

- interprete Caml è un risolutore di espressioni
- let consente di definire nomi
- ambiente di associazioni nome-valore
- con let posso definire funzioni che finiscono nell'ambiente

# let f x = 2 \* x ;;

f : int → int = <fun>

INFERENZA  
DI TIPI

f	λx. (2 * x)
⋮	

AMB

- funzioni con più di un parametro

I modo :

USO UNA COPPIA (O UNA ENNELLA)

# let Somma (x, y) = x + y ;;

Somma : int \* int → int = <fun>

PRODOTTO CARTESIANO

# Somma (3, 5) ;;

- : int = 8

Somma	$\lambda(x, y) \cdot (x + y)$
⋮	

Somma (3, 5) AL MOMENTO  
DELLA CHIAMATA  
SOSTITUISCO

$\lambda(x, y) \cdot x + y = 3 + 5 = 8$

## II MODO

# Per Somma  $x \ y = x + y$  ;

Somma :  $\text{int} \rightarrow \text{int} \rightarrow \text{int} = \langle \text{fun} \rangle$

### APPLICAZIONE

# Somma 3 5 ;

- :  $\text{int} = 8$

Somma	$\lambda x. \lambda y. (x+y)$
⋮	



QUELLO CHE FA È  
QUESTO

$\lambda x. \lambda y. (x+y)$

PASSO 1

↓ IL PRIMO VALORE PASSATO È 3  
QUINDI LO SOSTITUISCO A X

$\lambda y. (3+y)$

PASSO 2

↓ ORA PRENDE IL SECONDO ARGOMENTO  
(IL 5) E LO SOSTITUISCE A Y

$(3+5)$

||  
8

Dopo il passo 1 ho ottenuto una

funzione che si aspetta un intero  
e da un intero come risultato

$\lambda y. (3+y)$  Tipo  $\text{int} \rightarrow \text{int}$

quindi la funzione somma ha come tipo

$\text{int} \rightarrow (\text{int} \rightarrow \text{int})$

$\uparrow$                        $\uparrow$                        $\uparrow$   
x                              y                              risultato

QUESTO MODO DI PASSARE GLI ARGOMENTI  
A UNA FUNZIONE VIENE DETTO PASSAGGIO

"Curryed"

└──┬──┘  
    ↓

NOME DI UN MATEMATICO CHE HA STUDIATO QUESTE COSE

QUESTO METODO CONSENTE DI FARE

"APPLICAZIONE PARZIALE DI FUNZIONE"

PASSO SOLO UNA PARTE DEGLI ARGOMENTI

ALLA FUNZIONE

# APPLICAZIONE PARZIALE DI FUNZIONE

# let  $Summa\ x\ y = x + y;;$

$Summa : int \rightarrow int \rightarrow int = \langle fun \rangle$

# let  $Summa3 = Summa\ 3;;$

$Summa3 : int \rightarrow int = \langle fun \rangle$

Passo  
UN SOLO  
ARGOMENTO  
A SUMMA

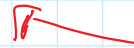
Summa	$\lambda x. \lambda y. (x+y)$
:	

Summa 3



3 viene sostituito a x

$\lambda y. (3+y)$



È LA FUNZIONE  
CHE PRENDE UN  
PARAMETRO E  
GLI SUMMA 3

Summa3	$\lambda y. (3+y)$
Summa	$\lambda x. \lambda y. (x+y)$
:	

# FUNZIONI POLIMORFE

La funzione identità

```
# let i x = x ;;
```

i : ???

INFERIAMO IL TIPO di i

1) i PRENDE UN PARAMETRO E QUINDI È UNA FUNZIONE

..... → .....

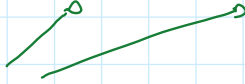
2) IL MODO IN CUI USIAMO IL PARAMETRO x IN i NON CI DICE NIENTE RIGUARDO AL SUO TIPO (NON LA SOMMIAMO, MOLTIPLICHIAMO, ECC...)

LA FUNZIONE i POTREBBE ESSERE APPLICATA AD ARGOMENTI DI QUALUNQUE TIPO (int, float, ...)

È UNA FUNZIONE POLIMORFA

PER ESPRIMERE QUESTA CARATTERISTICA L'INTERPRETE CAML USA UNA VARIABILE DI TIPO

$i: 'a \rightarrow 'a = \langle \text{fun} \rangle$

  
VARIABLE di  
TIPO 'a

- IL TIPO DELL'ARGOMENTO PUO' ESSERE QUALUNQUE
- IL RISULTATO AVRA' COME TIPO LO STESSO DELL'ARGOMENTO (STESSA VARIABLE DI TIPO)

## ALTRO ESEMPIO:

# let maggiore x y = x > y ;;

maggiore : 'a -> 'a -> bool = <fun>

prende due argomenti, di Tipo qualunque (ma lo stesso per entrambi) e restituisce un bool

# let negativo = maggiore 0 ;;

negativo : int -> bool = <fun>

applicazione parziale di maggiore

ha Tipo int -> bool (e non 'a -> bool)

perché applicando maggiore a 0

si specifica che 'a deve essere int!



