Tipi di Dato in SML

Luca Abeni, Csaba Kiraly

April 20, 2017

Tipi Predefiniti

- Tipi di dato predefiniti di ML: `bool`, `int`, `real`, `char`, `string` e `unit`
- A partire da questi, tipi aggregati / composti / strutturati:

```
val sq = fn (a,b) => a * a + b * b;
```

ha tipo

```
int * int -> int
```

- Dominio: `int * int`
  - prodotto cartesiano dell'insieme dei valori di `int` per se stesso (matematicamente, $\mathcal{Z} \times \mathcal{Z}$)
- Prodotti cartesiani di tipi più semplici: tuple

Tuple

- Definite tramite prodotto cartesiano
- `unit`: tuple di 0 elementi
- Tuple di 1 elemento corrispondano con l'elemento stesso:

```
> (6);
val it = 6 : int
> (6) = 6;
val it = true : bool
```

Tuple & Pattern Matching

- Su tuple, spesso pattern matching

```
> val c = ("pi greco" , 3.14);
val c = ("pi greco", 3.14) : string * real
> val (pigreco , pi) = coppia;
val pi = 3.14 : real
val pigreco = "pi greco" : string
```

- Usato per "spacchettare" elementi di una tupla

Funzioni come Tipo

- Funzione: valore memorizzabile
- Il tipo di un valore di tipo funzione è *dominio*$\rightarrow$*codominio*:

```
int * int -> int
```

- Abbiamo visto anche esempi più complessi

```
(int * int) -> int -> int
```

  - Funzione che riceve come argomenti una coppia di interi ed un intero e ritorna un intero (vedi Curry)

Nuovi Nomi per Tipi Esistenti...

- Standard ML permette di definire dei sinonimi per tipi di dato esistenti, usando il costrutto "`type`"

```
> type time_t = int;
type time_t
> val a:time_t = 5;
val a = 5 : time_t
> val b:time_t = 1;
val b = 1 : time_t
> a + b;
val it = 6 : time_t
```

Ma non Nuovi Tipi!

- `time_t` non è un nuovo tipo, ma solo un sinonimo per `int`:

```
> val c = 3;
val c = 3 : int
> a + c;
val it = 8 : time_t
```

Datatype

- "`type`" associa solo un nuovo nome ad un tipo esistente
- "`datatype`" definisce nuovi tipi di dato:

```
datatype col = red | blue | green;
```

- Il pattern matching funziona anche con questi tipi

```
fun translate red = "rot"
  | translate blue = "blau"
  | translate green = "gruen";
```

```
> translate green;
val it = "gruen" : string
```