

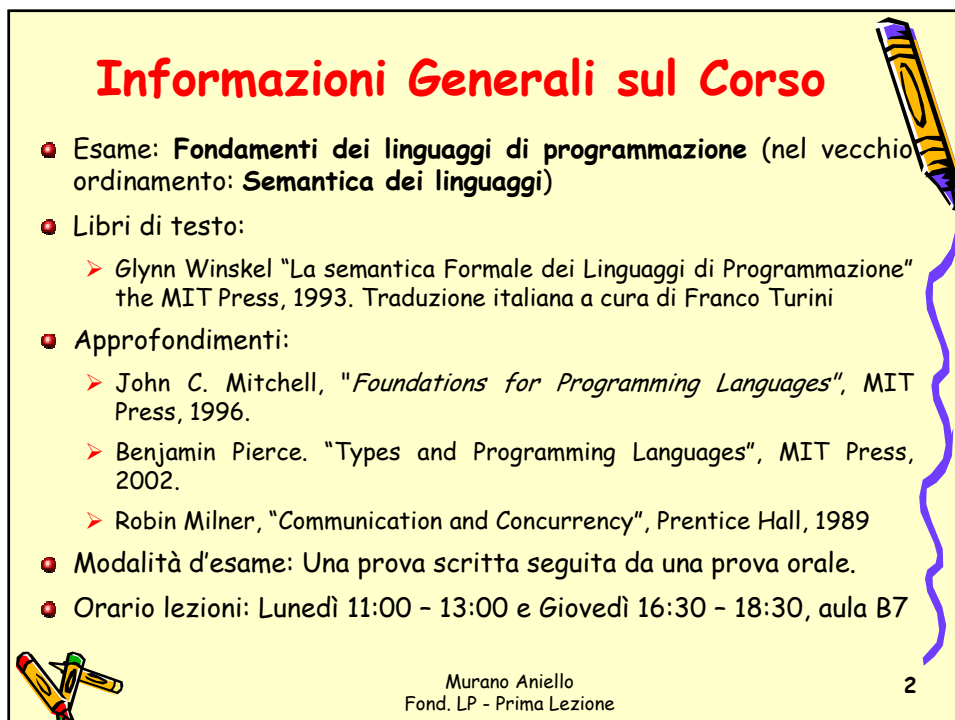


Fondamenti dei linguaggi di programmazione

Aniello Murano
Università degli Studi di Napoli
"Federico II"

Murano Aniello
Fond. LP - Prima Lezione

1



Informazioni Generali sul Corso

- **Esame: Fondamenti dei linguaggi di programmazione** (nel vecchio ordinamento: **Semantica dei linguaggi**)
- **Libri di testo:**
 - Glynn Winskel "La semantica Formale dei Linguaggi di Programmazione" the MIT Press, 1993. Traduzione italiana a cura di Franco Turini
- **Approfondimenti:**
 - John C. Mitchell, "*Foundations for Programming Languages*", MIT Press, 1996.
 - Benjamin Pierce. "Types and Programming Languages", MIT Press, 2002.
 - Robin Milner, "Communication and Concurrency", Prentice Hall, 1989
- **Modalità d'esame:** Una prova scritta seguita da una prova orale.
- **Orario lezioni:** Lunedì 11:00 - 13:00 e Giovedì 16:30 - 18:30, aula B7

Murano Aniello
Fond. LP - Prima Lezione

2

Informazioni sul Docente

- Prof. Dr. Aniello Murano, ricercatore universitario presso la Sezione di Informatica del Dipartimento di Fisica - Università degli Studi di Napoli, Federico II
- Sito web: <http://people.na.infn.it/~murano/>
- Ricevimento: Fino al 28 Ottobre, Venerdì 14:00 - 16:00, 2Ma10
- E-mail: murano@na.infn.it (da usarsi solo in casi di effettiva necessità)
- Altre informazioni sul docente:
- Laurea in Scienze dell'informazione e Dottorato di ricerca in Informatica presso l'Università degli Studi di Salerno. Visiting Student per un anno presso la Rice University di Houston(Texas - USA). Post-doc per un anno presso la Hebrew University di Gerusalemme (Israele)
- Altri insegnamenti: Lab. di ASD - Corso di Laurea in Informatica, Facoltà di Scienze MM.FF.NN. - Università di Napoli "Federico II".
- Interessi di ricerca: Teoria degli Automi e dei Linguaggi Formali. Logiche Temporal discrete e real-time, Metodi Formali per la Specifica, la Verifica e la Sintesi di sistemi hardware e software, Model Checking, Teoria dei Giochi.



Murano Aniello
Fond. LP - Prima Lezione

3

Obiettivi del Corso

- L'obiettivo del corso è quello di fornire i modelli formali necessari per capire il comportamento di un programma e ragionare su di esso.
- Vengono dunque presentate le nozioni matematiche, le tecniche ed i concetti sui quali si fonda la semantica formale dei linguaggi di programmazione.



