
**Corso di Laurea Magistrale in Ingegneria Informatica**
**A.A. 2010-2011**

# Linguaggi Formali e Compilatori

## *I linguaggi formali*

**Giacomo PISCITELLI**

# Traduttori

Un **traduttore** è un programma che effettua la traduzione automatica da un linguaggio ad un altro.

I linguaggi utilizzati nelle applicazioni informatiche sono linguaggi formali.

Data una frase $f_1$ nel linguaggio formale $L_1$ (**linguaggio sorgente**), il traduttore costruisce una frase $f_2$ del linguaggio formale $L_2$ (**linguaggio destinazione, target o pozzo**).

La frase $f_2$ deve "corrispondere" a $f_1$.

# Cosa è un linguaggio?

In matematica, logica, informatica e linguistica, per **linguaggio formale** si intende un insieme di **stringhe** di lunghezza finita costruite sopra un **alfabeto** finito, cioè sopra un insieme finito di oggetti tendenzialmente semplici che vengono chiamati **caratteri, simboli** o **lettere**.

# Alfabeto

Un **alfabeto** è un insieme finito di elementi, detti simboli o caratteri terminali.

Esempi:

{a, b, c}

{0, 1}

{α, β, γ, δ}

La **cardinalità** di un alfabeto è il numero di simboli dell'alfabeto.

Se Σ denota un alfabeto, |Σ| denota la sua cardinalità.

Esempi:

|{a, b, c}| = 3

|{0, 1}| = 2

|{α, β, γ, δ}| = 4

# Stringa

Una **stringa s** o **parola** su un alfabeto è una sequenza (o lista) di simboli appartenenti all'alfabeto.

Esempi:

- ✓ aabb, cac, cba, abba sono stringhe sull'alfabeto {a, b, c}
- ✓ i numeri scritti in binario sono stringhe sull'alfabeto {0, 1}

Due parole che differiscono solo per l'ordine dei simboli sono diverse: aabb e abba sono due parole diverse.

**Due parole sono uguali** solo se i loro caratteri letti ordinatamente da sinistra a destra coincidono.

La **lunghezza** di una stringa s, denotata da |s| è il numero dei suoi caratteri.

Esempi: |aabb| = 4  |cac| = 3  |101011| = 6

Due stringhe uguali hanno la stessa lunghezza (ma non vale il viceversa)

La **stringa vuota** ε (talvolta denotata da λ) è la stringa che non contiene nessun simbolo.

La lunghezza della stringa vuota è 0: |ε| = 0

# Linguaggio

Un **linguaggio** su un alfabeto è un insieme di stringhe su quell'alfabeto.

Esempi:

- ✓ {aabb, cac, cba, abba} è un linguaggio sull'alfabeto {a, b, c}
- ✓ l'insieme dei numeri scritti in binario è un linguaggio sull'alfabeto {0, 1}
- ✓ l'insieme delle stringhe palindrome contenenti solo i simboli a, b, c è un linguaggio sull'alfabeto {a, b, c}

Notare che il primo ed il terzo linguaggio hanno lo stesso alfabeto. In generale, dato un alfabeto, su di esso si possono definire infiniti linguaggi.

- ➢ $\phi$, l'insieme vuoto è un linguaggio.
- ➢ {ε} è il linguaggio contenente la sola stringa vuota.
- ➢ L'insieme di tutti i possibili programmi C è un linguaggio.
- ➢ L'insieme di tutti i possibili identificatori in un linguaggio è un linguaggio.

# Cardinalità di un linguaggio

La **cardinalità** di un linguaggio è il numero delle sue stringhe.

Se L denota un linguaggio, |L| denota la sua cardinalità.
Esempi:

- ✓ |{aabb, cac, cba, abba}| = 4
- ✓ |l'insieme dei numeri scritti in binario|=∞

Un linguaggio è finito se la sua cardinalità è finita: un linguaggio finito è anche detto un **vocabolario**.

Un linguaggio è infinito se la sua cardinalità è infinita.

