Giovanni Semeraro

Elementi di TEORIA dei LINGUAGGI FORMALI

SWAP researchgroup
Semantic Web Access & Personalization
"Antonio Bello" research group
www.di.uniba.it/~swap

GIOVANNI SEMERARO

Elementi di TEORIA dei LINGUAGGI FORMALI

Indice

1. Introduzione 1

2. Grammatiche e linguaggi 13

2.1 Linguaggi formali e monoidi liberi 13

2.2 Generazione e riconoscimento di linguaggi formali..... 18

2.3 Grammatiche generative..... 23

2.4 Derivazioni 26

2.5 Linguaggi generati da grammatiche e correttezza di una grammatica 29

2.6 Esercizi 34

3. Linguaggi liberi da contesto e linguaggi dipendenti da contesto 63

3.1 Grammatiche e linguaggi liberi da contesto 63

3.2 Grammatiche e linguaggi dipendenti da contesto 64

3.3 Relazione tra linguaggi C.F. e C.S. 65

3.4 Grammatiche e linguaggi monotoni 66

3.5 Esercizi proposti 75

4. Linguaggi liberi da contesto..... 77

4.1 Alberi di derivazione 77

4.2 Principio di sostituzione di sottoalberi 81

4.3 Pumping Lemma per i linguaggi liberi da contesto..... 89

4.4 Ambiguità di grammatiche e linguaggi 92

4.5 Esercizi 98

4.6 Esercizi proposti 114

5. Grammatiche e macchine 117

5.1 Classificazione delle grammatiche secondo Chomsky..... 117

5.2 Teorema della Gerarchia 118
5.3 Lemma della stringa vuota 121
5.4 Operazioni sui linguaggi..... 122
5.5 Proprietà di chiusura delle classi di linguaggi 126
5.6 L'operazione di riflessione 141
5.7 Esercizi 144
5.8 Esercizi proposti 153
6. Automi a stati finiti (deterministici e non deterministici).... 155
6.1 Automi a stati finiti deterministici..... 155
6.2 Linguaggi a stati finiti 159
6.3 Automi a stati finiti non deterministici..... 160
6.4 Linguaggi accettati da automi non deterministici..... 165
6.5 Equivalenza delle classi di linguaggi accettati da automi a stati finiti deterministici e non deterministici 166
6.6 Proprietà di chiusura della classe dei linguaggi a stati finiti .. 169
6.7 Esercizi 174
6.8 Esercizi proposti 180
7. Linguaggi regolari, espressioni regolari e teorema di Kleene 181
7.1 Linguaggi regolari ed espressioni regolari 181
7.2 Proprietà delle espressioni regolari 185
7.3 Teorema di Kleene 189
7.4 Pumping Lemma per i linguaggi regolari..... 197
7.5 Esercizi 201
7.6 Esercizi proposti 227
8. Automi a pila e grammatiche libere..... 235
8.1 Automi a pila 235
8.2 Linguaggi accettati da automi a pila..... 241
8.3 Esempi di linguaggi riconosciuti da automi a pila 245

8.4 Forme normali per grammatiche non contestuali 255
8.5 Proprietà decidibili dei linguaggi liberi da contesto 282
8.6 Esercizi proposti 284
9. Cenni su automi limitati lineari e macchine di Turing 291
9.1 Automi limitati lineari 291
9.2 Insiemi ricorsivi 295
9.3 Insiemi ricorsivamente enumerabili 300
9.4 Problema della derivabilità per grammatiche Tipo 0 309
10. Analisi sintattica. Grammatiche LL(k) e LR(k) 313
10.1 Analisi sintattica (Parsing) 313
10.2 Grammatiche LL(k) 318
10.3 Esercizi 321
10.4 Grammatiche LR(k) 334
10.5 Esercizi proposti 345
Bibliografia 349

1. Introduzione

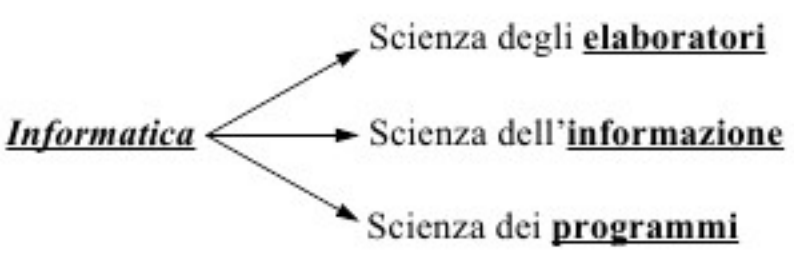
La teoria dei linguaggi formali rappresenta spesso uno scoglio per gli studenti di informatica a causa della sua forte dipendenza dalla notazione.

L'argomento, d'altra parte, non può essere evitato perché ogni laureato in informatica deve possedere una buona comprensione dell'operazione di compilazione e questa, a sua volta, si fonda pesantemente sulla teoria dei linguaggi formali.

Questo testo ha l'intenzione di fornire una base sufficiente alla comprensione della teoria dei linguaggi formali.

In particolare, ci concentreremo quasi esclusivamente su due tipi di grammatiche: regolari e libere da contesto, poiché i linguaggi formali utilizzati in informatica sono per lo più di questi tipi.

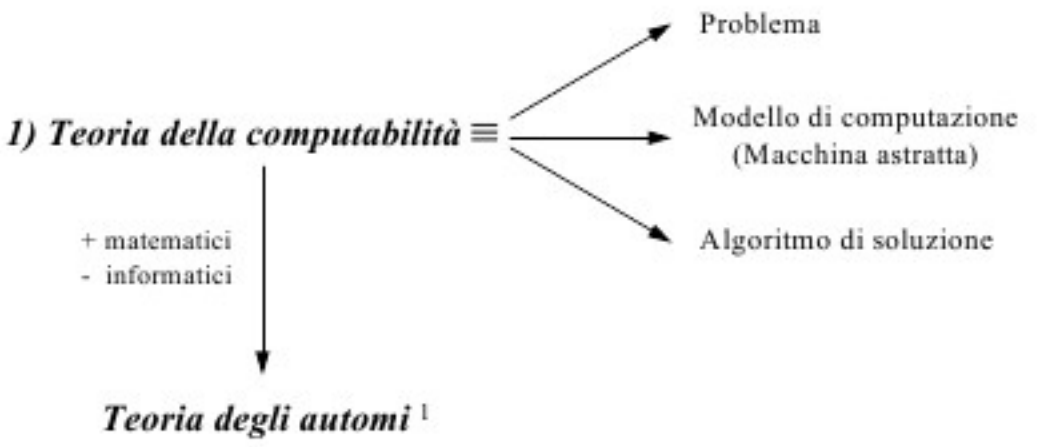
L'informatica teorica è la **scienza degli algoritmi**, in tutti i loro aspetti, poiché è il concetto di algoritmo che è in qualche modo comune a tutti i settori dell'informatica.



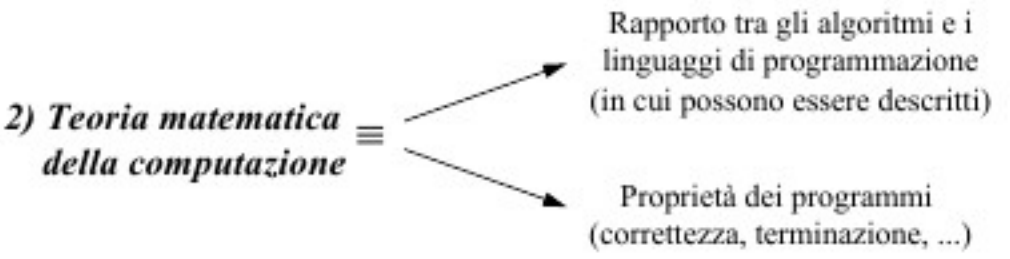
elaboratori	≡	macchine che eseguono gli algoritmi
informazione	≡	materia su cui lavorano gli algoritmi
programmi	≡	algoritmi descritti in un particolare linguaggio

Aree di ricerca dell'informatica teorica

Lo schema riportato in Figura 1.1 fornisce una panoramica delle aree di ricerca che ricadono nell'ambito più generale dell'informatica teorica. Tali aree sono:



Particolarmente feconda si è rivelata l'interazione tra la ***Teoria degli Automi*** e la ***Teoria dei Linguaggi Formali***. Quest'ultima, nata in ambito linguistico per caratterizzare i linguaggi naturali, è stata sviluppata ed utilizzata dagli informatici teorici, che avevano l'esigenza di descrivere gli algoritmi con linguaggi di programmazione comprensibili dalla macchina, ma non troppo difficili per l'uomo e soprattutto non ambigui.



---

[1] La teoria degli automi si occupa della definizione di diverse classi di dispositivi di calcolo e, per ogni classe, dello studio delle proprietà e delle classi di problemi che sono in grado di risolvere.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

3) Teoria della complessità computazionale ≡ Rapporto tra gli algoritmi e le risorse necessarie per eseguirli / Aspetti quantitativi del procedimento di soluzione di un problema

Quando iniziamo a studiare un linguaggio, formale o meno, tutti quanti godiamo di un grande vantaggio: siamo esperti in un linguaggio, quello con cui comunichiamo con gli altri. Qualunque sia la nostra lingua madre, l'italiano, l'inglese, o altro, abbiamo sviluppato una competenza considerevole e continueremo a svilupparla per il resto della nostra vita.

Oltre ad essere competenti in uno o più linguaggi naturali, lo studente in informatica ha familiarità con diversi linguaggi di programmazione, come il FORTRAN, il PASCAL, il C, il PROLOG, ...

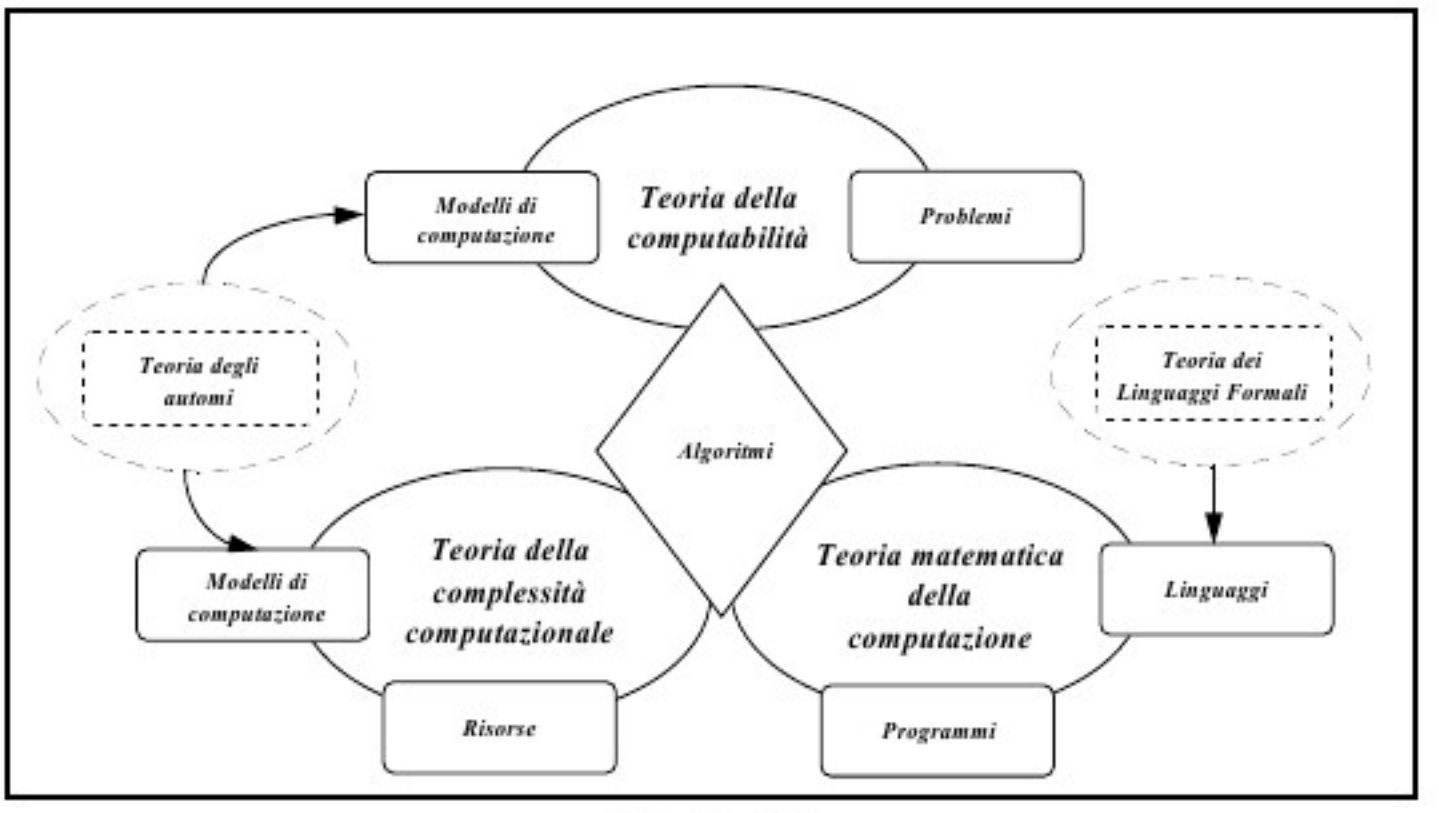


Figura 1.1

Questi linguaggi sono usati per scrivere programmi e quindi per comunicare con il computer. Senza dubbio, chiunque utilizzi un computer potrà notare che esso è particolarmente limitato nel comprendere il significato desiderato dei programmi. Mentre in linguaggio naturale possiamo di solito comunicare anche attraverso frasi mal strutturate e spesso incomplete, quando dobbiamo comunicare con un computer la situazione cambia.