Murano Aniello
Fond. LP - Prima Lezione

4

Che cosa questo corso non è

- Un corso su un particolare linguaggio di programmazione
- Un corso sulla costruzione di un particolare compilatore o sulla costruzione di un parser
 - Si presuppone la conoscenza da parte degli studenti del concetto di grammatiche
- Un corso per confrontare i vari linguaggi di programmazione
- Un corso sui vari paradigmi di programmazione (funzionali, orientati agli oggetti, logici, concorrenti, ecc.)



Murano Aniello
Fond. LP - Prima Lezione

5

Programma del corso (unofficial)

- Semantica operativa del linguaggio imperativo IMP (Cap. II, pag 15-32);
- Induzione ben fondata (Cap. III-IV, pag 33-60);
- Ordinamenti parziali completi, funzioni continue e teorema di punto fisso (Cap. V, pag 79-83);
- Semantica denotazionale del linguaggio imperativo IMP e corrispondenza con la semantica operativa (Cap. V, pag 65-78);
- Linguaggi con tipi di ordine superiore: tipizzazione, semantica operativa eager e lazy, semantica denotazionale lazy (Cap. XI, pag 201-209 e pag. 219-222);
- Nondeterminismo e parallelismo: i linguaggio CSP e CCS (Cap. XIV, pag 321-341);
- Nozioni di equivalenza per CCS: definizione di LTS (Labelled Transition System), equivalenza a tracce, simulazione, bisimulazione (forte e debole).



Murano Aniello
Fond. LP - Prima Lezione

6

Prerequisiti al corso

- Background in linguaggi di programmazione
 - Esperienza di programmazione in differenti linguaggi di programmazione (per esempio C, Java, Lisp, Prolog, ect.)
- Background in logica matematica
 - Abilità nello scrivere e leggere prove formali (per esempio, dimostrazioni per induzione)
 - Una certa predisposizione ai "ragionamenti formali".



Murano Aniello
Fond. LP - Prima Lezione

7

Lo stato dell'arte

- Quello dei linguaggi di programmazione è senza dubbio uno dei campi più vecchi, fondamentali e ferventi dell'informatica
- Ancora oggi, i linguaggi di programmazione sono in continua evoluzione
- I linguaggi di programmazione sono solitamente adottati per riempire dei vuoti:
 - Per implementare un'applicazione che prima era difficile o impossibile da implementare
 - Per eseguire programmi su una nuova piattaforma
- Istruire nuovi programmatori comporta costi notevoli
 - I linguaggi più largamente utilizzati sono raramente sostituiti
 - I linguaggi più popolari con il passare del tempo si arrugginiscono
 - Al contrario, nuove applicazioni e nuove piattaforme rappresentano interessanti opportunità per innovazioni.



Murano Aniello
Fond. LP - Prima Lezione

8

Perché ci sono così tanti linguaggi

- Molti linguaggi furono costruiti per specifiche applicazioni
- I domini di applicazione hanno solitamente bisogni differenti (e conflittuali)
- Per esempio:
 - Intelligenza Artificiale: computazione simbolica (Lisp, Prolog)
 - Computazione Scientifica: grande performance (Fortran)
 - Computazione Commerciale: generazione di report (Cobol)
 - Programmazione di Sistemi: accesso a basso livello (C)
 - Customizzazione: scripting (Perl, Javascript)
 - Sistemi distribuiti: computazione mobile (Java, C++)
 - e altri (Latex, AWK, Q, Hancock,...)



Murano Aniello
Fond. LP - Prima Lezione

9

Paradigmi di Programmazione

- Imperativo, procedurale
 - Fortran, Algol, Cobol, C, Pascal
- Funzionali (principalmente liberi da assegnamento)
 - Lisp, Scheme, ML, Haskell
- Orientato agli oggetti
 - C++, Java, Visual Basic, Simula, Smalltalk, Eiffel, Self
- Logici
 - Prolog, lambda-Prolog
- Concorrenti
 - Fortran90



Murano Aniello
Fond. LP - Prima Lezione

10

Cosa rende efficiente un linguaggio?

- Una definizione sulla misura della scienza di Lord Kelvin (1824-1907):
- ..."quando puoi misurare ciò di cui stai parlando e lo puoi esprimere in numeri, tu conosci qualcosa di ciò, ma quando non puoi esprimerlo in numeri, la tua conoscenza è povera e insoddisfacente; quello potrebbe essere l'inizio della conoscenza, ma si è lontani dall'aver elevato quel concetto allo stadio della scienza".
- Semplicità (nella sintassi e nella semantica)
- Facilità di lettura e scrittura
- Familiarità
- Sicurezza
- Indipendenza dalla macchina
- Efficienza (di compilazione ed esecuzione)



Efficienza di un linguaggio

- Spesso, gli obiettivi precedenti sono in conflitto tra di loro:
 - I controlli di sicurezza costano in termini di tempo di compilazione o esecuzione
 - Sicurezza e indipendenza dalla macchina potrebbero vietare operazioni efficienti di basso livello.
 - Alcuni tipi di sistema possono restringere lo stile di programmazione negli scambi di informazione per rafforzare le garanzie di sicurezza
- Solitamente i compromessi sono difficili da trovare
- Dunque, progettare un buon linguaggio di programmazione è davvero difficile!