# ANCORA UN ESEMPIO

```
# let first (x, y) = x;;
```

```
first : 'a * 'b -> 'a = (fun)
```

(analogo a  
fst  
gk - vito)



PRENDE UNA COPPIA DI  
VALORI CHE POSSONO  
AVERE TIPO DIVERSO

IL RISULTATO  
HA LO STESSO  
TIPO DEL PRIMO  
ELEMENTO DELLA  
COPPIA

(USA DUE VARIABILI DI TIPO  
'a E 'b PER DIRE  
QUESTO)

```
# first (10, 4);;
```

```
- : int = 10;
```

← 'a e 'b  
sono  
entrambi int

```
# first (10, 4.3);;
```

```
- : int = 10;
```

← 'a = int  
'b = float

## PASSAGGIO DI FUNZIONI A FUNZIONI

Abbiamo visto che in Camel le funzioni sono valori (possono, ad esempio, essere associate a un nome e memorizzate nell'ambiente assieme ai valori interi, float, ecc..)

POSSIAMO PASSARE UNA FUNZIONE COME ARGOMENTO AD UN'ALTRA FUNZIONE

ESEMPIO:

# let apply f x = f x ;;

apply : ('a -> 'b) -> 'a -> 'b = <fun>

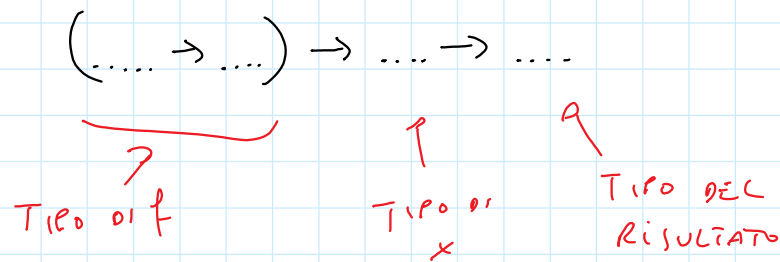
La funzione apply prevede due parametri f e x. Quindi, il suo tipo sarà

..... -> ..... -> .....

↖      ↖      ↖  
TIPO DI f      TIPO DI x      TIPO DEL RISULTATO

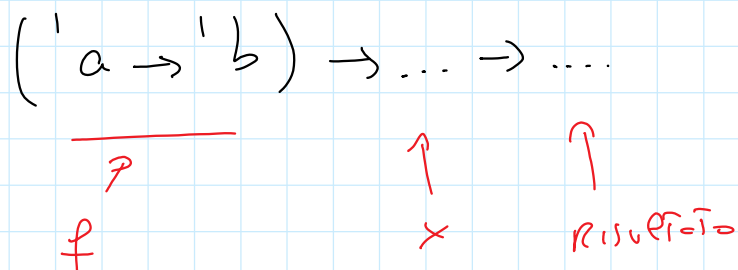
Guardando il corpo di apply vediamo che f è applicato ad x (cioè f viene richiamata passando x come argomento)

quindi  $f$  è una funzione (ha Tipo  $\dots \rightarrow \dots$ )

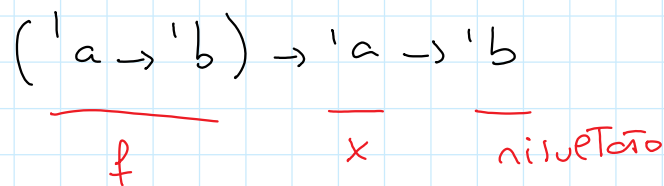


Il corpo di `apply` non fa niente altro, quindi non possiamo dedurre altro riguardo  $f$ . Il suo tipo sarà genericamente

$'a \rightarrow 'b$  (il risultato potrebbe avere tipo diverso rispetto al parametro)



ma se  $f$  ha Tipo  $'a \rightarrow 'b$ , allora  $x$  ha Tipo  $'a$  e il risultato di `apply` avrà il tipo del risultato di  $f$



Quindi `apply` prende una funzione qualunque e lo applica ...

... -

# let succ  $x = x + 1$ ;;

succ : int  $\rightarrow$  int = (fun)

# apply succ 10;;

- : int = 11

# let doppio  $x = 2 * x$ ;;

doppio : int  $\rightarrow$  int = (fun)

# apply doppio 10;;

- : int = 20

## PER ESERCIZIO

Definire le seguenti funzioni cercando di intuire il tipo che sarà inferito dall'interprete Caml

FATE ANCHE UN PO' DI PROVE CON LE VARIE FUNZIONI!!