Il linguaggio vuoto (denotato da **Ø**) è il linguaggio che non contiene alcuna stringa.

|Ø| = 0

**Attenzione!!!**

Ø ≠ {ε}, in quanto |Ø| = 0 ≠ |{ε}| = 1

# Operazioni sulle stringhe: concatenamento

Il **concatenamento di 2 stringhe** è la stringa formata da tutti i simboli della prima stringa seguiti da tutti quelli della seconda stringa.
x.y oppure xy denota il concatenamento delle stringhe x e y
Se $x = a_1 \ldots a_h$ e $y = b_1 \ldots b_k$ allora $xy = a_1 \ldots a_h\ b_1 \ldots b_k$
Esempi:
nano.tecnologie = nanotecnologie
tele.visione = televisione

Il concatenamento non è commutativo: $x.y \neq y.x$
visione.tele = visionetele

Il concatenamento è associativo: x.(y.z) = (x.y).z
nano.(tecno.logie) = nano.tecnologie = nanotecnologie
(nano.tecno).logie = (nanotecno).logie = nanotecnologie

# Operazioni sulle stringhe: concatenamento

La lunghezza del concatenamento di due stringhe è la somma delle lunghezze delle stringhe: $|x.y| = |x| + |y|$
$|televisione| = 11 = 4 + 7 = |tele| + |visione|$

La stringa vuota è l'elemento neutro del concatenamento: $\varepsilon x = x = x\varepsilon$

La stringa $y$ è una **sottostringa** della stringa $x$ se esistono delle stringhe $u$, $v$ tali che $x = uyv$

La stringa $y$ è un **prefisso** della stringa $x$ se esiste una stringa $u$ tale che $x = yu$

La stringa $y$ è un **suffisso** della stringa $x$ se esiste una stringa $u$ tale che $x = uy$

Una sottostringa (prefisso, suffisso) di una stringa è **propria** se non coincide con la stringa data.

Se $|x| \geq k$ indichiamo con $k{:}x$ il prefisso di $x$ di lunghezza $k$ (**inizio di lunghezza** $k$ di $x$).

# Operazioni sulle stringhe: concatenamento

Esempi:

- ✓ le sottostringhe di abbc sono {ε, a, b, c, ab, bb, bc, abb, bbc, abbc}
- ✓ le sottostringhe proprie di abbc sono {ε, a, b, c, ab, bb, bc, abb, bbc}
- ✓ i prefissi di abbc sono {ε, a, ab, abb, abbc}
- ✓ i prefissi propri di abbc sono {ε, a, ab, abb}
- ✓ i suffissi di abbc sono {ε, c, bc, bbc, abbc}
- ✓ i suffissi propri di abbc sono {ε, c, bc, bbc}
- ✓ 2 : abbc = ab
- ✓ 3 : abbc = abb

# Operazioni sulle stringhe: riflessione

La **riflessione** di una stringa è la stringa ottenuta scrivendo i caratteri in ordine inverso.

$x^R$ denota la riflessione della stringa $x$
$(a_1 \ldots a_h)^R = a_h \ldots a_1$

$(abbc)^R = cbba$

La riflessione è idempotente: $(x^R)^R = x$

La riflessione della concatenazione di due stringhe è la concatenazione inversa delle loro riflessioni: $(xy)^R = y^R x^R$

La riflessione della stringa vuota è la stringa vuota: $\varepsilon^R = \varepsilon$

La riflessione ha la precedenza sul concatenamento: $abbc^R = abbc$

# Operazioni sulle stringhe: potenza

La **potenza** m**-esima** della stringa x è il concatenamento di x con se stessa m volte.

$x^m$ denota potenza m-esima di x

$$x^m \quad \rightarrow \quad \text{if } m = 0 \text{ then } \varepsilon \text{ else } x^{m-1}x$$

Esempi:
$(abbc)^3 = abbcabbcabbc$
$(abbc)^6 = abbcabbcabbcabbcabbcabbc$

La potenza ha la precedenza sul concatenamento: $abbc^3 = abbccc$

# Operazioni sui linguaggi

I linguaggi sono insiemi!

