CAML

Titolo nota 24/09/2015

interprete che valuta espressioni

```
# 4+3;;
- : int = 7

# let n = 3+4;;
n : int = 7

# let f n = n+1;;
f : int -> int = <fun>
```

```
#let g (n,m) = n+m+1;;
 g : int * int -> int = <fun>

# let (s n m = n+m+1;;
 s : int -> (int -> int) = <fun>

# let i x = x;;
i : 'a -> 'a = <fun>

# i 5;;
- : int = 5

# i 'a';;
- : char = 'a'
```

Curryed

polimorfa

variabili di tipo

Titolo nota 24/09/2015

applicazione di funzione $f(x)$

```
# let apply f x = f x ;;
apply : ('a -> 'b) -> 'a -> 'b = <fun>
```

('a -> 'b) tipo di f; 'a tipo di x; 'b tipo risultato

```
# let s n = n+2 ;;
s : int -> int = <fun>
```

'a -> 'b diventa int -> int

```
# apply s 3 ;;
- : int = 5
```

s: apply : (int -> int) -> int -> int

Titolo nota 24/09/2015

applicazione di funzione $f(x)$

```
# let apply f x = f x ;;
apply : ('a -> 'b) -> 'a -> 'b = <fun>
```

('a -> 'b) tipo di f; 'a tipo di x; 'b tipo risultato

```
# let s n = n+2 ;;
s : int -> int = <fun>

# apply s 3 ;;
- : int = 5
```

```
# apply s ;;
- : int -> int = <fun>

# let t = apply s ;;
t : int -> int = <fun>

# t 3 ;;
- : int = 5
```

int -> int

apply : ('a -> 'b) -> 'a -> 'b

s : (int -> int) ->

Titolo nota 24/09/2015

```
# let apply f x = f x ;;
  apply : ('a -> 'b) -> 'a -> 'b = <fun>
```

int bool int bool

```
# let f1 x = x > 0 ;;
  f1 : int -> bool = <fun>
```

```
let f1 x = x > 0.0 ;;
f1 : float -> bool = <fun>
```

```
# apply f1 ;;
- : int -> bool = <fun>

# apply f1 3 ;;
- : bool = true
```

definizione di funzioni per casi

Titolo nota 24/09/2015

$$f(n) = \begin{cases} 1 & \text{se } n = 0 \\ f(n-1) + 3 & \text{se } n > 0 \end{cases}$$

$f : \mathbb{N} \to \mathbb{N}$

- In CAML il tipo corrispondente a $\mathbb{N}$ non esiste (esistono solamente gli interi $\mathbb{Z}$)

- In CAML abbiamo la possibilità di usare espressioni condizionali (if ... then ... else ...)

- In CAML possiamo definire funzioni ricorsive

definizione di funzioni per casi

Titolo nota 24/09/2015

$$f(n) = \begin{cases} 1 & \text{se } n = 0 \\ f(n-1) + 3 & \text{se } n > 0 \end{cases}$$

$f : \mathbb{N} \to \mathbb{N}$

definizione ricorsiva

```
# let rec f n = if n=0 then 1
f : int -> int = <fun>   else f (n-1) + 3 ;;
```

l'applicazione di funzione ha PRECEDENZA sulle applicazioni degli operatori

$(f\ n) - 1 + 3$

$(f\ 2)$
$=$ {def. f, ramo else}
$(f\ 1) + 3$
$=$ {def f, ramo else}
$(f\ 0) + 3 + 3$
$=$ {def f, ramo then}
$1 + 3 + 3 = 7$

definizione di funzioni per casi

Titolo nota 24/09/2015

$$f(n) = \begin{cases} 1 & \text{se } n = 0 \\ f(n-1) + 3 & \text{se } n > 0 \end{cases}$$

$f : \mathbb{N} \to \mathbb{N}$

definizione ricorsiva

```
# let rec f n = if n=0 then 1
f : int -> int = <fun>   else f (n-1) + 3 ;;
```

