

# Linguaggi Funzionali 2006/2007

Dipartimento di Informatica, Verona

Fausto Spoto

`fausto.spoto@univr.it`

ricevimento: mercoledì 14:30-16:30

2 ottobre 2006

# Orario delle lezioni: i venerdì in laboratorio

ottobre 3 (9:40-11:30), 4 (10:40-11:30),  
5 (14:40-16:30), 10 (9:40-11:30), 11 (10:40-11:30),  
12 (14:40-16:30), 17 (9:40-11:30), 18 (10:40-11:30),  
19 (14:40-16:30), 24 (9:40-11:30), 25 (10:40-11:30),  
26 (14:40-16:30), 31 (9:40-11:30)

novembre 2 (14:40-16:30), 7 (9:40-11:30), 8 (10:40-11:30),  
9 (14:40-16:30), 14 (9:40-11:30), 15 (10:40-11:30),  
16 (14:40-16:30), 21 (9:40-11:30), 22 (10:40-11:30),  
23 (14:40-16:30)

- Guy Cousineau, Michel Mauny: *The Functional Approach to Programming*, Cambridge University Press
- Xavier Leroy: *The Objective Caml System*
- Il compilatore OCaml
- Chris Okasaki: *Purely Functional Data Structures*, Cambridge University Press
- Questi stessi lucidi!

## Riferimento internet:

[profs.sci.univr.it/~spoto/teaching.html](http://profs.sci.univr.it/~spoto/teaching.html)

Consiste in una prova al computer in cui si devono realizzare o modificare dei programmi.

# 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)