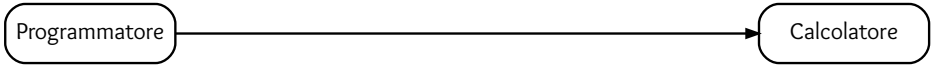
Linguaggi Formali e Traduttori

1.1 Motivazione

- A cosa servono i compilatori?
- Dov'è la difficoltà?
- Perché studiare i compilatori?

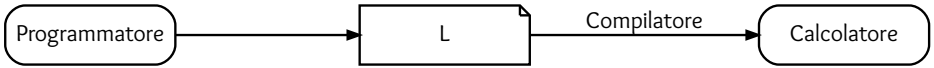
A cosa servono i compilatori?

Programmazione senza compilatori (anni 40/50)



- Il programmatore usa il **linguaggio macchina** (basso livello)
- La programmazione è **difficile** e noiosa

Programmazione con compilatori



- Il programmatore usa un linguaggio di programmazione L (alto livello)
- La programmazione è (più) **facile** e (più) divertente
- Il compilatore **traduce** programmi dal linguaggio L al linguaggio macchina

Dov'è la difficoltà?

```
public static int metodo(int n) {
    int r = 1;
    for (int i = 1; i <= n; i++)
        r = r * i;
    return r;
}
```

Dov'è la difficoltà?

```
public static int metodo(int n) {
    int r = 1;
    for (int i = 1; i <= n; i++)
        r = r * i;
    return r;
}
```

Il calcolatore vede questo codice così

```
70 75 62 6c 69 63 20 73 74 61 74 69 63 20 69 6e
74 20 6d 65 74 6f 64 6f 28 69 6e 74 20 6e 29 20
7b 0a 20 20 20 20 69 6e 74 20 72 20 3d 20 31 3b
0a 20 20 20 20 66 6f 72 20 28 69 6e 74 20 69 20
3d 20 31 3b 20 69 20 3c 3d 20 6e 3b 20 69 2b 2b
29 0a 20 20 20 20 20 20 20 20 72 20 3d 20 72 20
2a 20 69 3b 0a 20 20 20 20 72 65 74 75 72 6e 20
```

Morale

- il processo di riconoscimento e traduzione **non è banale**
- occorre insegnare l'A-B-C al calcolatore (letteralmente!)

Perché studiare i compilatori?

I fatti

- Nel mondo reale (quasi) nessuno realizza compilatori, ma…
- le tecniche studiate qui sono **utili anche per altri scopi** più comuni: lettura e analisi di dati strutturati, lettura di file di configurazione, ecc.

+ aspetti culturali

- Alcune **nozioni fondamentali** dell'informatica sono legate ai compilatori
- L'editor, il word processor, il terminale permettono di fare ricerche (e talvolta sostituzioni) usando **espressioni regolari**. Cosa sono e a cosa servono?
- La sintassi di un (nuovo) linguaggio di programmazione è specificata da una **grammatica**. Che cos'è e come si interpreta?
- Per riconoscere **sequenze** (di simboli, di azioni, di messaggi, ecc.) con certe caratteristiche, di quanta memoria e di quali strutture dati ho bisogno?

+ aspetti metodologici

- I compilatori sono complessi (il primo ha richiesto oltre **10 anni/uomo**!)
- Realizzarne uno richiede una **progettazione** adeguata, strumenti di **generazione automatica del codice** con il supporto di **matematica** (teoria degli insiemi, strutture algebriche) e **logica** (principio di induzione, ragionamento per assurdo)

Linguaggi Formali e Traduttori

1.2 Organizzazione del corso

- Libri
- Fasi di un compilatore
- Altro materiale didattico
- Esame

È proibito condividere e divulgare in qualsiasi forma i materiali didattici caricati sulla piattaforma e le lezioni svolte in videoconferenza: ogni azione che viola questa norma sarà denunciata agli organi di Ateneo e perseguita a termini di legge.