L'**unione** $L_1 \cup L_2$ dei linguaggi $L_1$ ed $L_2$ è l'insieme delle stringhe di $L_1$ o di $L_2$

$$L_1 \cup L_2 = \{x \mid x \in L_1 \vee x \in L_2\}$$

L'**intersezione** $L_1 \cap L_2$ dei linguaggi $L_1$ ed $L_2$ è l'insieme delle stringhe di $L_1$ e di $L_2$

$$L_1 \cap L_2 = \{x \mid x \in L_1 \,\&\, x \in L_2\}$$

La **differenza** $L_1 - L_2$ del linguaggio $L_1$ meno il linguaggio $L_2$ è l'insieme delle stringhe di $L_1$ che non appartengono a $L_2$

$$L_1 - L_2 = \{x \mid x \in L_1 \,\&\, x \notin L_2\}$$

Esempi:

$\{ab, abc\} \cup \{ab, aa, cb\} = \{ab, abc, aa, cb\}$

$\{ab, abc\} \cap \{ab, aa, cb\} = \{ab\}$

$\{ab, abc\} - \{ab, aa, cb\} = \{abc\}$

$\{ab, aa, cb\} - \{ab, abc\} = \{aa, cb\}$

# Operazioni sui linguaggi

## Inclusione

Il linguaggio $L_1$ è **incluso** nel linguaggio $L_2$ (notazione $L_1 \subseteq L_2$) se tutte le stringhe di $L_1$ appartengono ad $L_2$

$$L_1 \subseteq L_2 \Leftrightarrow \quad \forall x \in L_1 \Rightarrow x \in L_2$$

Il linguaggio $L_1$ è **propriamente incluso** nel linguaggio $L_2$ (notazione $L_1 \subset L_2$) se tutte le stringhe di $L_1$ appartengono ad $L_2$ ed almeno una stringa di $L_2$ non appartiene ad $L_1$

$$L_1 \subset L_2 \quad \Leftrightarrow \quad (\forall x \in L_1 \Rightarrow x \in L_2) \,\&\, (\exists\, y \in L_2 \,.\, y \notin L_1)$$

Due linguaggi sono **uguali** se contengono lo stesso insieme di stringhe.

$$L_1 = L_2 \quad \Leftrightarrow \quad L_1 \subseteq L_2 \,\&\, L_2 \subseteq L_1$$

$L_1 \subseteq L_1 \cup L_2$

$L_1 \cap L_2 \subseteq L_1$

$L_1 - L_2 \subseteq L_1$

# Operazioni sui linguaggi

La **riflessione** del linguaggio L (notazione $L^R$) è l'insieme delle stringhe riflesse di L

$$L^R = \{x^R \mid x \in L\}$$

$\{ab, abc\}^R = \{ba, cba\}$

Il **concatenamento** dei linguaggi $L_1$ ed $L_2$ (notazione $L_1\ L_2$ ) è l'insieme ottenuto concatenando in tutti i modi possibili le stringhe di $L_1$ con le stringhe di $L_2$

$$L_1\ L_2 = \{xy \mid x \in L_1 \ \&\ y \in L_2\}$$

Esempi:

$\{ab, abc\}\{ab, aa, cb\} = \{abab, abaa, abcb, abcab, abcaa, abccb\}$

$L\ \emptyset = \emptyset = \emptyset\ L$

$L\{\varepsilon\} = L = \{\varepsilon\}L$

# Operazioni sui linguaggi

La **potenza** m**-esima** del linguaggio L (notazione $L^m$) è il concatenamento di L con se stesso m volte.

$$L^m \quad \rightarrow \quad \text{if } m = 0 \text{ then } \{\varepsilon\} \text{ else } L^{m-1}L$$

$\{ab, abc\}^2 = \{abab, ababc, abcab, abcabc\}$

$\emptyset^0 = \{\varepsilon\}$

La **stella di Kleene** (o **chiusura rispetto al concatenamento**) del linguaggio L (notazione $L^*$) è l'unione di tutte le potenze di L

$$L^* = \cup_{h=0\ldots\infty} L^h = \{\varepsilon\} \cup L^1 \cup L^2 \ldots$$

$\{ab, abc\}^* = \{\varepsilon, ab, abc, abab, ababc, abcab, abcabc, \ldots\}$

$L \subseteq L^*$ (monotonicità)

$(x \in L) \,\&\, (y \in L) \Rightarrow xy \in L$ (chiusura rispetto al concatenamento)

$(L^*)^* = L^*$ (idempotenza)

$(L^*)^R = (L^R)^*$ (commutatività della stella e della riflessione)

