UNIVERSITÀ
DEGLI STUDI
DI TORINO

# *Linguaggi Formali e Traduttori*

**a.a. 2018-2019**

## Cosa significa e come si studia

*Linguaggi*: successioni di parole su un dato alfabeto secondo certe regole
*Formali*: definiti in modo rigoroso con metodi logico-matematici
*Traduttori*: di un linguaggio in un altro conservandone il significato

Argomento da sempre considerato fondamentale in Computer Science

Perché : - aspetti metodologici (importanza di sviluppare concetti in modo astratto)
- importanti applicazioni

Tre parti: automi e espressioni regolari (facile in parte noto)
grammatiche e parsing (un pò meno usuale)
traduzione diretta dalla sintassi (dipende fortemente dalle prime due)
Segue che: è *importante* almeno seguire lo sviluppo del programma.

Struttura del corso: - lezioni frontali (48 ore)
- laboratorio (30 ore)
- esercitazioni e tutorato (12 ore)
- consulenza su appuntamento

Prerequisiti: i contenuti di Programmazione I e II e Logica

# Modalità d'esame

L'esame consiste in una prova scritta ed in una discussione orale del progetto di laboratorio.

La prova scritta è suddivisa in due parti,
1. linguaggi formali (automi, espressioni regolari, grammatiche)
2. parsificazione e la traduzione.

In entrambe le parti sono presenti sia domande di studio sia esercizi.
La media dei voti riportati nelle due parti, nel caso in cui siano entrambe sufficienti, fornirà il voto dello scritto.

Il superamento dello scritto permette di accedere a una delle prove di laboratorio previste per la sessione in cui lo scritto è stato superato. Nel caso in cui questa seconda prova non venga superata, lo scritto dovrà essere ripetuto.
Lo studente ha a disposizione tre prove scritte per ogni anno accademico.
E` possibile ritirarsi da una prova scritta a partire da un'ora dopo il suo inizio.

Il progetto di laboratorio può essere svolto individualmente o in gruppi formati da al massimo 3 studenti. La discussione della prova di laboratorio è sempre individuale.

# Modalità d’esame

Il voto d’esame viene calcolato come media pesata sul numero di crediti tra il voto dello scritto e il voto di laboratorio.
Per modificare il voto così ottenuto, purché sufficiente, lo studente può chiedere di sostenere una prova orale. La prova orale può aumentare o diminuire il voto risultato dalle prove precedenti; la prova orale è obbligatoria se lo studente desidera ottenere la lode.

Per presentarsi ad una prova scritta o di laboratorio lo studente deve essere iscritto alla prova stessa. Lo studente che, pur essendo iscritto, non può presentarsi è tenuto a darne tempestiva comunicazione al docente.
La registrazione dei voti è contestuale all’esame di laboratorio, o, nel caso di prova orale, contestuale a quest’ultima.

**Norma transitoria**: l’esercitazione finale di laboratorio assegnata nell’a.a. 2017-18 verrà mantenuta valida fino all’appello di febbraio 2019 compreso.

# Testi

[1] Automi, linguaggi e calcolabilita`

J. E. Hopcroft, R. Motwani, and J. D. Ullman,

terza edizione, Pearson/Addison-Wesley, 2009.

]2] Compilatori: principi, tecniche e strumenti

A.H. Aho, M. S.Lam, R. Sethi, J. D. Ullman,

seconda edizione, Pearson/Addison-Wesley, 2009.

# Analisi sintattica e traduzione diretta dalla sintassi

Diverse applicazioni delle tecniche di analisi sintattica e traduzione diretta dalla sintasssi

*(syntax directed translation)*

***Editor intelligienti***: analizza il testo evidenziandone la struttura gerarchica, suggerisce costrutti, aggiunge automaticamente parole chiave, rivela errori sintattici...