$f(-2)$
$= \{$ def f, ramo else $\}$
$f(-3) + 3$
$= \{$ def. f. ramo else $\}$
$f(-4) + 3 + 3$
⋮
calcolo infinito

Titolo nota 24/09/2015

```
let rec f n = if n = 0 then 1
              else if n > 0 then f(n-1) + 3 ;;
```

errore perché non esiste l'espressione (if ... then ...)

```
let rec f n = if n = 0 then 1
              else f (n-1) + 3 ;;
```

Per tutte le definizioni ricorsive (anche quelle CAML) è possibile fare dimostrazioni induttive (su parte del dominio)

f : int → int = (fun) (Tralasciamo le parte $\mathbb{Z}^-$ (interi negativi) facciamo vedere che una Proprietà vale negli interi positivi o nulli)

Titolo nota 24/09/2015

```
let rec f n = if n=0 then 1 else f(n-1) +3;;
f: int → int = <fun>
```

Dimostriamo **induttivamente** su $\mathbb{Z} \setminus \mathbb{Z}^-$ (interi positivi o nulli = $\mathbb{N}$)

$(\forall n \in \mathbb{N}.\ f\ n = 3 \cdot n + 1)$

precedenza nei $\mathbb{N}$ indotta dalla definizione di funzione

$(\forall n, n' \in \mathbb{N}.\quad n \sqsubset n' \equiv n = n' - 1)$

Caso base

$f\ 0 = 3 \cdot 0 + 1$

$= \{$def f, ramo then$\}$

1

$= \{$calcolo$\}$

$3 \cdot 0 + 1$

Caso induttivo

$f\ (n+1)$

$= \{$def f, ramo else$\}$

$f\ n + 3$

$= \{$ip. induttiva$\}$

$f\ n = 3 \cdot n + 1 \Rightarrow f\ (n+1) = 3 \cdot (n+1) + 1$

(ip. induttiva)

$3n + 1 + 3$

$= \{$calcolo$\}$

$3(n+1) + 1$

fattoriale

Titolo nota 24/09/2015

```
# let rec fact n = if n=0 then 1
                   else n * fact (n-1) ;;
fact : int -> int = <fun>
```

int → tipo di n

int = <fun> → tipo del risultato

```
# fact 3 ;;
- : int = 6

# fact (-2) ;;
-   ) calcolo infinito
```

Definizioni ricorsive con ACCUMULATORE

Titolo nota 24/09/2015

```
# let rec f n = if n=0 then 1 else f(n-1) + 3 ;;
  f : int -> int = <fun>
```

f 4
= {def. f}
f 3 +3
= {def f}
f 2 +3 +3
= {def f}
f 1 +3 +3 +3
⋮

f 4 0
= {def f}
f 3 3
= {def f}
f 2 6
= {def..}
f 1 9

=
f 0 12
=
12 +1

Titolo nota 24/09/2015

```
let rec f n a = if n=0 then a+1
                else f (n-1) (a+3)
```

accumulo nel secondo parametro il risultato

quando il primo argomento di una chiamata di f diventa 0 nel secondo argomento ho accumulato il risultato (in questo caso - 1)

Titolo nota 24/09/2015

```
# let rec f n a = if n=0 then a+1
                  else f (n-1) (a+3)
```

$f : \underbrace{int}_{\text{tipo di } n} \to \underbrace{int}_{\text{tipo di } a} \to \underbrace{int}_{\text{tipo risultato}} = \langle fun \rangle$

$$\left(\forall n, m \in \mathbb{N}.\ f\ n\ m = 3 \cdot n + 1 + m\right)$$

Precedenza indotta . INTUITIVA

$$\left((n-1),(a+3)\right) \sqsubset (n, a)$$

(n, m)
$(n-1, m+3)$
$(0, m+3n)$

$f\ 2\ 0$
$= \{ \text{def } f \}$
$f\ 1\ 3$
$= \{ \text{def } f \}$
$f\ 0\ 6$
$= \{ \text{def } f \}$
$6+1$

$f\ 2\ 2$
$= \{ \text{def} \}$
$f\ 1\ 5$
$= \{ \text{def} \}$
$f\ 0\ 8$
$= \{ \text{def} \}$
$8+1$

