

# Indice

<b>1</b>	<b>Introduzione</b>	<b>2</b>
1.1	Motivazione . . . . .	2
1.2	Definizioni base . . . . .	2
1.3	Contenuti del corso . . . . .	3
1.4	Informazioni utili . . . . .	3
<b>2</b>	<b>Linguaggi regolari</b>	<b>5</b>

# 1 Introduzione

## 1.1 Motivazione

Un linguaggio è uno strumento per descrivere come risolvere i problemi in maniera rigorosa, in modo tale che sia eseguibile da un calcolatore Perché è utile studiare come creare un linguaggio di programmazione?

- non rimanere degli utilizzatori passivi
- capire il funzionamento dietro le quinte di un linguaggio
- domain-specific language (DSL): è un linguaggio pensato per uno specifico problema
- model driven software development: modo complesso per dire UML e simili
- model checking

## 1.2 Definizioni base

Un linguaggio è composto da:

- lessico e sintassi
- compilatore: parser + generatore di codice oggetto

La generazione automatica di codice può essere dichiarativa lessico (espressioni regolari o automa a stati finite) o sintassi(grammatiche o automa a pile). Un automa a stati finiti consuma informazioni una alla volta, ne salva una quantità finita. Alcuni esempi di applicazione di automa a stati finiti: software di progettazione di circuiti, analizzatore lessicale, ricerca di parole sul web e protocolli di comunicazione.

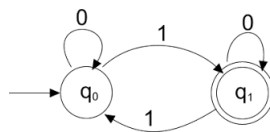


Figura 1: Semplice automa

### 1.3 Contenuti del corso

- Linguaggi formali e Automi:
  - Automi a stati finiti, espressioni regolari, grammatiche libere, automi a pila, Macchine di Turing, calcolabilità
- Compilatori:
  - Analisi lessicale, analisi sintattica, analisi semantica, generazione di codice
- Logica di base:
  - Logica delle proposizioni e dei predicati
- Modelli computazionali:
  - Specifica di sistemi tramite sistemi di transizione, logiche temporali per la specifica e verifica di proprietà dei sistemi (model checking), sistemi concorrenti (algebre di processi e reti di Petri)

### 1.4 Informazioni utili

Parte integrante del corso:

- Supporto alla parte teorica usando tool specifici.
  - JFLAP 7.1: <http://www.jflap.org> (automi/grammatiche)
  - Tina 3.7.5: <http://projects.laas.fr/tina> (model checking di sistemi di transizione e reti di Petri)
  - LTSA 3.0: <http://www.doc.ic.ac.uk/ltsa> (sistemi di transizione definiti tramite algebre di processi)
- Nel resto del corso utilizzeremo un ambiente di sviluppo per generare parser/compileri
  - IntelliJ esteso con plug-in ANTLRv4, ultima versione 1.20 (generatore ANTLR: <http://www.antlr.org/>)

Libri di testo suggeriti:

- J. E. Hopcroft, R. Motwani e J. D. Ullman: Automi, linguaggi e calcolabilit , Addison-Wesley, Terza Edizione, 2009. Cap. 1–9
- A. V. Aho, M. S. Lam, R. Sethi e J. D. Ullman: Compilatori: principi tecniche e strumenti, Addison Wesley, Seconda Edizione, 2009. Cap. 1–5
- M. Huth e M. Ryan: Logic in Computer Science: Modelling and Reasoning about Systems, Cambridge University Press, Second Edition, 2004. Cap. 1–3

## 2 Linguaggi regolari