I nostri programmi devono aderire rigorosamente a regole stringenti e anche differenze minori vengono rigettate come errate. Idealmente, ogni frase in un linguaggio dovrebbe essere corretta sia ***semanticamente*** (avere cioè il corretto significato) sia ***sintatticamente*** (avere la corretta struttura grammaticale).

Nell'italiano parlato, le frasi sono spesso sintatticamente sbagliate, ciononostante convogliano la semantica desiderata.

In programmazione, comunque, è essenziale che la sintassi sia corretta al fine di comunicare una qualsivoglia semantica. Quando studiamo la semantica di una frase, ne stiamo studiando il significato. Tutte le frasi che seguono hanno la stessa interpretazione semantica, cioè lo stesso significato, sebbene siano sintatticamente differenti:

The man hits the dog

The dog is hit by the man

L'homme frappe le chien

Lo studio della sintassi è lo studio della grammatica, cioè della struttura delle frasi. La frase:

The man hits the dog

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

può essere analizzata sintatticamente, cioè risolta nelle parti grammaticali componenti, come segue:

The man	hits	the dog
< parte nominale >	< parte verbale >	< parte nominale >

ed ogni frase in questa forma è sintatticamente valida in inglese. Possiamo descrivere un particolare insieme di tali frasi semplici in inglese usando le seguenti regole:

< frase semplice > :: = < parte nominale > < parte verbale > < parte nominale >

< parte nominale > :: = < articolo > < nome >

< nome > :: = ***car*** | ***man*** | ***dog***

< articolo > :: = ***The*** | ***a***

< parte verbale > :: = ***hits*** | ***eats***

Queste regole sono scritte in BNF[2], una notazione usata comunemente per descrivere la sintassi dei linguaggi di programmazione.

Nell'esempio, < frase semplice > è definita come una < parte nominale > seguita da una < parte verbale > seguita, a sua volta, da un'altra < parte nominale >.

Ciascuna delle due occorrenze di < parte nominale > deve essere espansa in < articolo > seguito da un < nome >.

---

[2] BNF (Backus Naur Form) è un metalinguaggio.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Scegliendo di espandere la prima occorrenza di < articolo > in ***The***, la prima occorrenza di < nome > in ***man***, la < parte verbale > in ***hits***, il secondo < articolo > in ***The*** ed infine l'ultimo <nome > in ***dog***, si dimostra che la frase

The man hits the dog

è una delle nostre frasi semplici.

Tutto ciò si può riassumere nel seguente albero di derivazione (Figura 1.2).

Albero di derivazione

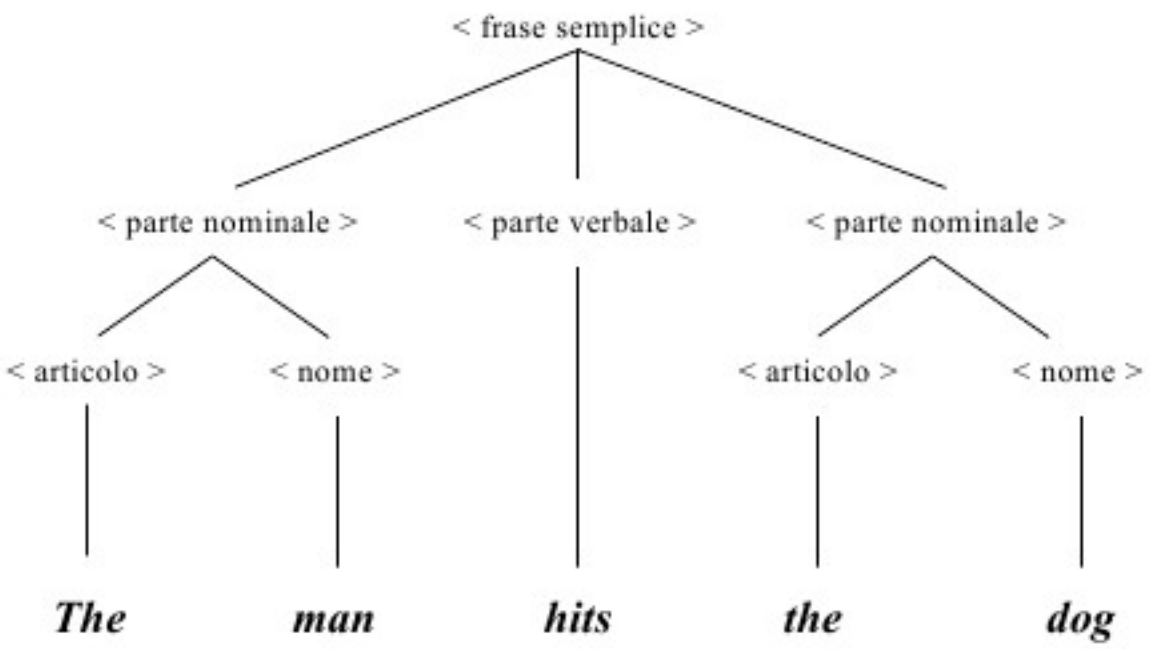


Figura 1.2

La definizione data di < frase semplice > dà luogo a 72 diverse frasi derivabili. Sebbene tutte siano sintatticamente corrette, in base alla nostra definizione, alcune non hanno un'interpretazione semantica sensata (non hanno senso compiuto).

Una di queste è la frase:

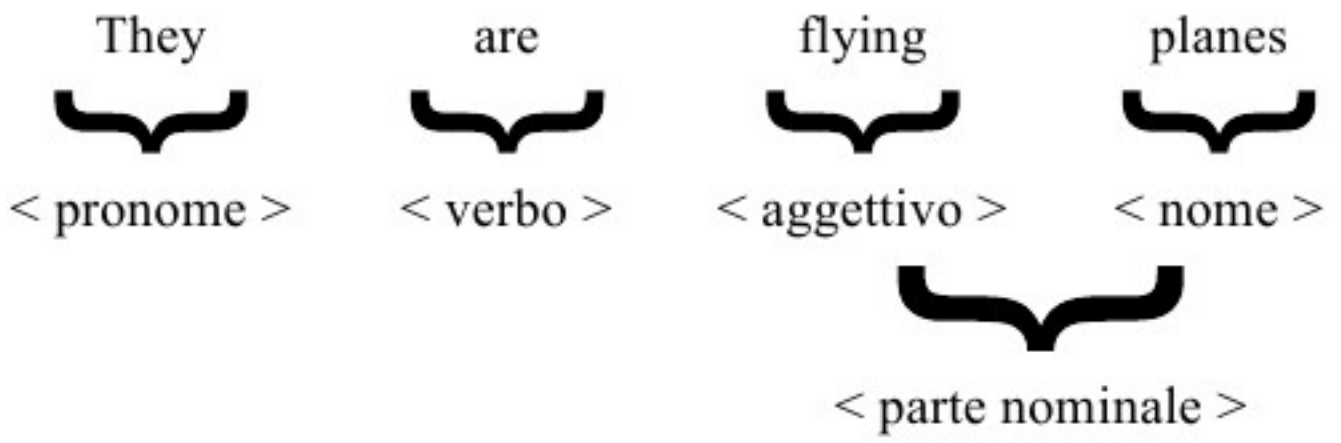
The car eats the man

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

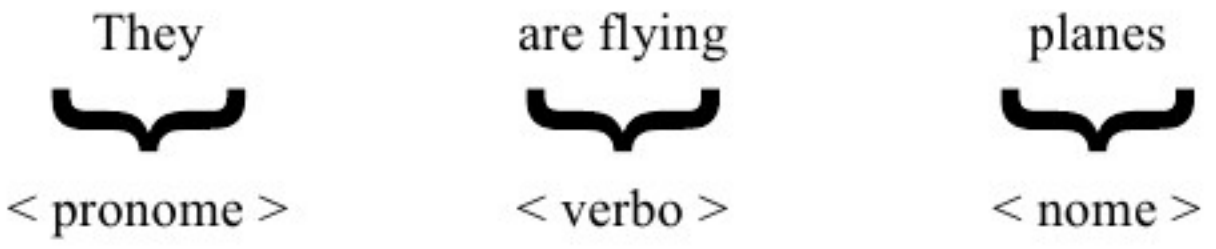
Non necessariamente una frase corretta ha senso. Consideriamo la seguente frase (non derivabile da <frase semplice> utilizzando le regole date in precedenza):

They are flying planes

Un'analisi sintattica di questa frase è:



Quest'analisi suggerisce che ***they*** si riferisce a ***planes*** (*aerei nel cielo che volano*). Un'analisi alternativa sarebbe



e questa implicherebbe un'interpretazione semantica completamente differente, in cui ***they*** si riferisce a chiunque sia attualmente al controllo dei ***planes*** (*essi stanno pilotando gli aerei*).

Nell'esempio, l'analisi sintattica è di aiuto all'interpretazione semantica. Per quanto sia scelto "ad hoc" e mostri un fenomeno poco comune nel linguaggio naturale, l'esempio è illustrativo di un concetto importante in computazione.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Una fase cruciale nel processo di compilazione di un programma è l'analisi sintattica, come passo essenziale per la sua interpretazione semantica.

Teoria dei Linguaggi Formali

Negli anni '50, N. Chomsky, linguista americano (1928 -) cerca di descrivere la sintassi del linguaggio naturale secondo semplici regole di riscrittura e trasformazione. Chomsky considera alcune restrizioni sulle regole sintattiche e classifica i linguaggi in base alle restrizioni imposte alle regole che generano tali linguaggi. Una classe importante di regole che generano linguaggi formali va sotto il nome di ***grammatiche libere da contesto*** (prive di contesto) o ***Context-free grammars*** (C.F.).

L'importanza di tale classe (e dei linguaggi da essa generati) risiede nel fatto che lo sviluppo dei primi linguaggi di programmazione di alto livello (ALGOL 60), che segue di pochi anni il lavoro di Chomsky, dimostra che le grammatiche C.F. sono strumenti adeguati a descrivere la sintassi di base di molti linguaggi di programmazione.

Esempio 1.1 (Linguaggio C.F.)

Un linguaggio C.F. è il linguaggio delle parentesi ben formate. Tale linguaggio comprende tutte le stringhe di parentesi aperte e chiuse bilanciate correttamente. Ad esempio,

( ) è ben formata;

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

(( ) ( )) è ben formata;

(( ) ( ) non è ben formata.

Questo linguaggio gioca un ruolo fondamentale in informatica come notazione per contrassegnare il "raggio d'azione" nelle espressioni matematiche e nei linguaggi di programmazione (in cui spesso si usano *begin* ed *end* - PASCAL, ALGOL 60 - oppure $\{$ e $\}$ - C - al posto di "(" e ")", rispettivamente).

<u>Definizione 1.1</u> (Linguaggio delle parentesi ben formate)

i) La stringa $(\ )$ è ben formata;

ii) se la stringa di simboli A è ben formata, allora lo è anche (A);

iii) se le stringhe A e B sono ben formate, allora lo è anche la stringa AB. ■

In corrispondenza di questa definizione induttiva, possiamo considerare un sistema di riscrittura che genera esattamente l'insieme delle stringhe lecite di parentesi ben formate:

(1) $S \to (\ )$

(2) $S \to (S)$

(3) $S \to SS$

Le regole di riscrittura (1), (2) e (3) sono dette *produzioni* o *regole di produzione*. Esse stabiliscono che "data una stringa, si può formare una nuova stringa sostituendo una S o con $(\ )$ o con (S) o con SS".

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Se confrontiamo la definizione di parentesi ben formate e le tre produzioni, si osserva una corrispondenza diretta tra le parti i) e ii) della definizione e le produzioni (1) e (2). A differenza delle prime due, la produzione (3) è apparentemente diversa dalla parte (iii) della definizione induttiva data in precedenza. Difatti, abbiamo utilizzato simboli distinti A e B nella definizione induttiva per evidenziare il fatto che le due stringhe di parentesi ben formate che consideriamo non sono necessariamente uguali. Nella produzione corrispondente non c'è necessità di usare simboli distinti perché, in $S \to SS$, le due S che compaiono a destra possono essere sostituite indipendentemente.

Possiamo cioè applicare una produzione ad una delle due S (alla prima o alla seconda) indifferentemente. Il risultato non cambia se consideriamo prima la S a sinistra e poi quella a destra o viceversa.