- 1) funzione che prende un intero e restituisce la coppia formata dal numero stesso e dal successivo
- 2) funzione che prende una coppia di valori di qualunque tipo (anche diversi tra loro) e restituisce la coppia in cui i due elementi hanno posizione scambiata
- 3) funzione `double_apply` che prende una funzione  $f$  e un argomento  $x$  e applica due volte la  $f$  ad  $x$  (ossia calcola  $f^2(x)$ )
- 4) funzione `pair_apply` che prende una funzione  $f$  e una coppia

e restituisce la coppia ottenuta applicando  $f$  ad ognuno degli elementi della coppia ricevuta

# DEFINIZIONE di FUNZIONI PER CASI

calcolo del valore assoluto

$$\text{abs}(x) = \begin{cases} x & \text{se } x \geq 0 \\ -x & \text{se } x < 0 \end{cases}$$

→ USO UNA ESPRESSIONE CONDIZIONALE IF

→ NON UN COMANDO!!

if CONDIZIONE Then ESPRESSIONE else ESPRESSIONE

↑  
↑  
CI VOLLONO  
ENTRambi  
SEMPRE!!

# Let abs x = if x >= 0 Then x  
else -x ;;

abs : int → int = (fun)

↑  
IMPLICA  
CHE x ∈ int

↑  
↑  
QUESTE  
DUE  
ESPRESSIONI  
DEVONO  
AVERE LO  
STESSO  
TIPO!

## ALTRO ESEMPIO

massimo tra due valori

# let max x y = if x > y then x  
else y ;

max : 'a -> 'a -> 'a = <fun>



# DEFINIZIONI LOCALI (di nomi ALL'INTERNO di FUNZIONI)

$$\text{areaCerchio}(r) = r^2 \cdot \pi \quad \text{dove } \underline{\underline{\pi = 3,14}}$$

#let areaCerchio r =

let pi = 3.14

in

r \*. r \*. pi ;;

areaCerchio : float → float = <fun>

- IL NOME pi È LOCALE AD areaCerchio.
- POTRÀ ESSERE USATO SOLO NEL SUO CORPO
- PI NON SARÀ INSERITO NELL'AMBIENTE

UN LET ALL'INTERNO DI UN ALTRO LET PUO' ESSERE USATO PER DEFINIRE UNA FUNZIONE AUSILIARIA LOCALE (UTILIZZABILE SOLO NELL'AMBITO DEL LET PIU' ESTERNO)

```
# let f x =
  let g z = 2 * z
  in
  (g x) + 1;;
```

$f : \text{int} \rightarrow \text{int} = \langle \text{fun} \rangle$

e' come dire :

$$f(x) = g(x) + 1 \quad \text{dove} \quad g(z) = 2z$$

g e' locale ad f. Non viene creata un'associazione nell'ambiente per g.!!

```
# f 10;;
- : int = 21
```

# g h<sub>i</sub>

ERRORE: g non definita

## ALTRI ESERCIZI (segue)

- 5) funzione **non\_neg** che prende una funzione  $f$  e un valore  $x$  e restituisce il massimo  $\max(f(x), 0)$ .
- Provare ad applicare parzialmente questa funzione. Che cosa si ottiene?
- 6) funzione **best** che prende due funzioni  $f$  e  $g$  e un valore  $x$  e restituisce il massimo  $\max(f(x), g(x))$ .
- Provare ad applicare questa funzione senza passare  $x$ . Che cosa si ottiene?
- 7) funzione **ordina** che prende una tupla di 3 valori e restituisce la tupla degli stessi 3 valori messi in ordine crescente.
- 8) funzione **soluzioni** che prende 3 valori  $a$ ,  $b$  e  $c$  che rappresentano i coefficienti di una equazione di secondo grado

$$ax^2 + bx + c = 0$$

e restituisce una coppia in cui il primo elemento è un bool che dice se l'equazione ha soluzioni (discriminante  $\geq 0$ ) e il secondo è a sua volta una coppia con le due soluzioni (possibilmente coincidenti) dell'equazione o con (0,0) se non ci sono soluzioni reali.

Definire un nome locale per memorizzare il discriminante dell'equazione.

Può anche essere conveniente memorizzare le due soluzioni definendo nomi locali ma solo dopo aver controllato che il discriminante sia  $\geq 0$ .

NOTA: LA RADICE QUADRATA SI FA CON LA FUNZIONE PREDEFINITA `sqrt`

### ESEMPI D'USO

- 1)  $a=1.0$   $b=4.0$   $c=3.0 \Rightarrow (\text{True}, (-3.0, -1.0))$
- 2)  $a=2.0$   $b=4.0$   $c=2.0 \Rightarrow (\text{True}, (-1.0, -1.0))$
- 3)  $a=2.0$   $b=2.0$   $c=2.0 \Rightarrow (\text{False}, (0.0, 0.0))$