Libri

Linguaggi e automi

- Hopcroft, Motwani, Ullman, **Automi, Linguaggi e Calcolabilità**, Pearson-Addison Wesley, 2009 (qualunque edizione successiva alla prima va bene)
- L'insegnamento usa **buona parte** di questo libro

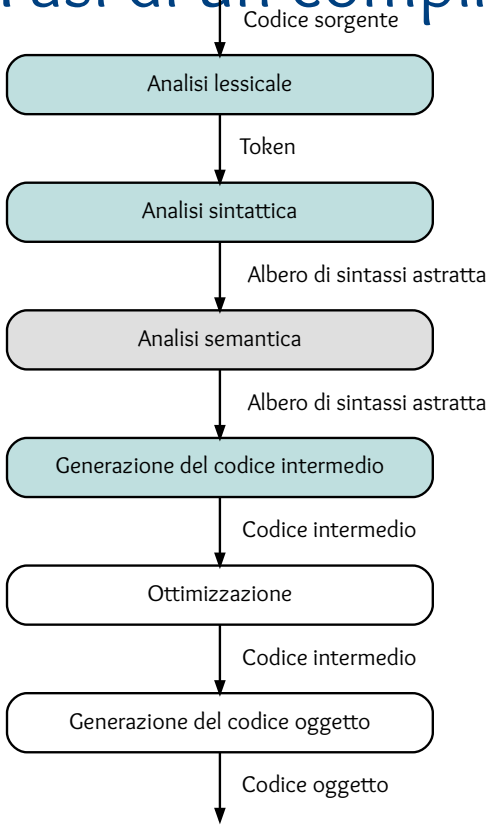
Compilatori

- Aho, Lam, Sethi, Ullman, **Compilatori: Principi, tecniche e strumenti**, Mondadori, 2009
- L'insegnamento usa una **minima parte** di questo libro

Biblioteca

- Entrambi i libri sono disponibili presso la biblioteca del dipartimento in copie multiple

Fasi di un compilatore



Analisi lessicale

- **Automi** → Cap. 2-3-4
- **Compilatori** → Cap. 2-3

Analisi sintattica

- **Automi** → Cap. 5-6-7
- **Compilatori** → Cap. 4

Analisi semantica

- Non affrontata in questo corso
- Un esempio è studiato nel corso di **Linguaggi e Paradigmi di Programmazione** (3º anno)

Generazione del codice intermedio

- **Compilatori** → Cap. 5-6
- Note fornite dai docenti di laboratorio

Altro materiale didattico

Slide di teoria (HTML e PDF)

- Il formato HTML è consultabile se si è online
- Il formato PDF offre lo stesso contenuto di quello HTML, ma differisce lievemente nello stile e può essere annotato e/o stampato su carta

Registrazioni audio/video

- Presentazione degli argomenti con slide e lavagne virtuali
- Sviluppo di esempi
- Risoluzione di esercizi

Esame

Prova scritta (obbligatoria)

- **domande ed esercizi** su tutta la teoria

Prova di laboratorio (obbligatoria)

- **progetto** sviluppato durante le ore di laboratorio (**max 3 studenti** per gruppo)
- discussione **individuale**
- prima occorre **superare la prova scritta** nella **stessa sessione** d'esame

Prova orale

- a **discrezione dello studente** entro la terza consegna della prova scritta
- a **discrezione del docente** dopo la terza consegna della prova scritta
- comunque obbligatoria per chi vuole ottenere la **lode**

Voto

- **media pesata** dei punteggi delle prime due prove (⅔ scritto + ⅓ laboratorio) con eventuale aggiustamento (± 30) dopo prova orale

Linguaggi Formali e Traduttori

1.3 Linguaggi formali

- Alfabeti
- Stringhe
- Operazioni e nozioni sulle stringhe
- Linguaggi
- Esempi
- Operazioni su linguaggi
- Approcci per la descrizione di linguaggi
- Il problema del riconoscimento
- Esercizi
- Soluzioni

Alfabeti

Definizione

Un **alfabeto** è un insieme finito e non vuoto di **simboli**

Notazione

- Usiamo Σ per indicare un alfabeto generico
- Usiamo $a, b, c, \ldots$ per indicare simboli generici di un alfabeto (non necessariamente lettere dell'alfabeto latino!)