<u>Esempio 1.2</u> (Generazione di $((\ )(((\ )(\ ))$)

Primo modo:

$$S \underset{(3)}{\Rightarrow} SS \underset{(2)}{\Rightarrow} (S)S \underset{(1)}{\Rightarrow} ((\ ))S \underset{(2)}{\Rightarrow} ((\ ))(S) \underset{(3)}{\Rightarrow} ((\ ))(SS) \underset{(1)}{\Rightarrow} ((\ ))((\ )S) \underset{(1)}{\Rightarrow} ((\ ))((\ )(\ ))$$

La corrispondente sequenza di applicazione delle produzioni è:

$(3) \to (2) \to (1) \to (2) \to (3) \to (1) \to (1)$

Secondo modo:

$$S \underset{(3)}{\Rightarrow} SS \underset{(2)}{\Rightarrow} S(S) \underset{(3)}{\Rightarrow} S(SS) \underset{(1)}{\Rightarrow} S((\ )S) \underset{(1)}{\Rightarrow} S((\ )(\ )) \underset{(2)}{\Rightarrow} (S)((\ )(\ )) \underset{(1)}{\Rightarrow} ((\ ))((\ )(\ ))$$

La corrispondente sequenza di applicazione delle produzioni è:

$(3) \to (2) \to (3) \to (1) \to (1) \to (2) \to (1)$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Una sequenza di applicazioni di regole di produzione prende il nome di *derivazione*.

Notazione

Nell'esempio 1.2 abbiamo fatto uso della notazione $\alpha \underset{(n)}{\Rightarrow} \beta$. La sua interpretazione è la seguente: "da α si produce direttamente β per effetto dell'applicazione della regola di riscrittura n". Ad esempio,

$$SS \underset{(2)}{\Rightarrow} (S)S$$

si legge: "SS produce direttamente $(S)S$ per effetto dell'applicazione della regola di riscrittura (2)".

Le derivazioni precedenti (modi 1 e 2) vengono riassunte attraverso la notazione:

$$S \overset{*}{\Rightarrow} ((\ ))((\ )(\ ))$$

che leggiamo come: "S produce $((\ ))((\ )(\ ))$".

Nelle regole di produzione si è fato uso di due tipi di simboli: caratteri che possono apparire nelle derivazioni, ma non nelle stringhe finali, detti *simboli nonterminali* o *variabili* (nell'esempio, S è il solo nonterminale) e caratteri che possono apparire nelle stringhe finali, detti *simboli terminali* (nell'esempio, "(" e ")" sono i simboli terminali).

In BNF si usa la seguente notazione:

S	$<S>$
$\Rightarrow$	$::=$

2. Grammatiche e linguaggi

2.1 Linguaggi formali e monoidi liberi

Il concetto di linguaggio formale è strettamente correlato a quello di *monoide libero* (generato da un insieme).

Definizione 2.1 (Alfabeto)

Un insieme X finito e non vuoto è un *alfabeto*. ■

Esempio 2.1

- L'alfabeto latino, con l'aggiunta dei simboli di interpunzione e dello spazio bianco: *a b c …z* ; , . :
- L'insieme delle dieci cifre arabe: 0 1 … 9

Con i simboli primitivi dell'alfabeto si formano le parole (es.: *abc*, 127, *casa*,…).

Definizione 2.2 (Parola o stringa)

Una sequenza finita di simboli $x_1 x_2 ... x_n$, dove ogni x_i è preso da uno stesso alfabeto X è una *parola* (su X). ■

Esempio 2.2

$X = \{0,1\}$.

001110 è una parola su X.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Una parola è ottenuta giustapponendo o concatenando simboli (caratteri) dell'alfabeto.

Se una stringa ha m simboli (non necessariamente distinti) allora diciamo che ha lunghezza m.

La lunghezza di una stringa w è denotata con $|w|$. Le parole di lunghezza 1 sono i simboli di X.

Quindi 001110 è una parola di lunghezza 6:

$$|001110| = 6$$

La parola vuota (o stringa vuota), denotata con λ, è una stringa priva di simboli ed ha lunghezza 0:

$$|\lambda| = 0.$$

Definizione 2.3 (Uguaglianza tra stringhe)

Due stringhe sono *uguali* se i loro caratteri, letti ordinatamente da sinistra a destra, coincidono. ■

Definizione 2.4 (X^*)

L'insieme di tutte le stringhe di lunghezza finita sull'alfabeto X si denota con X^*. ■

Esempio 2.3

Se $X = \{0,1\}$, allora $X^* = \{\lambda, 0, 1, 00, 01, 10, 11, ...\}$.

X^* ha un numero di elementi che è un infinito numerabile. Dalla definizione, segue che $\lambda \in X^*$, per ogni insieme X.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Definizione 2.5 (Concatenazione o prodotto)

Sia $\alpha \in X^*$ una stringa di lunghezza m e $\beta \in X^*$ una stringa di lunghezza n, la *concatenazione* di α e β, denotata con $\alpha\beta$ o $\alpha \cdot \beta$, è definita come la stringa di lunghezza $m+n$, i cui primi m simboli costituiscono una stringa uguale a α ed i cui ultimi n simboli costituiscono una stringa uguale a β.

Quindi se $\alpha = x_1 x_2 ... x_m$ e $\beta = x'_1 x'_2 ... x'_n$, si ha: $\alpha\beta = x_1 x_2 ... x_m x'_1 x'_2 ... x'_n$.

La concatenazione di stringhe su X è un'operazione binaria su X^*:

$$\cdot \; : \; X^* \times X^* \to X^*$$

- è associativa: $(\alpha\beta)\gamma = \alpha(\beta\gamma) = \alpha\beta\gamma, \quad \forall \alpha, \beta, \gamma \in X^*$;
- non è commutativa: $\exists \alpha, \beta \in X^* \; : \; \alpha\beta \neq \beta\alpha$;
- ha elemento neutro λ: $\lambda\alpha = \alpha\lambda = \alpha, \quad \forall \alpha \in X^*$.

Dunque $(X^*, \cdot)$ è un monoide (non commutativo). ■

Osservazione 2.1

In base alla definizione di prodotto, ogni parola non vuota $\alpha = x_1 x_2 ... x_n$ si può scrivere in uno ed un solo modo come prodotto di parole di lunghezza 1, cioè di elementi di X.

Ciò si esprime dicendo che:

$(X^*, \cdot)$ è il monoide libero generato dall'insieme X.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Definizioni 2.6 (Prefisso, suffisso, sottostringa)

Se $\gamma \in X^*$ è della forma $\gamma = \alpha\beta$, ove $\alpha, \beta \in X^*$, allora α è un *prefisso* di γ e β è un *suffisso* di γ.

Se $\delta, \beta \in X^*$ e δ è della forma $\delta = \alpha\beta\gamma$, ove $\alpha, \gamma \in X^*$, allora β è una *sottostringa* di δ. ■

Esempio 2.4

Sia $\gamma = 00110$. Allora $\{\lambda, 0, 00, 001, 0011, \gamma\}$ è l'insieme dei prefissi di γ, $\{\lambda, 0, 10, 110, 0110, \gamma\}$ è l'insieme dei suffissi di γ e $\{\lambda, 0, 1, 00, 01, 10, 11, 001, 011, 110, 0011, 0110, \gamma\}$ è l'insieme delle sottostringhe di γ.

Definizione 2.7 (Potenza di una stringa)

Data una stringa α su X, la *potenza* h-esima di α è definita (induttivamente) come segue:

$$\alpha^h = \begin{cases} \lambda & \text{se } h = 0 \\ \alpha\alpha^{h-1} & \text{altrimenti} \end{cases}$$

con $h = 0, 1, 2, \ldots$ ■

Dunque, la potenza h-esima di una stringa è un caso speciale di concatenamento (in quanto la si ottiene concatenando una stringa h volte con se stessa).

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Definizione 2.8 (Potenza di un alfabeto)

Sia X un alfabeto, poniamo:

1) $X^1 = X$;

2) $X^2 = \{x_1x_2 \mid x_1, x_2 \in X,\ x_1x_2 \equiv x_1 \cdot x_2\}$

3) $X^3 = \{x_1x_2x_3 \mid x_1x_2 \in X^2,\ x_3 \in X,\ x_1x_2x_3 \equiv x_1x_2 \cdot x_3\}$

..) .....

i) $$X^i = \{x_1x_2...x_{i-1}x_i \mid x_1x_2...x_{i-1} \in X^{i-1},\ x_i \in X,\ x_1x_2...x_i \equiv x_1x_2...x_{i-1} \cdot x_i\}$$

Se $i \geq 2$ si ha:

$$X^+ = X \cup X^2 \cup ... \cup X^i \cup ... = \bigcup_{i=1}^{+\infty} X^i$$

Se λ è la parola vuota e prendiamo un $w \in X^+$ tale che $w \cdot \lambda = \lambda \cdot w = w$ si ha:

$$X^* = \{\lambda\} \cup X^+$$

Inoltre si ha:

$$X^h = \begin{cases} \{\lambda\} & \text{se } h = 0 \\ X^{h-1} \cdot X & \text{altrimenti} \end{cases}$$

■

Definizione 2.9 (Linguaggio Formale)

Un *linguaggio formale* L su un alfabeto X è un sottoinsieme di X^*.

$$L \subseteq X^*.$$

■

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Esempio 2.5

Il linguaggio delle parentesi ben formate è un linguaggio formale in quanto, denotato con M tale linguaggio, si ha:

$$M \subset \{ (,) \}^* .$$

I linguaggi formali possono essere di natura molto diversa l'uno dall'altro.

Esempio 2.6

Un linguaggio di programmazione può essere costruito a partire dall'alfabeto X dei simboli sulla tastiera.

L'insieme, finito o infinito, dei programmi ben costruiti sintatticamente (ossia, che rispettano la sintassi) costituisce un linguaggio.

Esempio 2.7

Consideriamo l'insieme dei teoremi di una teoria matematica. I teoremi sono particolari stringhe di simboli del nostro alfabeto. L'insieme dei teoremi "ben formati" rappresenta un linguaggio. Ad esempio, la stringa "$ab=ba$" non è un teorema della teoria dei gruppi, ma della teoria dei gruppi abeliani.

2.2 Generazione e riconoscimento di linguaggi formali

A noi interessano i linguaggi formali da almeno due *punti di vista*

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

1) **Descrittivo/Generativo**

Come possiamo *generare* gli elementi di un dato linguaggio *L*?

Un linguaggio finito può essere descritto/generato per elencazione dei suoi elementi (se il numero dei suoi elementi non è troppo grande).

Un linguaggio infinito non è elencabile. I linguaggi infiniti sono i più interessanti perché devono essere specificati necessariamente attraverso una *proprietà* che ne caratterizza gli elementi, che ne definisce l'intensione. Tale proprietà può essere vista come una regola da seguire per generare gli elementi del linguaggio.

Il vero problema è trovare la(e) regola(e) generativa(e) (di produzione) di un linguaggio. È quello che accade quando si impara un linguaggio: non è possibile memorizzare tutte le frasi del linguaggio.

Esempio 2.8

Non è possibile "elencare" tutti i teoremi della teoria dei gruppi, perché sono infiniti i teoremi realizzabili combinando quelli noti. Un libro di teoria dei gruppi non è l'elencazione dei teoremi, ma fornisce una serie di assiomi e le regole con le quali, a partire dagli assiomi, è possibile costruire tutti i teoremi della teoria dei gruppi. Per descrivere la regola di produzione di un linguaggio, utilizzeremo una notazione insiemistica.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Esempio 2.9

Sia L il linguaggio su $X = \{0\}$ costituito da tutte e sole le stringhe che hanno un numero pari di 0, cioè: $L = \{\lambda, 00, 0000, 000000, \ldots\}$.

2) **Riconoscitivo**

Come possiamo *riconoscere* gli elementi di un dato linguaggio L?

Questo secondo punto di vista ha come obiettivo la costruzione di "macchine" in grado di decidere/stabilire se una stringa è un elemento di L oppure no. Si intende costruire una "macchinetta" (Figura 2.1) cui dare in ingresso una particolare parola e che produce una tra due possibili risposte:

$$sì \equiv '\in L' \quad \text{e} \quad no \equiv '\notin L'$$

Figura 2.1

Esempio 2.10

L'esecuzione di un programma errato sintatticamente viene inibita. Questo è indice dell'esistenza di una "macchinetta" che

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

stabilisce se il programma appartiene o no all'insieme dei programmi sintatticamente ben costruiti.

Analizziamo il problema della generazione di L.

Esempio 2.11

Sia dato l'alfabeto:

$$X = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, +, -\}.$$

Voglio generare il linguaggio L dei numeri interi relativi. Ovviamente

$$L \subseteq X^*$$

Più precisamente, $L \subset X^*$ poiché, ad esempio, $1++-5 \notin L$.

Non possiamo elencare gli elementi di L. Cerchiamo dunque una serie di regole mediante le quali è possibile produrre tutti e soli gli elementi di L.

Assumiamo, per semplicità, che un numero relativo sia costituito da una serie di cifre precedute da + o −.

Adottiamo la BNF per descrivere le produzioni:

$$< S > ::= + < I > \mid - < I >$$

$$< I > ::= < D > \mid < I > < D >$$

$$< D > ::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$$

Queste regole generano tutti gli interi relativi purché partiamo dal simbolo nonterminale S.

I è il simbolo nonterminale (da ora in poi, talvolta abbreviato in *NT*), anche detto *categoria sintattica*, che sta ad indicare (e da cui si genera) la classe dei numeri interi.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

I è definito ricorsivamente o come una cifra oppure come un intero seguito da una cifra.