$\emptyset^* = \{\varepsilon\}$

$\{\varepsilon\}^* = \{\varepsilon\}$

# Operazioni sui linguaggi

La **croce** (o **chiusura non riflessiva rispetto al concatenamento**) del linguaggio L (notazione $L^+$) è l'unione di tutte le potenze positive di L

$$L^+ = \cup_{h=1\ldots\infty} L^h = L^1 \cup L^2 \ldots$$

$\{ab, abc\}^+ = \{ab, abc, abab, ababc, abcab, abcabc, \ldots\}$

$L^* = L^+ \cup \{\varepsilon\}$

$L^+ = LL^* = L^*L$

Il **quoziente** dei linguaggi $L_1$ ed $L_2$ (notazione $L_1/L_2$ ) è l'insieme dei prefissi di $L_1$ che, concatenati con le stringhe di $L_2$, producono stringhe di $L_1$

$$L_1/ L_2 = \{x \mid xy \in L_1 \,\&\, y \in L_2\}$$

$\{ab, abc\}/\{bc\} = \{a\}$

$\{bc\}/\{ab, abc\} = \{\}$

# Linguaggio universale (monoide libero)

Il **linguaggio universale** o **monoide libero** di un alfabeto $\Sigma$ è l'insieme (infinito) di tutte le stringhe di tale alfabeto.

Esso può essere definito attraverso un passaggio al limite dell'elevamento a potenza, cioè è la sua stella $\Sigma^*$

$\{a, b\}^* = \{\varepsilon, a, b, aa, bb, ab, ba, \ldots\}$

Ogni linguaggio su un alfabeto $\Sigma$ è un sottoinsieme di $\Sigma^*$

Il **complemento** di un linguaggio $L$ (notazione $\neg L$) su un alfabeto $\Sigma$ è la differenza fra $\Sigma^*$ ed $L$

$$\neg L = \Sigma^* - L$$

$\neg\{ab, ba\} = \{\varepsilon, a, b, aa, bb, aaa, \ldots\}$

# Sostituzione di una stringa in un linguaggio sorgente

La **sostituzione in una stringa sorgente** x (su un alfabeto $\Sigma$) di un **linguaggio pozzo** L (su un alfabeto $\Delta$) (notazione $\Phi_{a->L}(x)$) al posto di un carattere $a \in \Sigma$ è l'insieme delle stringhe sull'alfabeto $(\Sigma - \{a\}) \cup \Delta$ ottenute sostituendo le occorrenze di a nella stringa x con stringhe di L

L = {bc, bb}

$\Phi_{a->L}$ (aba) = {bcbbc, bcbbb, bbbbc, bbbbb}

L = {ac, bb}

$\Phi_{a->L}$ (aba) = {acbac, acbbb, bbbac, bbbbb}

# Sostituzione di un linguaggio pozzo in un linguaggio sorgente

La **sostituzione in un linguaggio sorgente** $L_1$ (su un alfabeto $\Sigma$) di un **linguaggio pozzo** $L_2$ (su un alfabeto $\Delta$) (notazione $\Phi_{a->L2}(L_1)$) al posto di un carattere $a \in \Sigma$ è l'insieme delle stringhe sull'alfabeto $(\Sigma - \{a\}) \cup \Delta$ ottenute sostituendo le occorrenze di $a$ nelle stringhe di $L_1$ con stringhe di $L_2$

$L_1 = \{ac, aba\}$

$L_2 = \{bc, bb\}$

$\Phi_{a->L2}(L_1) = \{bcc, bbc, bcbbc, bcbbb, bbbbc, bbbbb\}$

# Generalità sulla teoria dei linguaggi formali

La **teoria dei linguaggi formali** studia gli **insiemi di stringhe**, cioè le frasi di un linguaggio, al fine di definire quelle da considerare corrette e con quale significato.

Ma non è facile specificare un linguaggio completamente e rigorosamente, essendo impossibile elencare tutte le frasi valide, che sono infinite e di lunghezza *a priori* illimitata.

È necessario pertanto ricorrere ad un algoritmo, che, attraverso un insieme finito di regole, permetta di produrre le frasi del linguaggio – eventualmente infinite o, in alternativa, di verificare la loro correttezza.

# La specifica di un linguaggio

## L'approccio generativo

In questo caso si ricorre ad un algoritmo di enumerazione, che, attraverso un insieme finito di regole di calcolo, permette di produrre le frasi del linguaggio – eventualmente infinite.

Le regole dell'algoritmo di enumerazione costituiscono la cosiddetta **grammatica (o sintassi) generativa** del linguaggio.

## L'approccio riconoscitivo

In questo caso si ricorre ad un algoritmo che riconosce se una frase è corretta o no, e ne determina il significato.