***Interpreti di comandi***: nei sistemi operativi (UNIX. DOS,...)

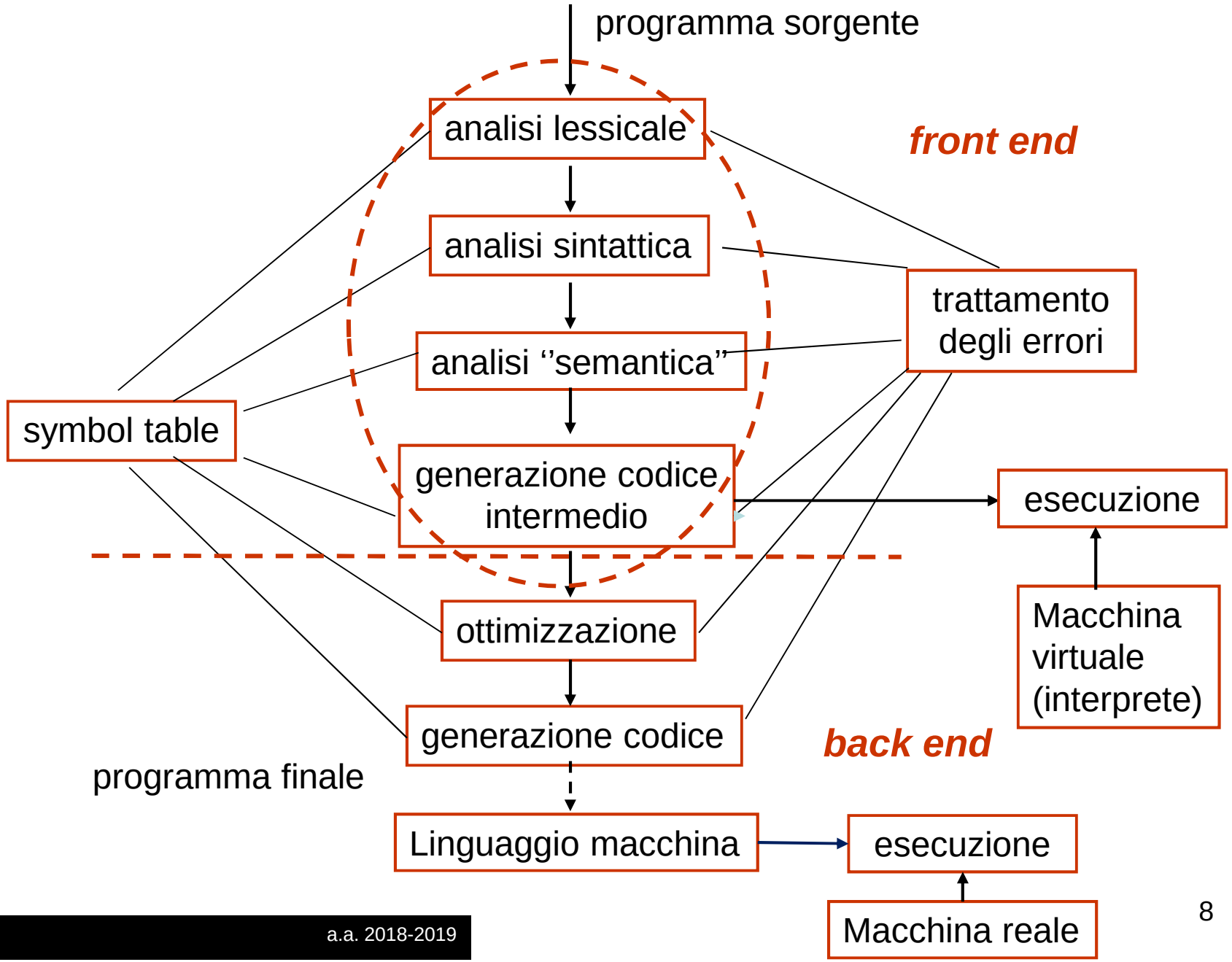
***Pretty printers***: analizzano un programma e lo stampano in modo che la sua struttura diventi visibile usando font speciali e indentazioni.

***Verificatori statici***: cercano di scoprire errori nei programmi senza mandarli in esecuzione. Può scoprire che parti del codice non vengono mai eseguite o che alcune variabili non sono definite o che una variabile di tipo reale è usata come puntatore.

***Interpreti di interrogazioni***: traducono un predicato che contiene operatori booleani e relazionali in comandi per cercare in un database i record che soddisfano il predicato

. . . . . .