Ogni intero relativo è generato da queste regole e niente che non sia un intero relativo può essere generato da queste regole.

Generazione ad albero

Proviamo a generare l'intero relativo –375:

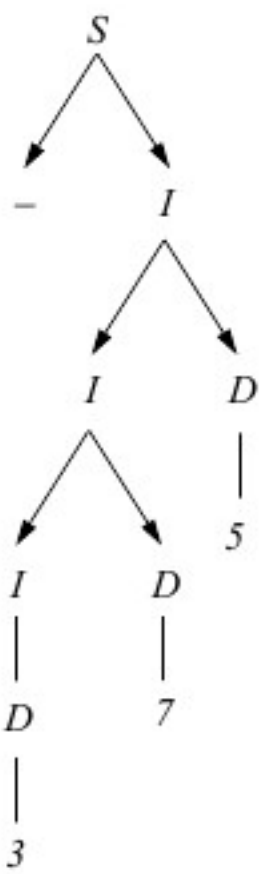


Figura 2.2

Tale albero prende il nome di ***albero di derivazione***.

$$S \overset{*}{\Rightarrow} -375 \quad \Leftrightarrow \quad -375 \in L$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Nella notazione vista per il linguaggio delle parentesi ben formate, tipica per i linguaggi formali, la grammatica diventa:

$$S \to +I \mid -I$$

$$I \to D \mid I\,D$$

$$D \to 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$$

2.3 Grammatiche generative

Dagli esempi di linguaggi visti, possiamo trarre le seguenti conclusioni.

Per generare un linguaggio sono necessari:

- un insieme X di simboli primitivi con cui si formano le parole del linguaggio, detto *alfabeto dei simboli terminali* o *alfabeto terminale*;
- un insieme V di simboli ausiliari o variabili con cui si identificano le categorie sintattiche del linguaggio, detto *alfabeto dei simboli nonterminali (ausiliari)* o *alfabeto nonterminale* o *alfabeto delle variabili*;
- un simbolo speciale S, scelto tra i nonterminali, da cui far partire la generazione delle parole del linguaggio. Tale simbolo è detto *assioma* o *scopo* o *simbolo distintivo* o *simbolo di partenza* o *simbolo iniziale*;
- un insieme P di produzioni, espresse in un formalismo quali regole di riscrittura $(\alpha \to \beta)$, BNF (α ::= β), carte sintattiche, …

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Possiamo, a questo punto, dare la definizione formale di grammatica (generativa) o grammatica a struttura di frase, come strumento per la generazione di un linguaggio.

Definizione 2.10 (Grammatica generativa o a struttura di frase)
Una *grammatica generativa* o *a struttura di frase* G è una quadrupla

$$G = (X, V, S, P)$$

ove:

(1) X è l'*alfabeto terminale* per la grammatica;

(2) V è l'*alfabeto nonterminale* o delle variabili per la grammatica;

(3) S è il *simbolo di partenza* per la grammatica;

(4) P è l'insieme delle *produzioni* della grammatica

ed inoltre valgono le seguenti condizioni:

$$X \cap V = \varnothing \quad \text{e} \quad S \in V$$

■

Definizione 2.11 (Produzione)
Una *produzione* è una coppia (v,w),

ove $v \in (X \cup V)^+$ e v contiene un NT

$\Leftrightarrow v \in (X \cup V)^* V (X \cup V)^*$

$w \in (X \cup V)^*$ (w può essere anche λ).

Un elemento (v,w) di P viene comunemente scritto nella forma:

$$v \to w$$

■

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Una produzione deve, in qualche modo, riscrivere un *NT*.

Per convenzione, gli elementi di X sono rappresentati di solito con lettere minuscole (con o senza pedici e di solito sono le prime lettere dell'alfabeto) o cifre ed operatori (connettivi), mentre gli elementi di V sono rappresentati con lettere maiuscole (con o senza pedici) o con stringhe delimitate dalle parentesi angolari "<" e ">".

La notazione

$$\alpha \to \beta_1 \mid \beta_2 \mid \ldots \mid \beta_k$$

è impiegata come abbreviazione della seguente:

$$\begin{array}{c} \alpha \to \beta_1 \\ \alpha \to \beta_2 \\ \vdots \\ \alpha \to \beta_k \end{array}$$

<u>Esempio 2.12</u>

- La grammatica per il linguaggio delle parentesi ben formate:

$$G_1 = \big(\{\,(\,,\,)\,\},\quad \{S\},\quad S,\quad \{S \to (\ \ ),\ S \to (S),\ S \to SS\}\big)$$

- La grammatica per il linguaggio dei numeri interi relativi:

$$\begin{aligned} G_2 = \big(&\{0,1,2,3,4,5,6,7,8,9,+,-\},\quad \{S,I,D\},\quad S,\\ &\{S \to +I,\ S \to -I,\ I \to D,\ I \to ID,\\ &D \to 0,\ D \to 1,\ \ldots,\ D \to 9\}\big) \end{aligned}$$

2.4 Derivazioni

Definizioni 2.12 (Derivazione o produzione diretta, derivazione)
Sia $G = (X, V, S, P)$ una grammatica e siano y e z due stringhe finite di simboli in $X \cup V$ (stringhe di terminali e nonterminali) tali che:

$$y = \gamma\alpha\delta \text{ e } z = \gamma\beta\delta \text{, ove } y \in (X \cup V)^+,\ z \in (X \cup V)^*,$$

$$\alpha, \beta, \gamma, \delta \in (X \cup V)^*.$$

1) Scriviamo

$$y \Rightarrow z$$

e diciamo che *y produce direttamente z* o che *z è derivata direttamente da y* se:

$$\alpha \to \beta \in P,$$

ossia se esiste in G una produzione $\alpha \to \beta$.

2) Scriviamo

$$y \overset{*}{\Rightarrow} z$$

e diciamo che *y produce z* o che *z è derivabile da y* se $y = z$ o esiste una sequenza di stringhe $w_1, w_2, ..., w_n$, con $w_1, w_2, ..., w_{n-1} \in (X \cup V)^+$, $w_n \in (X \cup V)^*$, $w_1 = y$ e $w_n = x$ tali che $\forall i,\ i = 1,2,...,n : w_i \underset{G}{\Rightarrow} w_{i+1}$ (w_i produce direttamente w_{i+1}),

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

cioè:

$$y \overset{*}{\Rightarrow} z \quad \Leftrightarrow \quad \begin{cases} y = z \\ \quad \text{oppure} \\ w_1 = y \Rightarrow w_2 \Rightarrow w_3 \Rightarrow \ldots \Rightarrow w_{n-1} \Rightarrow w_n = z \end{cases}$$

<u>Osservazione 2.2</u>

La nozione di derivazione diretta stabilisce una relazione binaria in $(X \cup V)^*$. Date due stringhe y e z, il simbolo $\Rightarrow$ può esserci o meno; dipende dall'esistenza di una produzione. Allora possiamo anche definire una composizione di relazioni:

$$y \overset{2}{\Rightarrow} z \quad \overset{def}{\Leftrightarrow} \quad \exists\, w : y \Rightarrow w \ \text{ e } \ w \Rightarrow z$$

dove 2 è il numero di trascrizioni necessarie per passare da y a z (ossia, la *lunghezza della derivazione*).

Da ciò si ha:

$$\overset{*}{\Rightarrow} \quad = \quad I \cup \Rightarrow \cup \overset{2}{\Rightarrow} \cup \overset{3}{\Rightarrow} \cup \ldots$$

ove I è la relazione identica e $\overset{n}{\Rightarrow}$ indica la composizione della relazione $\Rightarrow$ n volte con se stessa.

$\overset{*}{\Rightarrow}$ è la chiusura riflessiva e transitiva della relazione di derivazione diretta;

$\overset{+}{\Rightarrow}$ è la chiusura transitiva della stessa relazione.

<u>Esempio 2.13</u>

Costruzione di un flow-chart.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Sia dato il flow-chart in Figura 2.3.

La relazione $\Rightarrow$ è definita come segue: "due nodi sono in relazione $\Rightarrow$ se esiste un arco orientato tra essi".

$\overset{*}{\Rightarrow}$ è la chiusura riflessiva e transitiva dell'operatore $\Rightarrow$ e dunque va interpretata come: "esiste un cammino di lunghezza finita che congiunge i due nodi"

$R = \Rightarrow$

R^0	x	y	z	t	u
x	1				
y		1			
z			1		
t				1	
u					1

R^1	x	y	z	t	u
x		1		1	
y					1
z	1	1			
t					1
u					

R^2	x	y	z	t	u
x					1
y					
z		1		1	1
t					
u					

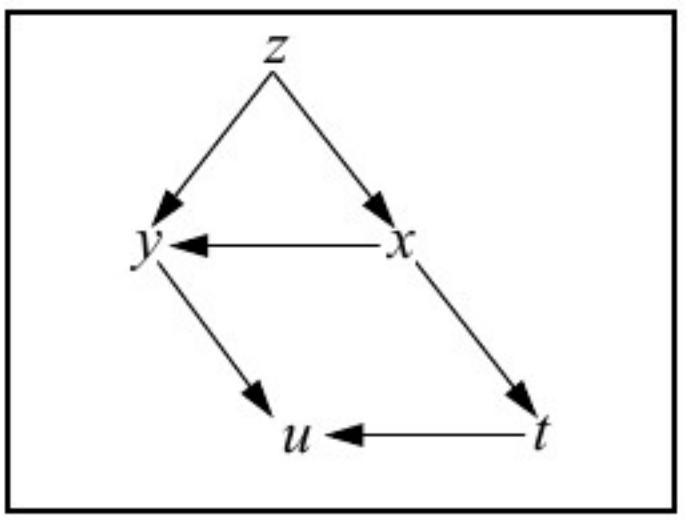


Figura 2.3

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

R^2 si può ottenere anche facendo il prodotto matriciale di R con se stessa.

R^3	x	y	z	t	u
x					
y					
z					1
t					
u					

…

La $R^* = \overset{*}{\Rightarrow}$ si ottiene facendo l'unione di queste relazioni, ossia raggruppando tutti gli 1 che compaiono.

R^*	x	y	z	t	u
x	1	1		1	1
y		1			1
z	1	1	1	1	1
t				1	1
u					1

2.5 Linguaggi generati da grammatiche e correttezza di una grammatica

<u>Definizione 2.13</u> (Linguaggio generato da una grammatica)

Sia $G = (X, V, S, P)$ una grammatica.

Il *linguaggio generato da G*, denotato con $L(G)$, è l'insieme delle stringhe di terminali derivabili dal simbolo di partenza S.

$$L(G) = \left\{ w \in X^* \;\middle|\; S \overset{*}{\underset{G}{\Rightarrow}} w \right\} \qquad \blacksquare$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Sono, dunque, stringhe di $L(G)$ le stringhe che:

a) consistono di soli terminali;

b) possono essere derivate da S in G.

Definizione 2.14 (Forma di frase)

Sia $G = (X, V, S, P)$ una grammatica. Una stringa w, $w \in (X \cup V)^*$, è una *forma di frase* di G se $S \overset{*}{\underset{G}{\Rightarrow}} w$. ■

Alle forme di frasi si applicano le stesse definizioni (es.: potenza) e gli stessi operatori (es.: concatenazione) dati per le stringhe.

Proposizione 2.1

Data una grammatica $G = (X, V, S, P)$, $L(G)$ è l'insieme delle forme di frase terminali (o *frasi*) di G.

Definizione 2.15 (Grammatiche equivalenti)

Due grammatiche G e G' si dicono *equivalenti* se generano lo stesso linguaggio, ossia se:

$$L(G) = L(G')$$ ■

Esempio 2.14

Sia $G = (X, V, S, P)$, ove

$$X = \{a,b\}, \qquad V = \{S\}, \qquad P = \left\{ S \overset{(1)}{\rightarrow} aSb, S \overset{(2)}{\rightarrow} ab \right\}$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Determiniamo $L(G)$.

$ab \in L(G)$ poiché $S \underset{(2)}{\Rightarrow} ab$.

Se numeriamo le produzioni, possiamo indicare la produzione usata immediatamente al di sotto del simbolo $\Rightarrow$.

$\underset{(n)}{\Rightarrow}$ $\equiv$ ho applicato la produzione n

$y \overset{k}{\Rightarrow} z$ $\equiv$ y produce z in k passi, dove k = lunghezza della derivazione.

$$a^2b^2 \in L(G) \text{ poiché } S \underset{(1)}{\Rightarrow} aSb \underset{(2)}{\Rightarrow} a^2b^2$$

$$a^3b^3 \in L(G) \text{ poiché } S \overset{3}{\Rightarrow} a^3b^3$$

$$\vdots$$

$$\{a^nb^n \mid n > 0\} \subseteq L(G)$$

Inoltre, qualsiasi derivazione da S in G produce frasi del tipo a^nb^n.

Dunque $L(G) \subseteq \{a^nb^n \mid n > 0\}$ e quindi $L(G) = \{a^nb^n \mid n > 0\}$.

Per rendere più concisa la descrizione di una grammatica, spesso ci limiteremo ad elencarne le produzioni, quando sia chiaro quale sia il simbolo di partenza e quali siano i terminali ed i nonterminali.