Valutare le caratteristiche di un linguaggio

- Un concetto o un costrutto di un linguaggio può essere studiato nel contesto di un reale e quindi di un grande linguaggio.
- Spesso però è più semplice studiare una caratteristica di un linguaggio studiando un linguaggio **minimo** che ingloba quella caratteristica. Questo permette
 - Una Sperimentazione semplificata dell'intero linguaggio
 - Una più profonda e precisa comprensione delle caratteristiche del linguaggio



Un approccio formale di Specifica

- Un approccio formale nella progettazione di un linguaggio consiste nella sua definizione rigorosa tramite una sintassi e una semantica
- Esistono tre approcci alla semantica dei linguaggi:
- Semantica operativa
 - Definisce la macchina astratta che interpreta i programmi
 - Utile per implementare un compilatore o un interprete
 - Di basso livello, ma flessibile
- Semantica Denotazionale
 - Assegna denotazioni, o significati, ai programmi
 - Usa oggetti matematici per descrivere il comportamento dell'input e dell'output
- Semantica assiomatica
 - Usa regole logiche per ragionare sul comportamento dei programmi
 - Dato un programma, degli stati iniziali e dei dati di input, fornisce l'insieme degli stati finali raggiungibili dopo l'esecuzione
 - Utile per provare la correttezza di un programma





Break Time

Murano Aniello
Fond. LP - Prima Lezione

15



Un semplice linguaggio di Espressioni Aritmetiche

Sintassi e Semantica operativa

Murano Aniello
Fond. LP - Prima Lezione

16

Sintassi di ARITHM

- Sintassi concreta: Le regole che permettono di esprimere i programmi come stringhe di caratteri
 - Utile per gli obiettivi di progettazione
 - Familiarità, leggibilità, velocità di compilazione
- Gli stessi concetti possono essere espressi in molte sintassi concrete, non tutte ugualmente efficienti
- Esistono comunque solidi principi per le sintassi concrete
 - Automi finiti e grammatiche context-free
 - Generatori di parser automatici



Sintassi astratta

- Noi definiamo la sintassi del linguaggio ARITH rispetto ad un albero di sintassi astratto (parse tree).
- L'albero di sintassi astratto è indipendente dalla sintassi concreta del linguaggio e meglio si adatta a manipolazioni formali e algoritmiche
- Una sintassi astratta per ARITH è

$e ::= n$	dove n è un letterale intero ($n \in \{0,1,2,\dots\}$)
$ e_1 + e_2$	somma
$ e_1 * e_2$	moltiplicazione

- Dove " $::=$ " significa "può essere" e " $|$ " significa "oppure"



Analisi di ARITHM

- Questioni da risolvere:
- Qual è il significato di una data espressione di ARITH?
- Come valutare una data espressione di ARITH?
- Come sono collegati valutatore e significato di una espressione?



Murano Aniello
Fond. LP - Prima Lezione

19

Una semantica Operazionale

- Specifica come le espressioni dovrebbero essere valutate
- È definita in base alla particolare forma dell'espressione valutata:
 - n si valuta n
 - ✓ n è una forma normale e non necessita di essere valutata ulteriormente
 - $e_1 + e_2$ si valuta n se
 - ✓ e_1 si valuta n_1
 - ✓ e_2 si valuta n_2
 - ✓ ed n è la somma di n_1 e n_2
 - $e_1 * e_2$ si valuta n se
 - ✓ e_1 si valuta n_1
 - ✓ e_2 si valuta n_2
 - ✓ ed n è il prodotto di n_1 e n_2



Murano Aniello
Fond. LP - Prima Lezione

20

Formulazione Alternativa

- Notazione: $e \Downarrow n$ significa che "e" si valuta in "n"
- Questa è una valutazione, un'affermazione sulla relazione tra "e" ed "n".
- Questa notazione permette di riscrivere le regole precedenti in modo più conciso
- $n \Downarrow n$
- $e_1 + e_2 \Downarrow n$ se
 - $e_1 \Downarrow n_1$
 - $e_2 \Downarrow n_2$
 - ed n è la somma di n_1 ed n_2
- $e_1 * e_2 \Downarrow n$ se
 - $e_1 \Downarrow n_1$
 - $e_2 \Downarrow n_2$
 - ed n è il prodotto di n_1 ed n_2



Murano Aniello
Fond. LP - Prima Lezione

21

Semantica operativa come regole di inferenza

$$\frac{}{n \Downarrow n}$$
$$\frac{e_1 \Downarrow n_1 \quad e_2 \Downarrow n_2 \quad n \text{ è la somma di } n_1 \text{ e } n_2}{e_1 + e_2 \Downarrow n}$$
$$\frac{e_1 \Downarrow n_1 \quad e_2 \Downarrow n_2 \quad n \text{ è il prodotto di } n_1 \text{ e } n_2}{e_1 * e_2 \Downarrow n}$$

- Interpretazione: "sopra la linea" implica "sotto la linea"



Murano Aniello
Fond. LP - Prima Lezione

22

Esercizio

- $(3+5) \cdot (4+2)$ in cosa si valuta?



Murano Aniello
Fond. LP - Prima Lezione

23