# Schema generale di un compilatore-interprete



# Analisi sintattica e traduzione

**posizione := iniziale + velocita * 60**

↓

analizzatore lessicale

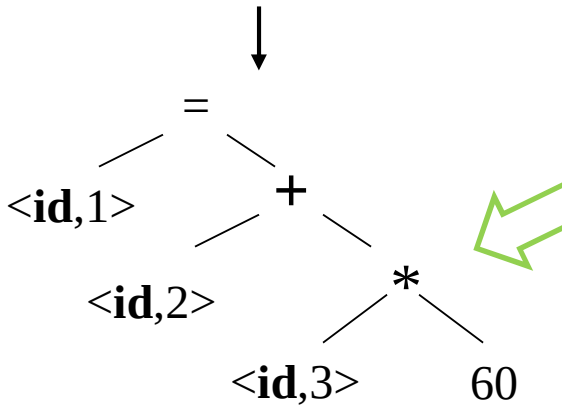
↓

<**id**,1> < = > <**id**,2> <+> <**id**,3> < * > <60>

↓

analizzatore sintattico

↓




↓

analizzatore semantico

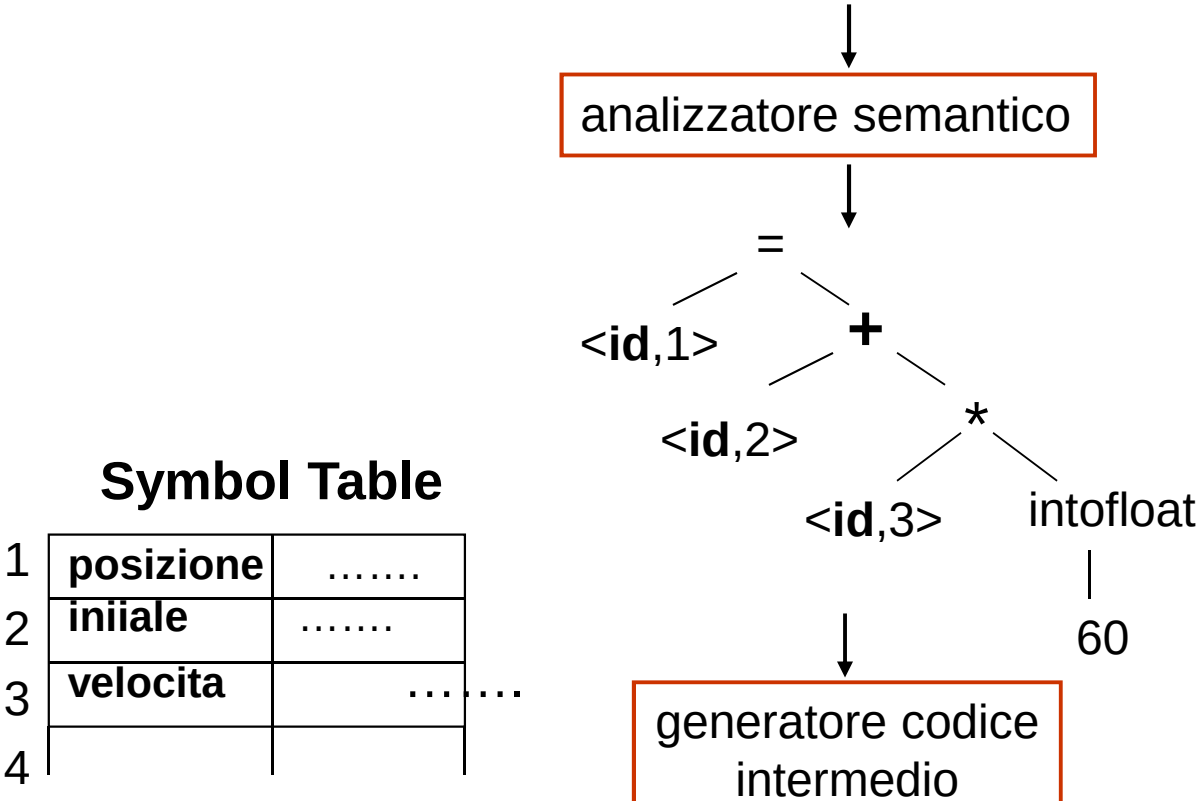
“albero” sintattico (spesso implicito)

**Symbol Table**

| | | |
|---|---|---|
| 1 | **posizione** | ……. |
| 2 | **iniziale** | ……. |
| 3 | **velocita** | ……. |
| 4 | | |

Tipo, indirizzo di allocazione, etc..

# Traduzione







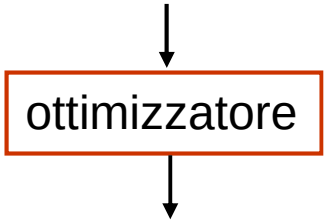
temp1 = intoreal (60)
temp2 = id3 * temp1
temp3 = id2 + temp2
id1 = temp3

# Compilazione

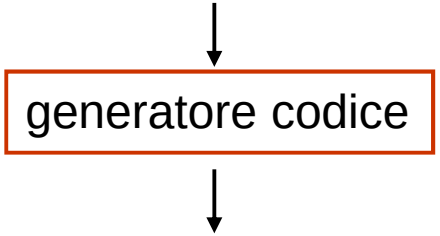
**Symbol Table**

| | | |
|---|---|---|
| 1 | **position** | ……. |
| 2 | **initial** | ……. |
| 3 | **rate** | ……. |
| 4 | | |

temp1 = intofloat (60)
temp2 = id3 * temp1
temp3 = id2 + temp2
id1 = temp3




temp1 = id3 * 60.0
id1 = id2 + temp1




```
LDF     R2, id3
MULF    R2, R2, #60.0,
LDF     R1, id2
ADDF    R1, R1, R2
STF     id1, R1
```

# Vari tipi di codice intermedio

Esempi:

- Codice a 4 indirizzi (compilatori)
- Codice a tre indirizzi (compilatori)
- Codice postfisso (interpreti)
- …..etc.

# Strumenti per la costruzione di compilatori

- *Generatori di scanner*: generano automaticamente un analizzatore lessicale a partire dalla descrizione dei lessemi del linguaggio sorgente tramite espressioni regolari.

- *Generatori di parser*: generano automaticamente un analizzatore sintattico partendo dalla descrizione della grammatica del linguaggio sorgente.

- *Traduttori guidati dalla sintassi*: producono una collezione di routine per la visita degli alberi di parsificazione e la generazione del codice intermedio (in generale di un linguaggio target).