Inoltre, le produzioni con la stessa parte sinistra vengono accorpate attraverso l'uso del simbolo " | " (preso a prestito dalla BNF).

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Infine, ometteremo l'indicazione della grammatica dalla simbologia di derivazione e derivazione diretta ($\overset{*}{\underset{G}{\Rightarrow}}$ e $\underset{G}{\Rightarrow}$) quando sia chiaro dal contesto a quale grammatica si fa riferimento.

<u>Esempio 2.15</u>

Sia data la seguente grammatica:

$$S \to \overset{(1)}{A} \overset{(2)}{|B}, \quad A \to \overset{(3)}{aA} \overset{(4)}{|a}, \quad B \to \overset{(5)}{bB} \overset{(6)}{|b}$$

Determiniamo $L(G)$.

Non sappiamo se applicare $S \overset{(1)}{\to} A$ oppure $S \overset{(2)}{\to} B$ inizialmente. I meccanismi di costruzione di un linguaggio sono generalmente *non deterministici*, poiché può non essere univoca la sostituzione da operare ad una forma di frase se uno stesso *NT* si trova a sinistra di 2 o più produzioni, come illustrato in Figura 2.4.

Dunque, una grammatica è uno *strumento generativo* di un linguaggio perché, data una qualsiasi parola di quel linguaggio, possiamo risalire mediante le produzioni al simbolo di partenza della grammatica.

Viceversa, dato il simbolo di partenza di una grammatica, seguendo uno qualsiasi dei cammini dell'albero di derivazione, si produce una parola "valida" del linguaggio.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

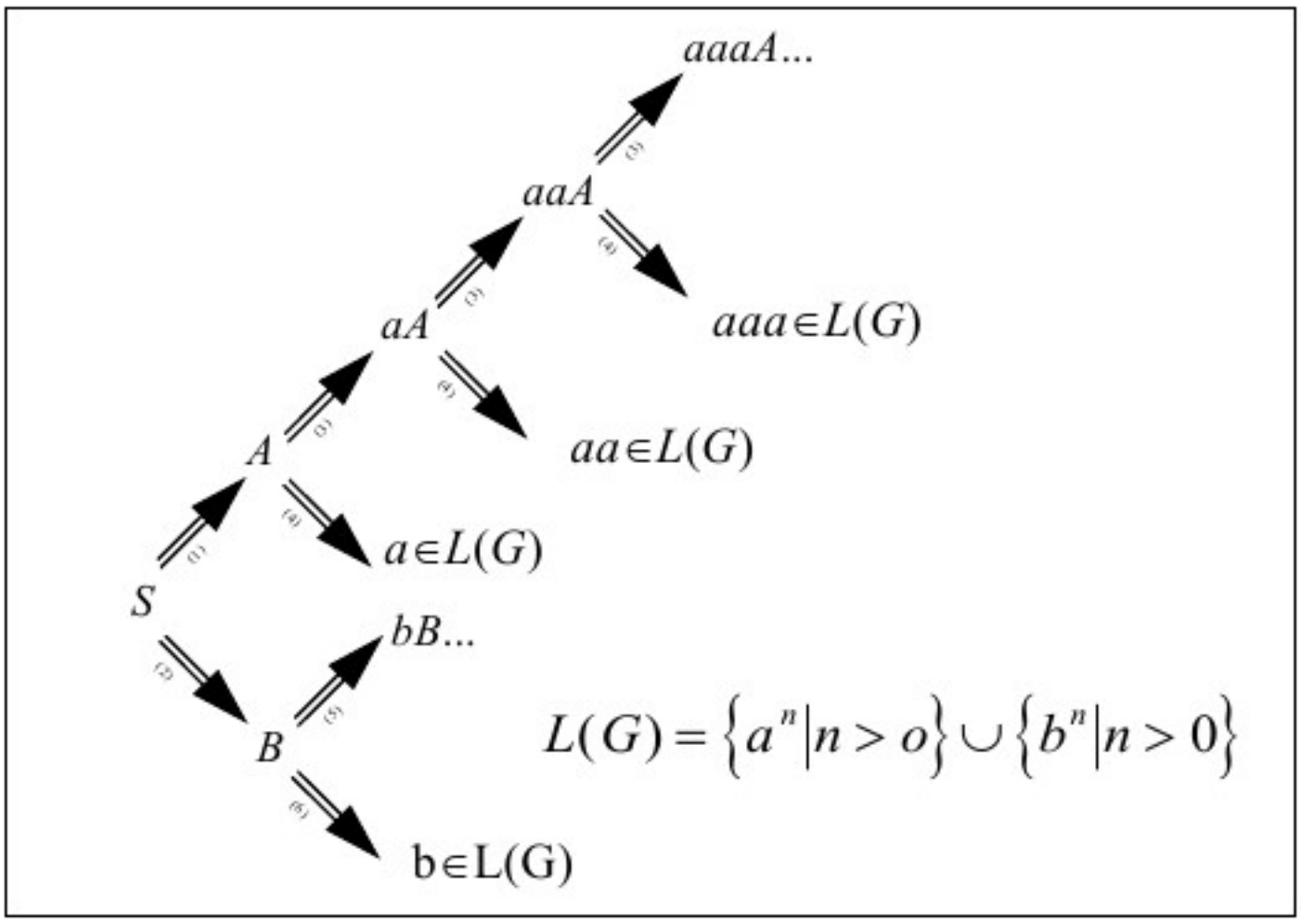


Figura 2.4

Osservazione 2.3

In generale, dato un linguaggio L ed una grammatica G, non esiste un algoritmo in grado di dimostrare che la grammatica genera il linguaggio, ossia che $L = L(G)$. Più specificamente, non esiste un algoritmo che stabilisce se una data stringa è generata o no dalla grammatica presa in considerazione.

Tutto ciò si riassume nella seguente:

Proposizione 2.2

Il problema di dimostrare la correttezza di una grammatica non è risolubile algoritmicamente, in generale.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

In molti casi importanti, però, è possibile dimostrare per induzione che una particolare grammatica genera proprio un particolare linguaggio.

Queste dimostrazioni ci consentono di stabilire se, data una grammatica G ed un linguaggio L, risulta:

1) $w \in L(G) \Rightarrow w \in L \quad (L(G) \subseteq L)$;[3]

2) $w \in L \Rightarrow w \in L(G) \quad (L \subseteq L(G))$;[4]

e dunque se $L = L(G)$.

2.6 Esercizi

Esercizio 2.1

1) Determinare la grammatica che genera il seguente linguaggio:

$$L = \{a^n b^n \mid n > 0\}$$

e dimostrare questo risultato.

2) Che tipo di grammatica genera L?

1)
$$G = (X, V, S, P)$$

$$X = \{a,b\}, \qquad V = \{S\}, \qquad P = \left\{S \underset{(1)}{\rightarrow} aSb,\ S \underset{(2)}{\rightarrow} ab\right\}$$

Dobbiamo dimostrare che $L = L(G)$.

i) $L \subset L(G)$

Sia w una stringa derivabile da S (in G).

---

[3] La grammatica G genera solo stringhe appartenenti al linguaggio L.

[4] Il linguaggio L comprende solo parole generabili dalla grammatica G.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

$$w \in L(G) \quad \overset{def}{\Leftrightarrow} \quad S \overset{*}{\Rightarrow} w, \;\; w \in X^*$$

Procediamo *per induzione sulla lunghezza della derivazione* di w da S. Denoto con n la lunghezza della derivazione di w da S.

<u>Passo base</u>

$n = 1$

$S \underset{(2)}{\overset{n}{\Rightarrow}} ab$ è l'unica derivazione di lunghezza $n = 1$ che genera stringhe di soli terminali.

È immediato verificare che $ab \in L$.

<u>Passo induttivo</u>

Dimostriamo che, per ogni $n > 1$, se supponiamo che il seguente enunciato è vero:

"se $w' \in L(G)$, $S \overset{n-1}{\Rightarrow} w'$ (w' è derivabile in $n-1$ passi da S)

allora $w' \in L$"

allora anche l'enunciato:

"se $w \in L(G)$, $S \overset{n}{\Rightarrow} w$ allora $w \in L$"

risulta vero.

Consideriamo:

$$w \in L(G), \text{ con } S \overset{n}{\Rightarrow} w.$$

Per definizione (di derivabilità in n passi), esiste una sequenza di forme di frase $w_1, w_2, ..., w_n$ con $w_n = w$, tale che

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

w_1 deriva direttamente da S e, per ogni i ($i = 1, 2, \ldots, n-1$), w_{i+1} deriva direttamente da w_i.

$$S \Rightarrow w_1 \Rightarrow w_2 \Rightarrow \ldots \Rightarrow w_n = w$$

È immediato osservare che il primo passo della derivazione è dato dalla applicazione della produzione (1) di G (altrimenti otterremmo la stringa ab, priva di nonterminali ed avremmo finito. Ma allora $n = 1$).

Si ha dunque:

$$S \underset{(1)}{\Rightarrow} aSb \overset{n-1}{\Rightarrow} w_n = w$$

Per ipotesi di induzione, ogni stringa derivabile da S in $n-1$ passi è una parola di L. Dunque, da S è possibile derivare in $n-1$ passi una stringa del tipo: $w' = a^k b^k$, $k > 0$.

Più precisamente, $w' = a^{n-1} b^{n-1}$, poiché:

$$S \overset{k}{\underset{(1)}{\Rightarrow}} a^k S b^k, \ k > 0 .$$

Ma allora la stringa:

$$aw'b = aa^{n-1}b^{n-1}b = a^n b^n$$

è ancora una stringa di L ed inoltre è derivabile da S in n passi.

Si ha, infatti:

$$S \underset{(1)}{\Rightarrow} aSb \overset{n-1}{\Rightarrow} aw'b = a^n b^n = w$$

Risulta così dimostrato $L(G) \subset L$.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

ii) $L \subset L(G)$

Sia w una parola di L. Procediamo *per induzione sulla lunghezza della stringa w*.

<u>Passo base</u>

Prendiamo in considerazione la parola di L di lunghezza minima.

$$n = 1 \Leftrightarrow |w| = 2 \qquad\qquad w = ab$$

Dobbiamo dimostrare che: $S \overset{*}{\Rightarrow} ab$.

Banale. Applichiamo la produzione (2) di G ed otteniamo che $w = ab$ è direttamente derivabile da S.

$$S \underset{(2)}{\Rightarrow} ab$$

<u>Passo induttivo</u>

Dimostriamo che, per ogni $n > 1$, se supponiamo che il seguente enunciato è vero:

$$\text{“se } w' \in L,\ |w'| = 2(n-1) \text{ allora } S \overset{*}{\Rightarrow} w'\text{”}$$

allora anche il seguente enunciato:

$$\text{“se } w \in L,\ |w| = 2n \text{ allora } S \overset{*}{\Rightarrow} w\text{”}$$

risulta vero.

Sia w una parola su X tale che:

$$w \in L,\ |w| = 2n,\ n > 1.$$

Ovviamente, $w = a^n b^n$ (unica parola di L di lunghezza $2n$). Nella (ipotetica) derivazione di w da S, devo

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

necessariamente applicare la produzione (1) di G, come 1° passo (altrimenti riotterrei la parola ab).

Dunque:

$$S \underset{(1)}{\Rightarrow} aSb \qquad \text{(a)}$$

Per ipotesi di induzione, ogni parola di L di lunghezza $2(n-1)$ è derivabile da S (in G). Dunque, anche $w' = a^{n-1}b^{n-1}$ è derivabile da S:

$$S \overset{*}{\Rightarrow} w' = a^{n-1}b^{n-1} \qquad \text{(b)}$$

Ne consegue che $w = a^n b^n$ è derivabile da S e la relativa derivazione è ottenuta applicando in successione (a) e (b).

$$S \underbrace{\underset{(1)}{\Rightarrow} aSb}_{(a)} \underbrace{\overset{*}{\Rightarrow} aw'b = aa^{n-1}b^{n-1}b = w}_{(b)}$$

Dunque, $L \subset L(G)$ e

$$L = L(G)$$

2) G è una grammatica libera da contesto.

<u>Esercizio 2.2</u>

Sia data la seguente grammatica:

$$G = (X, V, S, P)$$

$$X = \{0, 1\} \qquad V = \{S, A, B\}$$

$$P = \left\{ S \to \overset{(1)}{0B} \mid \overset{(2)}{1A},\ \ A \to \overset{(3)}{0} \mid \overset{(4)}{0S} \mid \overset{(5)}{1AA},\ \ B \to \overset{(6)}{1} \mid \overset{(7)}{1S} \mid \overset{(8)}{0BB} \right\}$$

Determinare il linguaggio generato da G e dimostrare il risultato.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Il linguaggio generato da G è il seguente:

$$L = \left\{ w \,\middle|\, w \in \{0, 1\}^+, w \text{ ha lo stesso numero di 0 e di 1} \right\}$$

dobbiamo dimostrare che:

$$L = L(G)$$

Quindi bisogna dimostrare che:

i) $L(G) \subset L$

ii) $L \subset L(G)$

i) Dimostriamo che $L(G) \subset L$. Questo corrisponde a dimostrare che G genera solo parole con lo stesso numero di 0 e di 1.