Esempi

1. $\Sigma_1 = \{0, 1\}$ alfabeto delle cifre binarie
2. $\Sigma_2 = \{0, 1, \ldots, 9\}$ alfabeto delle cifre decimali
3. $\Sigma_3 = \{\mathtt{a}, \mathtt{b}, \ldots, \mathtt{z}, \mathtt{A}, \mathtt{B}, \ldots, \mathtt{Z}\}$ lettere dell'alfabeto latino
4. $\Sigma_2 \cup \{.\}$ alfabeto dei simboli per rappresentare "numeri con virgola"
5. $\Sigma_2 \cup \Sigma_3 \cup \{\_\}$ alfabeto dei simboli degli identificatori in Java
6. $\Sigma_4 = \{▲, ■, ♦, \ldots\}$ alfabeto di figure geometriche

Stringhe

Definizione

Una **stringa** (o **parola** o **frase**) su un alfabeto Σ è una sequenza finita di simboli in Σ

Notazione

- Usiamo $u, v, w, \ldots$ per indicare stringhe generiche
- Usiamo ε per indicare la **stringa vuota**, quella composta da zero simboli

Definizione

Diciamo che due stringhe sono **uguali** se e solo se sono composte dagli stessi simboli nello stesso ordine (es. **caos** $\neq$ **caso**)

Operazioni e nozioni sulle stringhe

La **lunghezza** di una stringa u è il numero di simboli di cui è costituita e si indica con $|u|$. Ad esempio, $|aab| = 3$ e $|\varepsilon| = 0$

La **concatenazione** di u e v, indicata con uv, è la stringa ottenuta giustapponendo i simboli di u seguiti dai simboli di v. Esempio: la concatenazione di po e `sta` è `posta`

La concatenazione è neutra rispetto alla stringa vuota (cioè $u\varepsilon = \varepsilon u = u$), è associativa (cioè $u(vw) = (uv)w$), ma non commutativa (in generale $uv \neq vu$).

Una stringa u è un **prefisso** (rispettivamente, un **suffisso**) di un'altra stringa w se esiste v tale che $w = uv$ (rispettivamente, $w = vu$). Esempio: `ta` è prefisso di `tacco`

L'**inversa** di $w = a_1 a_2 \cdots a_n$ è la stringa $w^R = a_n \cdots a_2 a_1$. Esempio: $\mathtt{casa}^R = \mathtt{asac}$

Una stringa w è **palindroma** se è uguale alla sua inversa ($w = w^R$). Esempio: `radar`

La **potenza** n-esima di u, indicata con u^n, è la stringa ottenuta concatenando u n volte, ovvero $u^n = \underbrace{uu \cdots u}_{n \text{ volte}}$. Come casi particolari abbiamo $u^0 = \varepsilon$ e $u^1 = u$

Linguaggi

Definizione

Un **linguaggio** L su un alfabeto Σ è un qualunque insieme di stringhe su Σ

Notazione

- Usiamo Σ^* per indicare l'insieme di tutte le stringhe su Σ, inclusa quella vuota
- Usiamo Σ^+ per indicare l'insieme di tutte le stringhe non vuote su Σ

Esempi

- Se $\Sigma = \{0, 1\}$ abbiamo $\Sigma^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, 100, 011, \ldots\}$
- Se $\Sigma = \{a\}$ abbiamo $\Sigma^+ = \{a, aa, aaa, aaaa, \ldots\}$

Esempi

1. $\{a^{2n+1} \mid n \in \mathbb{N}\}$
2. $\{a^m b^n \mid m, n \in \mathbb{N}\}$
3. $\{a^n b^n \mid n \in \mathbb{N}\}$
4. $\{w \in \{a, b\}^* \mid w = w^R\}$

Linguaggi particolari

1. $L = \emptyset$ è il **linguaggio vuoto**, da non confondere con il seguente
2. $L = \{\varepsilon\}$ è il linguaggio composto dalla sola stringa vuota
3. $L = \Sigma$ è il linguaggio costituito dai simboli dell'alfabeto
4. $L = \Sigma^n = \{w \mid w \in \Sigma^* \wedge |w| = n\}$ è il linguaggio delle stringhe lunghe n

