Head  $\rightarrow$  0
  | fib#Tail#Head  $\rightarrow$  1
  | fib#Tail#Tail  $\rightarrow$  zipWith (+) fib fib#Tail

```

```

let fibonacci =
  let rec fib : (codata, int) stream =
    let f1 =
      let dispatch : type  $\sigma.(\sigma \text{ query}, \text{int}) \text{ stream} \rightarrow \sigma$  = function
        | Head  $\rightarrow$  1
        | Tail  $\rightarrow$  zipWith (+) fib (fib  $\triangleright$  tail)
      in Streamx { dispatch }
    in
      let dispatch : type  $\sigma.(\sigma \text{ query}, \text{int}) \text{ stream} \rightarrow \sigma$  = function
        | Head  $\rightarrow$  0
        | Tail  $\rightarrow$  f1
      in Streamx { dispatch }
    in fib

```

1.4 BNF

REFERENCES

- [] Anton Setzer, Andreas Abel, Brigitte Pientka, and David Thibodeau. 2014. Unnesting of copatterns. In *International Conference on Rewriting Techniques and Applications*. Springer, 31–45.

Received February 2007; revised March 2009; accepted June 2009