In pratica, invece di far uso di un algoritmo, si preferisce ricorrere ad una descrizione più astratta, mediante il concetto di **automa riconoscitore**.

# Altri approcci alla specifica di un linguaggio

Gli approcci indicati non sono gli unici impiegati dalla teoria dei linguaggi formali, che studia le proprietà dei linguaggi seguendo vari approcci:

***Approccio generativo***: un linguaggio viene definito come l'insieme di tutte e sole le stringhe che vengono prodotte da una grammatica generativa o da qualche altro sistema di riscrittura. Questo approccio è quello normalmente usato nel manuale o nei documenti che descrivono il linguaggio.

***Approccio riconoscitivo***: un linguaggio viene definito come l'insieme di tutte e sole le stringhe che vengono accettate da un automa. È l'approccio analitico usato per descrivere il compilatore (o interprete) di un linguaggio.

***Approccio denotazionale***: un linguaggio viene definito mediante espressioni simboliche compatte, come le espressioni regolari, che ne denotano (tutte e sole) le stringhe in forma concisa.

***Approccio algebrico***: un linguaggio viene definito attraverso sue proprietà algebriche.

***Approccio trasformazionale***: un linguaggio viene definito come ottenuto sottoponendo a una data trasformazione un altro linguaggio, tendenzialmente più semplice.

# Gli approcci oggetto di studio

In questo corso sarà principalmente esaminato l'**approccio generativo**.

Ci si limiterà al tipo più semplice e diffuso di grammatiche: quelle non contestuali (o del tipo 2 della gerarchia di Chomsky), e ad un loro sottoinsieme, le espressioni regolari.

Nel corso si esaminerà anche **l'approccio riconoscitivo**, basato su un **automa riconoscitore** o *analizzatore*.

# Approccio generativo vs. Approccio riconoscitivo

**I due approcci sono duali ed equivalenti**: infatti è sempre possibile passare da un algoritmo di enumerazione, cioè una grammatica, ad uno di riconoscimento, cioè un automa, in modo completamente meccanico.

Infatti le regole che permettono ad una grammatica di enumerare le frasi appartenenti ad un linguaggio possono, di converso, consentire ad un automa di riconoscere l'appartenenza di una frase ad un linguaggio.

La progettazione degli analizzatori sintattici fornisce un chiaro esempio di tale passaggio.

# Dalla verifica di correttezza alla traduzione della frase

Accanto all'esigenza primaria di riconoscere se una frase è corretta, si pone anche l'obiettivo di tradurre (o trasformare) la frase, come fa un compilatore (o un interprete) quando converte un programma dal linguaggio di alto livello al codice macchina di un processore.

Una **traduzione** è una corrispondenza (in particolare una funzione) tra le frasi del linguaggio sorgente e quelle del linguaggio pozzo.

Anche per la traduzione si danno due approcci:

- quello **generativo** impiega degli schemi sintattici per generare le coppie di frasi, sorgente e pozzo, che si corrispondono nella traduzione; uno schema sintattico è infatti l'accoppiamento di due grammatiche generative;
- quello **riconoscitivo**, più propriamente traduttivo, si avvale di automi trasduttori, che si differenziano dai riconoscitori per la capacità di emettere la traduzione voluta.

# Traduzione mediante Metodi sintattici o Metodi semantici

Come nella linguistica, anche nel campo dei linguaggi artificiali, non è semplice chiarire la distinzione, a volte arbitraria, tra sintassi e semantica.

Nella linguistica i due termini sono assunti ad emblema della distinzione tra forma e contenuto, anche se, quando si tenta di approfondire, tale distinzione, apparentemente chiara, diventa elusiva.

Volendo semplificare, la differenza fra i due metodi consiste ne:

- **il formalismo**, che nella sintassi fa uso di elementi (alfabeti, stringhe di simboli, …) e operazioni (di concatenazione, iterazione, sostituzione, omomorfismo, …), mentre nella semantica si considerano anche i numeri, le operazioni aritmetiche, il calcolo proposizionale e dei predicati;
- **la maggiore complessità degli algoritmi semantici**; i linguaggi che consideriamo sono quasi esclusivamente quelli riconoscibili e traducibili in tempo lineare, cioè proporzionale alla lunghezza della frase considerata; ciò è un'enorme semplificazione, ma esclude situazioni che richiederebbero complessità di calcolo almeno quadratiche.