Titolo nota 24/09/2015

Proprietà $(\forall n, m \in \mathbb{N}.\ f\ n\ m = 3n + 1 + m)$

```
# let rec f n a = if n=0 then a+1
                  else f (n-1) (a+3);;
```

$(\forall n, m, n', m' \in \mathbb{N}.\ (n, m) \sqsubset (n', m') \equiv n = n' - 1 \wedge m = m' + 3)$

Caso base

$f\ 0\ m = 3 \cdot 0 + 1 + m$

$f\ 0\ m$
$= \{\text{def. } f\}$
$m + 1$
$= \{\text{calcolo}\}$
$3 \cdot 0 + 1 + m$

Caso induttivo $f\ n\ m = 3n + 1 + m \Rightarrow f\ (n+1)\ (m-3) = 3 \cdot (n+1) + 1 + (m-3)$

$f\ (n+1)\ (m-3)$
$= \{\text{def. } f, \text{ramo else}\}$
$f\ n\ m$
$= \{\text{ip. induttiva}\}$
$3 \cdot n + 1 + m$
$= \{\text{calcolo}\}$

$3n + 1 + m + 3 - 3$
$= \{\text{calcolo}\}$
$3 \cdot (n+1) + 1 + (m-3)$

Titolo nota 24/09/2015

Proprietà $(\forall n, m \in \mathbb{N}.\ f\ n\ m = 3n+1+m)$ $3n+1+m$

```
# let rec f n a = if n=0 then a+1
                  else f (n-1) (a+3);;
```

```
# f 10 0;;
- : int = 31
```

dovrebbe essere 0

```
# f 10 2;;
- : int = 33
```

```
# let g n = f n 0;;
g : int -> int = <fun>
```

```
# g 10;;
- : int = 31
```

```
# f 10 3;;
- : int = 34
```

Dichiarazioni "LOCALI"

Titolo nota 24/09/2015

let ... in ...

questa dichiarazione è conosciuta solamente nella espressione dopo "in"

```
# let n = 10 in n + 1;;
- : int = 11
```

"-" non ho creato nessun nuovo nome

let ... in ... non crea nessuna associazione globale

```
# n;;
Unbound
```

Titolo nota 24/09/2015

```
# let g n = let rec f n m = if n=0 then m+1
                            else f (n-1) (m+3)
          in f n 2;;
```

```
g : int -> int = <fun>
```

```
# g 10;;
- : int = 31
```

```
# f 10 2;;
f unbound
```

Fattoriale con accumulatore

Titolo nota 24/09/2015

```
#let fact n = let rec f x a =
                  if x=0 then a
                  else f (x-1) (a*x)
              in
              f n 1 ;;
fact : int -> int = <fun>
```

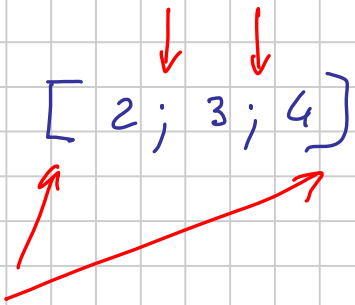
LISTE in CAML (sono la realtà delle PILE)

Titolo nota 24/09/2015

Si aggiunge in testa e si toglie in testa

Liste CAML sono sequenze di valori dello stesso tipo.

Notazione



tipo delle liste

Tipodei_Valori list

```
# [1;2;3];;
- : int list = [1;2;3]
```

```
# ['a';2];;
type error
```

Liste vuote

```
#[];;
-: 'a list = []
```

Operatori su liste

Titolo nota 24/09/2015

Esiste un operatore che si chiama "costruttore di valori" di liste

CONS e in CAML si indica con ::

CONS ha due argomenti:

- un valore v
- una lista l

dà come risultato la lista l con aggiunto il valore v in testa. Es:

```
# 3 :: [1;2];;
- : int list = [3;1;2]
```

```
# 'a' :: [1;2];;
errore di tipo
```