

# Linguaggi Funzionali 2005/2006

Dipartimento di Informatica, Verona

Fausto Spoto

`fausto.spoto@univr.it`

ricevimento: mercoledì 14:30-16:30

13 gennaio 2005

# Orario delle lezioni

**gennaio** 12 (8:30-10:15), 13 (14:30-15:15), 14 (11:30-13:15), 20 (14:30-15:15), 21 (11:30-13:15), 26 (8:30-10:15), 27 (14:30-15:15), 28 (11:30-13:15)

**febbraio** 3 (14:30-15:15), 4 (11:30-13:15), 9 (8:30-10:15), 10 (14:30-15:15), 11 (11:30-13:15), 16 (8:30-10:15), 17 (14:30-15:15), 18 (11:30-13:15), 23 (8:30-10:15), 24 (14:30-16:15), 25 (11:30-13:15)

**marzo** 4 (11:30-13:15), 9 (8:30-10:15), 10 (14:30-15:15), 11 (11:30-13:15)

Mancano 6 ore!

- Xavier Leroy: *The Objective Caml System*
- Il compilatore OCaml
- Chris Okasaki: *Purely Functional Data Structures*, Cambridge University Press
- Questi stessi lucidi!

## Dove?

Tutto (tranne Okasaki) disponibile all'indirizzo  
`profs.sci.univr.it/~spoto/teaching.html`

- La prima parte è un **progetto** che verrà definito verso la fine del corso. Lo scopo è di dimostrare di aver capito come si scrive del codice in programmazione funzionale
- La seconda parte è un **orale** che intende verificare l'acquisizione delle conoscenze pratiche e teoriche fornite dal corso
- Le due parti sono indipendenti e possono essere sostenute in momenti diversi
- Non esistono veri e propri appelli d'esame. Ci si mette d'accordo via email su una data che conviene a studente e docente

# Cenni storici: il $\lambda$ -calcolo di Church (1932-41)

$\lambda$ -astrazione

$\lambda x.x + 1$

$\beta$ -riduzione

$(\lambda x.x + 1)3 \xrightarrow{\beta} 4$

Tesi di Church

$f$  numerica è calcolabile  $\Leftrightarrow f$  è definibile in  $\lambda$ -calcolo

In effetti...

Church implementò naturali, liste, condizionali e ricorsione dentro il  $\lambda$ -calcolo

In effetti...

**Kleene** (1936) dimostrò che le funzioni ricorsive alla Gödel sono tutte e sole quelle definibili in  $\lambda$ -calcolo

In effetti...

**Turing** (1937) dimostrò che le funzioni calcolabili sulla macchina di Turing sono tutte e sole quelle definibili in  $\lambda$ -calcolo

## McCarthy definisce il LISP Processor (LISP, 1950)

*"... it seemed natural to use the  $\lambda$ -notation of Church, but I didn't understand the rest of his book..."*

### Cosa aveva il LISP di speciale?

- Gestione dinamica della memoria tramite un **garbage collector**
- Uso di **chiusure** (termine + ambiente) per rappresentare i  $\lambda$ -termini
- Uso della **valutazione lazy**
- Uso di **funzioni di ordine superiore** nelle mappature
- Sintassi criptica (mancavano gli studi sui compilatori)

## E Gordon e Milner il *Meta Language* (ML, 1978)

Nasce come linguaggio di specifica per strategie di prova. Poi si accorgono che poteva servire a programmare

### Cosa aveva ML di speciale?

- **Fortemente tipato**
  - i tipi classificano le espressioni
  - il compilatore può escludere alcuni errori dinamici
- **Inferenza di tipi**
  - il compilatore deduce i tipi delle espressioni
  - il programmatore non specifica i tipi delle espressioni
- **Polimorfismo parametrico dei tipi**
- **Tipi algebrici**
- **Pattern matching**
- **Valutazione stretta** (eager, non lazy)

# Migliorie o babele?

**Turner** definisce Miranda (1985)

Essenzialmente ML con valutazione lazy

Un **comitato internazionale** definisce Haskell (1987)

Un tentativo di standardizzare il linguaggio ML con valutazione lazy. In onore di Haskell Curry

**Xavier Leroy** definisce Caml (1993) e OCaml (2000)

Versione francese dell'ML. OCaml include anche gli oggetti. Macchina virtuale estremamente efficiente

Ma anche...

Standard ML, Curry, Scheme, ELISP, CLOS...

E secondo i suoi autori anche...

C (nacque come funzionale, poi è cresciuto male)