Procediamo *per induzione sulla lunghezza della stringa*. Possiamo ricondurre questa dimostrazione a dimostrare che:

i.a) ogni stringa derivabile da S in G possiede un numero uguale di 0 e di 1;

i.b) ogni stringa derivabile da A ha uno 0 in più;

i.c) ogni stringa derivabile da B ha un 1 in più.

Sia w una parola su $X = \{0, 1\}$.

<u>Passo base</u>

$|w| = 1$

i.a) Non vi sono parole di lunghezza 1 derivabili da S in G;

i.b) La sola parola di lunghezza 1 derivabile da A in G è $w = 0$

$$A \underset{(3)}{\Rightarrow} 0$$

e tale parola ha uno 0 in eccesso;

i.c) La sola parola di lunghezza 1 derivabile da B in G è $w = 1$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

$$B \underset{(6)}{\Rightarrow} 1$$

e tale parola ha un 1 in eccesso.

$|w| = 2$

i.a) Le sole parole di lunghezza 2 derivabili da S in G sono $w_1 = 01$ e $w_2 = 10$.

$$S \underset{(1)}{\Rightarrow} 0B \underset{(6)}{\Rightarrow} 01$$

$$S \underset{(2)}{\Rightarrow} 1A \underset{(3)}{\Rightarrow} 10$$

sia w_1 sia w_2 hanno lo stesso numero di 0 e di 1;

i.b) Non vi sono parole (stringhe di soli terminali) di lunghezza 2 derivabili da A in G;

$$A \underset{(3)}{\Rightarrow} 0 \qquad A \underset{(5)}{\Rightarrow} 1AA \underset{(3)}{\Rightarrow} 10A \qquad A \underset{(5)}{\Rightarrow} 1AA \underset{(3)}{\Rightarrow} 1A0$$

$$A \underset{(4)}{\Rightarrow} 0S \underset{(1)}{\Rightarrow} 00B \qquad A \underset{(5)}{\Rightarrow} 1AA \underset{(4)}{\Rightarrow} 10SA \qquad A \underset{(5)}{\Rightarrow} 1AA \underset{(4)}{\Rightarrow} 1A0S$$

$$A \underset{(4)}{\Rightarrow} 0S \underset{(2)}{\Rightarrow} 01A \qquad A \underset{(5)}{\Rightarrow} 1AA \underset{(5)}{\Rightarrow} 11AAA \qquad A \underset{(5)}{\Rightarrow} 1AA \underset{(5)}{\Rightarrow} 1A1AA$$

i.c) Non vi sono parole (stringhe di soli terminali) di lunghezza 2 derivabili da B in G;

$$B \underset{(6)}{\Rightarrow} 1 \qquad B \underset{(8)}{\Rightarrow} 0BB \underset{(6)}{\Rightarrow} 01B \qquad B \underset{(8)}{\Rightarrow} 0BB \underset{(6)}{\Rightarrow} 0B1$$

$$B \underset{(7)}{\Rightarrow} 1S \underset{(1)}{\Rightarrow} 10B \qquad B \underset{(8)}{\Rightarrow} 0BB \underset{(7)}{\Rightarrow} 01SB \qquad B \underset{(8)}{\Rightarrow} 0BB \underset{(7)}{\Rightarrow} 0B1S$$

$$B \underset{(7)}{\Rightarrow} 1S \underset{(2)}{\Rightarrow} 11A \qquad B \underset{(8)}{\Rightarrow} 0BB \underset{(8)}{\Rightarrow} 00BBB \qquad B \underset{(8)}{\Rightarrow} 0BB \underset{(8)}{\Rightarrow} 0B0BB$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Passo induttivo

Dimostriamo che, per ogni $n > 2$, se supponiamo che i.a), i.b) e i.c) siano vere per ogni parola di lunghezza m, con $2 \le m < n$, allora i.a), i.b) e i.c) risultano vere per parole di lunghezza n.

i.a) Sia w una stringa derivabile da S in G, tale che:

$$|w| = n \quad \text{e} \quad S \overset{*}{\Rightarrow} w$$

Si hanno due possibili casi:

$$S \underset{(1)}{\Rightarrow} 0B \overset{*}{\Rightarrow} w \ \text{ e } \ w = 0\beta, \quad \text{ove } \ B \overset{*}{\Rightarrow} \beta, \ |\beta| = n-1$$

$$S \underset{(2)}{\Rightarrow} 1A \overset{*}{\Rightarrow} w \ \text{ e } \ w = 1\alpha, \quad \text{ove } \ A \overset{*}{\Rightarrow} \alpha, \ |\alpha| = n-1$$

Per ipotesi di induzione i.b), α ha uno 0 in più. Tale 0 è bilanciato dall'1 iniziale in w. Ne segue che w ha lo stesso numero di 0 e di 1. Analogamente, per ipotesi di induzione i.c), β ha un 1 in eccesso, che viene bilanciato dallo 0 iniziale di w. Ne consegue che, anche in questo caso, w ha lo stesso numero di 0 e di 1.

i.b) Sia w una stringa di lunghezza n derivabile da A in G:

$$|w| = n \quad \text{e} \quad A \overset{*}{\Rightarrow} w$$

Si hanno due possibili casi:

$$A \underset{(4)}{\Rightarrow} 0S \overset{*}{\Rightarrow} w \ \text{ e } \ w = 0\beta, \quad \text{ove } \ S \overset{*}{\Rightarrow} \beta, \ |\beta| = n-1$$

$$A \underset{(5)}{\Rightarrow} 1AA \overset{*}{\Rightarrow} w \ \text{ e } \ w = 1\alpha, \quad \text{ove } \ AA \overset{*}{\Rightarrow} \alpha, \ |\alpha| = n-1$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Nel 1° caso, l'enunciato i.a) appena dimostrato ci garantisce che la stringa β possiede lo stesso numero di 0 e di 1. Dunque $w = 0\beta$ ha uno 0 in eccesso.

Nel 2° caso, per l'ipotesi di induzione i.b), la stringa α ha due 0 in eccesso (in quanto concatenazione di due stringhe derivabili da A). Di conseguenza, $w = 1\alpha$ ha uno 0 in eccesso.

i.c) Sia w una stringa di lunghezza n derivabile da B in G:

$$|w| = n \quad \text{e} \quad B \overset{*}{\Rightarrow} w\,.$$

Si hanno due possibili casi:

$$B \underset{(7)}{\Rightarrow} 1S \overset{*}{\Rightarrow} w \quad \text{e} \quad w = 1\alpha, \quad \text{ove} \quad S \overset{*}{\Rightarrow} \alpha,\ |\alpha| = n-1$$

$$B \underset{(8)}{\Rightarrow} 0BB \overset{*}{\Rightarrow} w \quad \text{e} \quad w = 0\beta, \quad \text{ove} \quad BB \overset{*}{\Rightarrow} \beta,\ |\beta| = n-1$$

Nel 1° caso, da i.a) sappiamo che α possiede lo stesso numero di 0 e di 1. Dunque $w = 1\alpha$ ha un 1 in eccesso.

Nel 2° caso, per l'ipotesi di induzione i.c), la stringa β ha due 1 in eccesso (in quanto concatenazione di due stringhe derivabili da B). Di conseguenza, $w = 0\beta$ ha uno 1 in eccesso.

Risulta così dimostrato il "teorema" i): $L(G) \subset L$

ii) Dimostriamo che $L \subset L(G)$. Questo corrisponde a dimostrare che "ogni parola con uguale numero di 0 e di 1 è generata da G".

Procediamo *per induzione sulla lunghezza della stringa*.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Possiamo ricondurre questa dimostrazione a dimostrare che:

i.a) ogni stringa con uguale numero di 0 e di 1 è derivabile da S in G;

i.b) ogni stringa con uno 0 in eccesso è derivabile da A (in G);

i.c) ogni stringa con un 1 in eccesso è derivabile da B (in G).

Sia w una parola su L (ossia, avente uno stesso numero di 0 e di 1).

<u>Passo base</u>

$|w| = 1$

ii.a) Non vi sono parole di lunghezza 1 che abbiano lo stesso numero di 0 e di 1;

ii.b) La sola parola di lunghezza 1 con uno 0 in eccesso è $w = 0$ ed è derivabile direttamente da A in G:

$$A \underset{(3)}{\Rightarrow} 0$$

ii.c) La sola parola di lunghezza 1 con un 1 in eccesso è $w = 1$ ed è derivabile da B in G:

$$B \underset{(6)}{\Rightarrow} 1$$

$|w| = 2$

ii.a) Le sole parole di lunghezza 2 che possiedono lo stesso numero di 0 e di 1 sono:

$$w_1 = 01 \text{ e } w_2 = 10$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Entrambe sono derivabili da S in G attraverso le seguenti derivazioni:

$$S \underset{(1)}{\Rightarrow} 0B \underset{(6)}{\Rightarrow} 01$$

$$S \underset{(2)}{\Rightarrow} 1A \underset{(3)}{\Rightarrow} 10$$

ii.b) Non esistono parole di lunghezza due con uno 0 in eccesso.

ii.c) Non esistono parole di lunghezza due con un 1 in eccesso.

<u>Passo induttivo</u>

Dimostriamo che, per ogni $n > 2$, se supponiamo che ii.a), ii.b) e ii.c) siano vere per ogni parola di lunghezza m, con $2 \leq m < n$, allora ii.a), ii.b) e ii.c) risultano vere per parole di lunghezza n.

ii.a) Sia w una parola di lunghezza n con lo stesso numero di 0 e di 1:

$$|w| = n$$

Supponiamo che w inizi con uno 0. Dunque, $w = 0\beta$, ove $|\beta| = n - 1$ e β ha un 1 in eccesso. Per ipotesi di induzione ii.c), β è derivabile da B in G:

$$B \overset{*}{\Rightarrow} \beta$$

Ma allora, w è derivabile da S in G e la relativa derivazione è:

$$S \underset{(1)}{\Rightarrow} 0B \overset{*}{\Rightarrow} 0\beta = w$$

titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Supponiamo che w inizi con un 1. Dunque, $w = 1\alpha$, ove $|\alpha| = n - 1$ e α ha uno 0 in eccesso.

Per ipotesi di induzione ii.b), α è derivabile da A in G:

$$A \overset{*}{\Rightarrow} \alpha$$

Ma allora, w è derivabile da S in G e la relativa derivazione è:

$$S \underset{(2)}{\Rightarrow} 1A \overset{*}{\Rightarrow} 1\alpha = w$$

ii.b) Sia w una parola di lunghezza n con uno 0 in eccesso:

$$|w| = n$$

w può avere una delle seguenti due forme:

1) $w = 0\beta$, ove $|\beta| = n - 1$ e β ha lo stesso numero di 0 e di 1

2) $w = 1\alpha$, ove $|\alpha| = n - 1$ e α ha due 0 in eccesso.

Se $w = 0\beta$, l'enunciato ii.a) vale per β e si ha:

$$S \overset{*}{\Rightarrow} \beta$$

Dunque, w è derivabile da A in G e la relativa derivazione è:

$$A \underset{(4)}{\Rightarrow} 0S \overset{*}{\Rightarrow} 0\beta = w$$

Se $w = 1\alpha$, nella derivazione di w da A (in G) che stiamo ricercando, il 1° passo consiste nell'applicazione della produzione (5)

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

$$A \to 1AA$$

Vogliamo dimostrare che:

$$AA \overset{*}{\Rightarrow} \alpha$$

ove α ha due 0 in eccesso. È sempre possibile considerare α come il risultato della concatenazione di due stringhe, α' e α'', entrambe con uno 0 in eccesso:

$$\alpha = \alpha'\alpha'' \text{ ove } |\alpha'| < n \text{ e } |\alpha''| < n .$$

Per ipotesi di induzione ii.b), si ha:

$$A \overset{*}{\Rightarrow} \alpha' \qquad \text{e} \qquad A \overset{*}{\Rightarrow} \alpha''$$

e dunque una derivazione di w da A è:

$A \underset{(5)}{\Rightarrow} 1AA \overset{*}{\Rightarrow} 1\alpha'A \overset{*}{\Rightarrow} 1\alpha'\alpha'' = 1\alpha = w$ (è l'unica derivazione di w da A?)

ii.c) Sia w una parola di lunghezza n con un 1 in eccesso:

$$|w| = n$$

w può avere una delle seguenti due forme:

1) $w = 1\alpha$, ove $|\alpha| = n - 1$ e α ha lo stesso numero di 0 e di 1.

2) $w = 0\beta$, ove $|\beta| = n - 1$ e β ha due 1 in eccesso.

Se $w = 1\alpha$, l'enunciato ii.a) vale per α e si ha:

$$S \overset{*}{\Rightarrow} \alpha$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Dunque, w è derivabile da B in G e la relativa derivazione è:

$$B \underset{(7)}{\Rightarrow} 1S \overset{*}{\Rightarrow} 1\alpha = w$$

Se $w = 0\beta$, nella derivazione di w da A (in G) che stiamo ricercando, il 1° passo consiste nell'applicazione della produzione (8)

$$B \to 0BB$$

Vogliamo dimostrare che:

$$BB \overset{*}{\Rightarrow} \beta$$

ove β ha due 1 in eccesso. È sempre possibile considerare β come il risultato della concatenazione di due stringhe, β' e β'', entrambe con un 1 in eccesso:

$$\beta = \beta'\beta'' \text{ ove } |\beta'| < n \text{ e } |\beta''| < n$$

Per ipotesi di induzione ii.c), si ha:

$$B \overset{*}{\Rightarrow} \beta' \qquad \text{e} \qquad B \overset{*}{\Rightarrow} \beta''$$

e dunque una derivazione di w da B è:

$$B \underset{(8)}{\Rightarrow} 0BB \overset{*}{\Rightarrow} 0\beta'B \overset{*}{\Rightarrow} 0\beta'\beta'' = 0\beta = w$$

Risulta così dimostrato il "teorema" ii): $L \subset L(G)$.