Esempi

1. $\{a^{2n+1} \mid n \in \mathbb{N}\}$ è il linguaggio delle stringhe di a di lunghezza dispari
2. $\{a^m b^n \mid m, n \in \mathbb{N}\}$
3. $\{a^n b^n \mid n \in \mathbb{N}\}$
4. $\{w \in \{a, b\}^* \mid w = w^R\}$

Linguaggi particolari

1. $L = \emptyset$ è il **linguaggio vuoto**, da non confondere con il seguente
2. $L = \{\varepsilon\}$ è il linguaggio composto dalla sola stringa vuota
3. $L = \Sigma$ è il linguaggio costituito dai simboli dell'alfabeto
4. $L = \Sigma^n = \{w \mid w \in \Sigma^* \wedge |w| = n\}$ è il linguaggio delle stringhe lunghe n

Esempi

1. $\{a^{2n+1} \mid n \in \mathbb{N}\}$ è il linguaggio delle stringhe di a di lunghezza dispari
2. $\{a^m b^n \mid m, n \in \mathbb{N}\}$ è il linguaggio delle parole composte da un numero arbitrario di a seguite da un numero arbitrario di b
3. $\{a^n b^n \mid n \in \mathbb{N}\}$
4. $\{w \in \{a, b\}^* \mid w = w^R\}$

Linguaggi particolari

1. $L = \emptyset$ è il **linguaggio vuoto**, da non confondere con il seguente
2. $L = \{\varepsilon\}$ è il linguaggio composto dalla sola stringa vuota
3. $L = \Sigma$ è il linguaggio costituito dai simboli dell'alfabeto
4. $L = \Sigma^n = \{w \mid w \in \Sigma^* \wedge |w| = n\}$ è il linguaggio delle stringhe lunghe n

Esempi

1. $\{a^{2n+1} \mid n \in \mathbb{N}\}$ è il linguaggio delle stringhe di a di lunghezza dispari
2. $\{a^m b^n \mid m, n \in \mathbb{N}\}$ è il linguaggio delle parole composte da un numero arbitrario di a seguite da un numero arbitrario di b
3. $\{a^n b^n \mid n \in \mathbb{N}\}$ è il linguaggio delle parole composte da un numero arbitrario di a seguite dallo stesso numero di b
4. $\{w \in \{a, b\}^* \mid w = w^R\}$

Linguaggi particolari

1. $L = \emptyset$ è il **linguaggio vuoto**, da non confondere con il seguente
2. $L = \{\varepsilon\}$ è il linguaggio composto dalla sola stringa vuota
3. $L = \Sigma$ è il linguaggio costituito dai simboli dell'alfabeto
4. $L = \Sigma^n = \{w \mid w \in \Sigma^* \wedge |w| = n\}$ è il linguaggio delle stringhe lunghe n

Esempi

1. $\{a^{2n+1} \mid n \in \mathbb{N}\}$ è il linguaggio delle stringhe di a di lunghezza dispari
2. $\{a^m b^n \mid m, n \in \mathbb{N}\}$ è il linguaggio delle parole composte da un numero arbitrario di a seguite da un numero arbitrario di b
3. $\{a^n b^n \mid n \in \mathbb{N}\}$ è il linguaggio delle parole composte da un numero arbitrario di a seguite dallo stesso numero di b
4. $\{w \in \{a, b\}^* \mid w = w^R\}$ è il linguaggio delle stringhe palindrome su $\{a, b\}$

Linguaggi particolari

1. $L = \emptyset$ è il **linguaggio vuoto**, da non confondere con il seguente
2. $L = \{\varepsilon\}$ è il linguaggio composto dalla sola stringa vuota
3. $L = \Sigma$ è il linguaggio costituito dai simboli dell'alfabeto
4. $L = \Sigma^n = \{w \mid w \in \Sigma^* \wedge |w| = n\}$ è il linguaggio delle stringhe lunghe n

