DP

Recap di let e pattern matching

Che cos'è una funzione?

- Nome “**funzione**” introdotto da *Leibinz* nel 1673, in ambito analitico, poi chiarito da *Bernoulli* e *Euler* come “ogni espressione fatta di variabili e costanti”
- *Hardy* 1908: una funzione è una ***relazione*** *tra x* ed *y* che "*to some values of x at any rate corresponds values of y*."
- *Bourbaki* 1939: “Let E and F be two sets, which may or may not be distinct. A relation between a variable element x of E and a variable element y of F is called a *functional relation* in y if, **for all** $x \in E$, **there exists a unique** $y \in F$ which is in the given relation with x.”
- Def *estensionali*. Per la nozione computazione: λ-calcolo

Espressioni

- Modello di computazione = *valutazione di espressioni*
- Ogni *espressione*:
  - ha o non ha un **tipo**
  - ha o non ha un **valore** (non-terminazione/run time error)
  - può generare un **effetto** (I/O, eccezioni etc)

Let

- Lego un valore "val" a una variabile "id", notazione "id $\rightarrow$ val" ("*binding*")
  - `let id = exp`
- Un "*enviroment*" (ambiente) è un insieme di binding
  - Più esattamente, ambiente è una **funzione parziale finita** tra *id* e *val*
  - La si può visualizzare come una lista *snoc* (che si estende a destra) con questa BNF
    - η ::= . | η , id $\rightarrow$ val

Mantra sul let

- *Let* **non** è assegnamento
- Le variabili non variano, sono cioè per default **immutabili**
- Le variabili hanno un ambito ("*scope*") in cui hanno senso
- Se ri-lego un valore a una variable, vale il legame più recente ("*shadowing")*

Pattern matching

- PM su expressioni con tipi *primitivi*, es interi:

```
match n with
    | 0 -> e1
    | m -> e2
```

- PM su espressioni *complesse*:

```
let and (x,y)  =
  match (x,y) with
    | (true, true) -> true
    | _ -> false
match (n % 2) with 0 → e1 | 1 → e2
```

Pattern matching cont.

- PM è **cronologico** – ordine conta
- PM dovrebbe essere:
  - *Esaustivo*: pattern coprono ogni possibilità
  - *Disgiunto*: non vi sono pattern con overlap
    - Interprete segnala un warning
- In un ramo di PM `p → e,` le variabili che occorrono in `p` sono **vincolate** e come tali
  - Il loro nome non conta, ma meglio siano differenti da altre vars
  - Entrano a far parte dell'enviroment locale
  - Non possono avere ripetizioni, e.g. (x,x) non è pattern valido

PM & let

- Più generalmente una espressione *let* prende non solo un *id*, ma un **pattern**, di cui *id* è una istanza
  - `let pat = exp1 in exp2`
- **let** è definito in termini di **match**

```
let (x,y) = (1,2) in x + y //equivalente
match (1,2) with (x,y) -> x + y
```

- Grammatica dei pattern

```
p::= c | x | _ |(pat1,pat2)| … (to be continued)
```

Ricorsione

- Potere anche leggere le slides di ripasso sulla ricorsione (prese dal libro di testo)
  - File ric-recap.pdf