Si ha: $L = L(G)$.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Esercizio 2.3

1) Determinare la grammatica che genera il seguente linguaggio:

$$L = \{a^n b^{2n} \mid n > 0\}$$

e dimostrare questo risultato.

2) Di che tipo è la grammatica che genera L?

1)

$$G = (X, V, S, P)$$

$$X = \{a, b\} \qquad V = \{S\} \qquad P = \left\{ S \underset{(1)}{\rightarrow} aSbb,\ S \underset{(2)}{\rightarrow} abb \right\}$$

Dobbiamo dimostrare:

$$L = L(G)$$

Quindi bisogna dimostrare che:

i) $L(G) \subset L$

ii) $L \subset L(G)$

i) $L(G) \subset L$

Sia w una parola derivabile da S in G.

$$w \in L(G) \overset{def}{\Leftrightarrow} S \overset{*}{\Rightarrow} w, \quad w \in X^*$$

Procediamo *per induzione sulla lunghezza della derivazione* di w da S. Sia n tale lunghezza.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Passo base

$n = 1$

$S \underset{(2)}{\overset{n}{\Rightarrow}} abb$ è la sola derivazione di lunghezza 1 che genera parole su $X = \{a, b\}$

Si ha che: $abb \in L$.

Passo induttivo

Dimostriamo che:

$$\forall n,\ n > 1 : \left[\left(\left(w' \in L(G),\ S \overset{n-1}{\Rightarrow} w' \right) \Rightarrow w' \in L \right) \Rightarrow \right.$$
$$\left. \Rightarrow \left(\left(w \in L(G),\ S \overset{n}{\Rightarrow} w \right) \Rightarrow w \in L \right) \right]$$

Consideriamo:

$$w \in L(G), \text{ con } S \overset{n}{\Rightarrow} w$$

$$S \overset{n}{\Rightarrow} w \overset{def}{\Leftrightarrow} \exists\, w_1, w_2, \ldots, w_n : S \Rightarrow w_1,\ w_i \Rightarrow w_{i+1},$$
$$i = 1, 2, \ldots, n-1 \text{ e } w_n = w$$

$$S \Rightarrow w_1 \Rightarrow w_2 \Rightarrow \ldots \Rightarrow w_n = w$$

Necessariamente si ha: $w_1 = aSbb$ (altrimenti $w_1 = abb$ ed $n = 1$). Dunque:

$$S \underset{(1)}{\Rightarrow} aSbb \overset{n-1}{\Rightarrow} w_n = w$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Per ipotesi di induzione, ogni stringa derivabile da S in $n-1$ passi è una parola di L. Dunque, da S è possibile derivare in $n-1$ passi una stringa del tipo:

$$w' = a^k b^{2k}, \ k > 0$$

Più precisamente, $w' = a^{n-1} b^{2(n-1)}$ poiché:

$$S \underset{(1)}{\overset{k}{\Rightarrow}} a^k S b^{2k}, \ k > 0$$

Ma allora la stringa:

$$aw'bb = aa^{n-1} b^{2(n-1)} bb = a^n b^{2n}$$

è ancora una parola di L ed è derivabile da S in G in n passi, attraverso la seguente derivazione:

$$S \underset{(1)}{\overset{n-1}{\Rightarrow}} aSbb \Rightarrow aw'bb = a^n b^{2n} = w$$

Si ha dunque:

$$L(G) \subset L$$

ii) $L \subset L(G)$

Sia w una parola di L.

Procediamo *per induzione sulla lunghezza della parola w*.

<u>Passo base</u>

$n = 1 \ \Leftrightarrow \ |w| = 3$

Sia $w = abb$. Dobbiamo determinare una derivazione di w da S in G.

$$S \underset{(2)}{\Rightarrow} abb$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Passo induttivo

Dimostriamo che, per ogni $n > 1$:

$$\left(\left(w' \in L,\ |w'| = n-1+2(n-1 == 3n-3\right) \ \Rightarrow\ S \overset{*}{\Rightarrow} w'\right) \Rightarrow$$

$$\Rightarrow \left(\left(w \in L,\ |w| = n+2n == 3n\right) \ \Rightarrow\ S \overset{*}{\Rightarrow} w\right)$$

Sia w una parola su $X = \{a,b\}$ tale che:

$$w \in L, \quad |w| = 3n, \quad n > 1$$

L'unica parola di L di lunghezza $3n$ è:

$$w = a^n b^{2n}$$

Nella derivazione che stiamo cercando, dobbiamo necessariamente applicare la produzione (1) di G, come 1° passo:

$$S \underset{(1)}{\Rightarrow} aSbb$$

Per ipotesi di induzione, ogni parola di L di lunghezza $3n-3$ è derivabile da S in G. Anche $w' = a^{n-1}b^{2(n-1)}$ è dunque derivabile da S in G:

$$S \overset{*}{\Rightarrow} w' = a^{n-1}b^{2(n-1)}$$

Ne consegue che $w = a^n b^{2n}$ è derivabile da S e la sua derivazione è data da:

$$S \underset{(1)}{\Rightarrow} aSbb \overset{*}{\Rightarrow} aw'bb = aa^{n-1}b^{2(n-1)}bb = a^n b^{2n} = w$$

Dunque: $L \subset L(G)$ e $L = L(G)$.

2) G libera da contesto.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Esercizio 2.4

Dimostrare per induzione che il linguaggio L generato dalla seguente grammatica è vuoto:

$$G = (X, V, S, P)$$

$$X = \{a, b, c\} \qquad V = \{S, A, B\}$$

$$P = \left\{ S \to \overset{(1)}{aBS} \mid \overset{(2)}{bA},\ \ aB \to \overset{(3)}{Ac} \mid \overset{(4)}{a},\ \ bA \to \overset{(5)}{S} \mid \overset{(6)}{Ba} \right\}$$

Dobbiamo dimostrare che:

$$L(G) = \varnothing$$

È sufficiente dimostrare che $L(G) \subset \varnothing$, in quanto l'inclusione opposta è una tautologia ($\varnothing \subset L(G)$ è vera per ogni G).

Dimostrare che $L(G) \subset \varnothing$ significa dimostrare che, in qualsiasi passo di una derivazione da S, la stringa ottenuta presenta almeno un simbolo nonterminale. Cioè:

$$w \in L(G) \Rightarrow w \in \varnothing$$

dove $$L(G) = \left\{ w \in X^* \;\middle|\; S \overset{*}{\underset{G}{\Rightarrow}} w \right\}$$

e $$\forall n,\ S \overset{n}{\underset{G}{\Rightarrow}} w \ \Rightarrow\ w = \alpha N \beta,\ N \in V,\ \alpha, \beta \in (X \cup V)^*.$$

Procediamo *per induzione sulla lunghezza di una derivazione* da S. Denotiamo con n la lunghezza di una derivazione da S.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Passo base

$n = 1$

$S \overset{n}{\underset{(1)}{\Rightarrow}} aBS$ e $S \overset{n}{\underset{(2)}{\Rightarrow}} bA$ sono le uniche derivazioni possibili di lunghezza $n = 1$. Entrambe generano stringhe che presentano almeno un nonterminale.

Passo induttivo

Dimostriamo che, per ogni $n > 1$, se supponiamo che il seguente enunciato è vero:

$$\text{"se } S \overset{n-1}{\Rightarrow} w' \text{ allora } \exists\, N \in V : w' = yNz, \quad y, z \in (V \cup X)^* \text{"}$$

allora anche l'enunciato:

$$\text{"se } S \overset{n}{\Rightarrow} w \text{ allora } \exists\, N \in V : w = yNz, \quad y, z \in (V \cup X)^* \text{"}$$

risulta vero.

Consideriamo una qualunque derivazione in G costituita da n passi:

$$S \overset{n}{\Rightarrow} w$$

Per definizione (di derivabilità in n passi), esiste una sequenza di stringhe $w_1, w_2, ..., w_n$, con $w_n = w$, tale che w_1 deriva direttamente da S e, per ogni i, $i = 1, 2, \ldots, n-1$, w_{n+1} deriva direttamente da w_i. Dunque:

$$S \overset{n-1}{\Rightarrow} w_{n-1} \Rightarrow w_n = w$$

Per ipotesi di induzione, ogni stringa derivabile da S in $n-1$ passi presenta un nonterminale.

Dunque, anche $w_{n\text{-}1}$ presenta un nonterminale (o variabile).

Si hanno le seguenti possibilità:

- in $w_{n\text{-}1}$ compare il nonterminale S:

  allora in w abbiamo ancora un nonterminale, in quanto le uniche due produzioni in cui S compare nella parte sinistra sono la (1) $S \to aBS$ e la (2) $S \to bA$.

$$S \overset{n-1}{\Rightarrow} w_{n-1} = ySz \underset{(1)}{\Rightarrow} w_n = w = yaBSz$$

$$S \overset{n-1}{\Rightarrow} w_{n-1} = ySz \underset{(2)}{\Rightarrow} w_n = w = ybAz$$

- in $w_{n\text{-}1}$ compare il nonterminale A:

  allora se A è preceduto dal terminale b è possibile effettuare l'n-esimo passo della derivazione, altrimenti non esistono derivazioni di lunghezza n.

  Se $w_{n-1} = ybAz$, possiamo applicare le produzioni (5) e (6).

$$S \overset{n-1}{\Rightarrow} w_{n-1} = ybAz \underset{(5)}{\Rightarrow} w_n = w = ySz$$

$$S \overset{n-1}{\Rightarrow} w_{n-1} = ybAz \underset{(6)}{\Rightarrow} w_n = w = yBaz$$

  e in w abbiamo, in entrambi i casi, ancora un nonterminale.

- in $w_{n\text{-}1}$ compare il nonterminale B:

  se B non è preceduto dal terminale a non è possibile effettuare l'n-esimo passo della derivazione.

  Se $w_{n-1} = yaBz$, possiamo applicare le produzioni (3) e (4).

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

$$S \overset{n-1}{\Rightarrow} w_{n-1} = yaBz \underset{(3)}{\Rightarrow} w_n = w = yAcz$$

$$S \overset{n-1}{\Rightarrow} w_{n-1} = yaBz \underset{(4)}{\Rightarrow} w_n = w = yaz$$

Ora, se $y, z \in X^*$ allora $w \in X^*$ e dunque non possiamo provare la veridicità dell'enunciato.

Infatti, esiste (almeno) una derivazione di una stringa di soli terminali:

$$S \underset{(1)}{\Rightarrow} aBS \underset{(4)}{\Rightarrow} aS \underset{(2)}{\Rightarrow} abA \underset{(6)}{\Rightarrow} aBa \underset{(4)}{\Rightarrow} aa$$

Dunque: $L(G) \neq \varnothing$.

Esercizio 2.5

Si consideri il seguente linguaggio:

$$L = \{a^n b^k c^{2n} \mid n, k \geq 1\}$$

Determinare una grammatica G tale che $L = L(G)$ e dimostrare per induzione tale uguaglianza.

$$G = (X, V, S, P)$$

$$X = \{a, b, c\} \qquad V = \{S, B\}$$

$$P = \left\{ S \to \overset{(1)}{aScc} \mid \overset{(2)}{aBcc},\ \ B \to \overset{(3)}{bB} \mid \overset{(4)}{b} \right\}$$

Si intende dimostrare che $L = L(G)$.

Possiamo ricondurre questa dimostrazione alla dimostrazione del seguente asserto:

$$S \overset{*}{\underset{G}{\Rightarrow}} w \quad \Leftrightarrow \quad w \in I$$

ove:

$$I = \{a^n S c^{2n} \mid n \geq 1\} \cup \{a^n B c^{2n} \mid n \geq 1\} \cup$$
$$\cup \{a^n b^k B c^{2n} \mid n,k \geq 1\} \cup \{a^n b^k c^{2n} \mid n,k \geq 1\}$$

$\Rightarrow$) Procediamo per induzione sulla lunghezza della derivazione da S in G. Sia w una forma di frase derivabile da S in G e denoto con t la lunghezza della derivazione:

$$S \overset{t}{\underset{G}{\Rightarrow}} w$$

<u>Passo base</u>

$t = 1$

Le possibili derivazioni da S di lunghezza 1 sono:

$$S \underset{(1)}{\Rightarrow} aScc \quad \text{ed} \quad aScc \in I \quad (n = 1)$$
$$S \underset{(2)}{\Rightarrow} aBcc \quad \text{ed} \quad aBcc \in I \quad (n = 1)$$

<u>Passo induttivo</u>

Dimostriamo che, per ogni $t > 1$, se supponiamo che il seguente enunciato è vero:

$$"S \overset{t}{\underset{G}{\Rightarrow}} w \quad \Rightarrow \quad w \in I",$$

allora anche l'enunciato:

$$"S \overset{t+1}{\underset{G}{\Rightarrow}} w \quad \Rightarrow \quad w \in I"$$

risulta vero.