Operazioni su linguaggi

Sono definite le seguenti **operazioni** su linguaggi:

Operazione	**Definizione**
Unione	$L_1 \cup L_2$
Intersezione	$L_1 \cap L_2$
Complemento (rispetto a Σ)	$\overline{L} = \Sigma^* - L$
Concatenazione	$L_1 L_2 = \{uv \mid u \in L_1, v \in L_2\}$
Potenza	$L^0 = \{\varepsilon\} \qquad L^{n+1} = LL^n$
Chiusura di Kleene	$L^* = L^0 \cup L^1 \cup L^2 \cup \cdots = \bigcup_{i \in \mathbb{N}} L^i$
Chiusura transitiva	$L^+ = L^1 \cup L^2 \cup \cdots = \bigcup_{i \in \mathbb{N} - \{0\}} L^i$

Note

- La concatenazione è associativa ma non commutativa in generale.
- La chiusura di Kleene di L produce un linguaggio infinito, a meno che $L \subseteq \{\varepsilon\}$. Ad esempio, $\{a\}^* = \{a^n \mid n \in \mathbb{N}\}$.

Approcci per la descrizione di linguaggi

Problema

- I linguaggi interessanti contengono solitamente un numero **infinito** di stringhe
- Non è pensabile descriverli semplicemente elencandone tutte le stringhe (come accade, ad esempio, con le parole della lingua italiana)
- Occorre un approccio **finito** per descrivere un linguaggio **infinito**

Approccio generativo

- linguaggio = stringhe **generate** da una **grammatica** o **espressione regolare**

Approccio riconoscitivo

- linguaggio = stringhe **riconosciute** da un **automa**

Perché due approcci?

- grammatiche ed espressioni regolari sono facili da leggere e scrivere per gli umani
- gli automi sono efficienti da "eseguire" per i calcolatori
- i due approcci possono essere messi in relazione! (lo vedremo in questo corso)

Il problema del riconoscimento

Data una descrizione (finita) di un linguaggio L (potenzialmente infinito) e una stringa w, determinare se $w \in L$

- Il linguaggio L è solitamente descritto usando espressioni regolari o grammatiche libere
- L'automa o il parser che riconosce L è **generato** automaticamente

Esercizi

Esercizio 1

Dimostrare con dei controesempi che non sono valide le seguenti relazioni. Per ciascuna di esse, trovare la forma corretta o delle condizioni sufficienti a farla valere:

1. $L\emptyset = \emptyset L = L$
2. $L\{\varepsilon\} = \{\varepsilon\}L = \{\varepsilon\}$
3. $L_1 L_2 = L_2 L_1$
4. $L^+ = L^* - \{\varepsilon\}$

Esercizio 2

Elencare dieci stringhe dei seguenti linguaggi definiti sull'alfabeto $\Sigma = \{a, b, c\}$:

1. $(\Sigma^2 \cup \Sigma^3)\{a, b\}$
2. $\Sigma^+ - \{b, c\}^*$
3. $\{w \in \Sigma^* \mid w \text{ contiene un egual numero di } a, b \text{ e } c\}$
4. $\{w \in \Sigma^* \mid w \text{ è palindroma, ovvero } w = w^R\}$

Soluzioni

Esercizio 1

1. $L\emptyset = \emptyset L = \emptyset$
2. $L\{\varepsilon\} = \{\varepsilon\}L = L$
3. $L_1 L_2 = L_2 L_1$ se $L_1 = \emptyset$ o $L_1 = \{\varepsilon\}$ o $L_1 = L_2$ o $L_1 = L_2^*$ ecc.
4. $L^+ = L^* - \{\varepsilon\}$ se $\varepsilon \notin L$

Esercizio 2

1. aaa, aba, aca, caa, cab, baa, bba, bca, bcb, $aaab$, ...
2. tutte le stringhe che contengono almeno una a, ad es. a, ab, ba, ac, ca, abb, bac, ...
3. ε, abc, acb, bac, bca, cab, cba, $aabbcc$, $aabcbc$, $aaccbb$, ...
4. ε, a, b, c, aa, bb, cc, aaa, aba, aca, ...