Consideriamo una generica forma di frase derivabile da S in G in t+1 passi e denotiamo tale forma di frase con w.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Per definizione di derivazione in t+1 passi, esiste una sequenza di forme di frase $w_1, w_2, \ldots, w_{t+2}$, con $w_i \in (X \cup V)^+$, $i = 1, 2, \ldots, t+2$, $w_1 = S$, $w_{t+2} = w$ tali che:

$$w_1 = S \underbrace{\Rightarrow w_2 \Rightarrow w_3 \Rightarrow \ldots \Rightarrow w_{t+1} \Rightarrow}_{t+1 \text{ passi}} w_{t+2} = w$$

Inoltre, per ipotesi di induzione, $w_{t+1} \in I$. Dunque, w_{t+1} è in una delle seguenti forme:

$$w_{t+1} = \begin{cases} a^n S c^{2n} \\ a^n B c^{2n} \\ a^n b^k B c^{2n} \\ a^n b^k c^{2n} \end{cases} \qquad k, n \geq 1$$

Analizziamo l'ultimo (il t+1-esimo) passo di derivazione:

$$S \overset{t}{\underset{G}{\Rightarrow}} w_{t+1}$$

Se $w_{t+1} = a^n S c^{2n}$, si hanno due possibili forme per w_{t+2}:

$$S \overset{t}{\Rightarrow} a^n S c^{2n} \underset{(1)}{\Rightarrow} a^{n+1} S c^{2n+2} = w_{t+2} \quad \text{e} \quad w_{t+2} \in I$$

$$S \overset{t}{\Rightarrow} a^n S c^{2n} \underset{(2)}{\Rightarrow} a^{n+1} B c^{2n+2} = w_{t+2} \quad \text{e} \quad w_{t+2} \in I$$

Se $w_{t+1} = a^n B c^{2n}$, si hanno due possibili forme per w_{t+2}:

$$S \overset{t}{\Rightarrow} a^n B c^{2n} \underset{(3)}{\Rightarrow} a^n b B c^{2n} = w_{t+2} \quad \text{e} \quad w_{t+2} \in I$$

$$S \overset{t}{\Rightarrow} a^n B c^{2n} \underset{(4)}{\Rightarrow} a^n b c^{2n} = w_{t+2} \quad \text{e} \quad w_{t+2} \in I$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Se $w_{t+1} = a^n b^k B c^{2n}$, si hanno due possibili forme per w_{t+2}:

$$S \overset{t}{\Rightarrow} a^n b^k B c^{2n} \underset{(3)}{\Rightarrow} a^n b^{k+1} B c^{2n} = w_{t+2} \quad \text{e} \quad w_{t+2} \in I$$

$$S \overset{t}{\Rightarrow} a^n b^k B c^{2n} \underset{(4)}{\Rightarrow} a^n b^{k+1} c^{2n} = w_{t+2} \quad \text{e} \quad w_{t+2} \in I$$

Se $w_{t+1} = a^n b^k c^{2n}$, si ha:

$$S \overset{t}{\Rightarrow} a^n b^k c^{2n}$$

e non ci sono ulteriori passi di derivazione possibili (non esistono derivazioni di lunghezza t+1 la cui t+1-esima forma di frase sia del tipo $w_{t+1} = a^n b^k c^{2n}$).

Risulta così dimostrato l'asserto.

$\Leftarrow$) Dimostriamo la seguente implicazione:

$$\left.\begin{array}{l} w \in I \\ I = I_1 \cup I_2 \cup I_3 \cup I_4 \\ I_1 = \{a^n S c^{2n} \mid n \geq 1\} \\ I_2 = \{a^n B c^{2n} \mid n \geq 1\} \\ I_3 = \{a^n b^k B c^{2n} \mid k, n \geq 1\} \\ I_4 = \{a^n b^k c^{2n} \mid k, n \geq 1\} \end{array}\right. \Rightarrow \quad S \overset{*}{\underset{G}{\Rightarrow}} w$$

Sia $w \in I_1 = \{a^n S c^{2n} \mid n \geq 1\}$.

Procediamo per induzione su n.

$n = 1 \qquad w = aScc$

e la derivazione cercata è:

$$S \underset{(1)}{\Rightarrow} aScc$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Supponiamo vera:

$$w' = a^{n-1}Sc^{2(n-1)} \;\Rightarrow\; S \overset{*}{\underset{G}{\Rightarrow}} w'$$

e dimostriamo che $w = a^n Sc^{2n}$ è derivabile da S in G.

Per ipotesi di induzione:

$$S \overset{*}{\Rightarrow} a^{n-1}Sc^{2(n-1)}$$

Dunque si ha:

$$S \overset{*}{\Rightarrow} a^{n-1}Sc^{2(n-1)} \underset{(1)}{\Rightarrow} a^{n-1}\ a\ S\ cc\ c^{2(n-1)} = a^n Sc^{2n}$$

Sia $w \in I_2 = \{a^n Bc^{2n} \mid n \geq 1\}$.

Procediamo *per induzione su n*.

$n = 1 \qquad w = aBcc$

e la derivazione cercata è:

$$S \underset{(2)}{\Rightarrow} aBcc$$

Sia $w = a^n Bc^{2n}$. Dal risultato precedente sappiamo che ($w' \in I_1 \;\Rightarrow\; S \overset{*}{\Rightarrow} w'$) con $w' = a^{n-1}Sc^{2(n-1)}$; si ha:

$$S \overset{*}{\Rightarrow} a^{n-1}Sc^{2(n-1)}$$

Dunque $w = a^n Bc^{2n}$ è derivabile da S in G attraverso la seguente derivazione:

$$S \overset{*}{\Rightarrow} a^{n-1}Sc^{2(n-1)} \underset{(2)}{\Rightarrow} a^{n-1}\ a\ B\ cc\ c^{2(n-1)} = a^n Bc^{2n}$$

Sia $w \in I_3 = \{a^n b^k Bc^{2n} \mid k, n \geq 1\}$.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Fissiamo un generico k e dimostriamo per induzione su n che $w = a^n b^k B c^{2n}$ è derivabile da S in G.

$n = 1 \qquad w = ab^k Bcc$

e la derivazione cercata è:

$$S \underset{(2)}{\Rightarrow} aBcc \overset{k}{\underset{(3)}{\Rightarrow}} ab^k Bcc$$

Sia $w = a^n b^k B c^{2n}$. Per ipotesi di induzione:

$$S \overset{*}{\Rightarrow} a^{n-1} b^k c^{2(n-1)}$$

Dunque $w = a^n b^k B c^{2n}$ è derivabile da S in G attraverso la seguente derivazione:

$$S \overset{*}{\Rightarrow} a^n B c^{2n} \overset{k}{\underset{(3)}{\Rightarrow}} a^n b^k B c^{2n}$$

in cui abbiamo utilizzato il risultato precedente ($w' \in I_2 \;\Rightarrow\; S \overset{*}{\Rightarrow} w'$ con $w' = a^n B c^{2n}$).

Fissiamo ora un generico n e dimostriamo per induzione su k che $w = a^n b^k B c^{2n}$ è derivabile da S in G.

$k = 1 \qquad w = a^n b^k B c^{2n}$ e la derivazione cercata è:

$$S \overset{*}{\Rightarrow} a^n B c^{2n} \underset{(3)}{\Rightarrow} a^n b B c^{2n}$$

Sia $w = a^n b^k B c^{2n}$. Per ipotesi di induzione, si ha:

$$S \overset{*}{\underset{G}{\Rightarrow}} a^n b^{k-1} B c^{2n}$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Pertanto, $w = a^n b^k B c^{2n}$ è derivabile da S in G e la derivazione è:

$$S \overset{*}{\Rightarrow} a^n b^{k-1} B c^{2n} \underset{(3)}{\Rightarrow} a^n b^k B c^{2n}$$

Sia $w \in I_4 = \{a^n b^k c^{2n} \mid k,n \geq 1\}$.

Fissiamo arbitrariamente il valore di k e dimostriamo per induzione su n che w è derivabile da S in G.

$n = 1$ $\quad w = ab^k Bc^2$

e la derivazione cercata è:

$$S \underset{(2)}{\Rightarrow} aBcc \overset{k-1}{\underset{(3)}{\Rightarrow}} ab^{k-1} Bcc \underset{(4)}{\Rightarrow} ab^k cc$$

Sia $w = ab^k Bc^2$. Per ipotesi di induzione:

$$S \overset{*}{\Rightarrow} a^{n-1} b^k c^{2(n-1)}$$

Dunque $w = a^n b^k c^{2n}$ è derivabile da S in G attraverso la seguente derivazione:

$$S \overset{*}{\Rightarrow} a^n b^{k-1} B c^{2n} \underset{(4)}{\Rightarrow} a^n b^k c^{2n}$$

in cui abbiamo utilizzato il risultato precedente ($w' \in I_3 \;\Rightarrow\; S \overset{*}{\Rightarrow} w'$ con $w' = a^n b^{k-1} B c^{2n}$).

Fissiamo ora un generico n e dimostriamo per induzione su k che $w = a^n b^k c^{2n}$ è derivabile da S in G.

$k = 1$ $\quad w = a^n bc^{2n}$ e la derivazione cercata è:

$$S \overset{*}{\Rightarrow} a^n B c^{2n} \underset{(4)}{\Rightarrow} a^n bc^{2n}$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

in cui abbiamo utilizzato il risultato $w' \in I_2 \;\Rightarrow\; S \overset{*}{\Rightarrow} w'$ con $w' = a^n B c^{2n}$.

Sia $w = a^n b^k c^{2n}$. Abbiamo già dimostrato che una forma di frase $a^n b^k B c^{2n}$ è derivabile da S in G per ogni valore di n e di k. Dunque anche $a^n b^{k-1} B c^{2n}$ è derivabile.

Ma allora anche $w = a^n b^k c^{2n}$ è derivabile e la sua derivazione è:

$$S \overset{*}{\Rightarrow} a^n b^{k-1} B c^{2n} \underset{(4)}{\Rightarrow} a^n b^k c^{2n}$$

Risulta così completata la dimostrazione dell'asserto:

$$w \in I \;\Rightarrow\; S \overset{*}{\underset{G}{\Rightarrow}} w$$

e dunque vale il risultato:

$$S \overset{*}{\underset{G}{\Rightarrow}} w \;\Leftrightarrow\; w \in I$$

per ogni forma di frase $w \in (X \cup V)^*$.

Poiché $L(G)$ è per definizione costituito di tutte e sole le <u>*frasi*</u> ($w \in X^*$) derivabili da S in G, si ha, come caso particolare, che:

$$S \overset{*}{\Rightarrow} w \;\Leftrightarrow\; w \in \{a^n b^k c^{2n} \mid n,k \geq 1\} = L\,.$$

3. Linguaggi liberi da contesto e linguaggi dipendenti da contesto

3.1 Grammatiche e linguaggi liberi da contesto

Definizione 3.1 (Grammatica libera da contesto)

Una grammatica $G = (X, V, S, P)$ è *libera da contesto* (o context-free - C.F.) se, per ogni produzione $v \to w$, v è un nonterminale.

$$G \text{ è libera da contesto } \overset{def}{\Leftrightarrow} \forall\, v \to w \in P : v \in V \qquad \blacksquare$$

Definizione 3.2 (Linguaggio libero da contesto)

Un linguaggio L su un alfabeto X è *libero da contesto* se può essere generato da una grammatica libera da contesto. [5]

$$L \text{ libero da contesto } \overset{def}{\Leftrightarrow} \exists G \text{ libera da contesto tale che } L(G) = L. \qquad \blacksquare$$

La maggior parte dei linguaggi di programmazione sono C.F.

Il termine C.F. nasce dal fatto che la sostituzione di un *NT* non è condizionata dal contesto - ossia dai caratteri adiacenti - in cui compare.

Un *NT A* in una forma di frase può sempre essere sostituito usando una produzione del tipo $A \to \beta$. La sostituzione è sempre valida.

---

[5] Se si ha una grammatica C.F. che genera L, non è detto che non esista un'altra grammatica che generi lo stesso linguaggio.

Viceversa, se $L = L(G)$e G non è C.F., non possiamo concludere che L non è C.F. perché non possiamo escludere che esista una grammatica C.F. G' per cui $L = L(G')$.

Esempi di linguaggi C.F.

1) Il linguaggio delle parentesi ben formate.
2) Il linguaggio dei numeri interi relativi.
3) Il linguaggio $\{a^n b^n \mid n > 0\}$.
4) Il linguaggio delle stringhe con ugual numero di 0 e di 1.
5) Il linguaggio $\{a^n b^{2n} \mid n > 0\}$.

3.2 Grammatiche e linguaggi dipendenti da contesto

Definizione 3.3 (Grammatica dipendente da contesto)

Una grammatica $G = (X, V, S, P)$ è *dipendente da contesto* (o context-sensitive - C.S.) se ogni produzione è in una delle seguenti forme:

(1) $yAz \to ywz$,

con $A \in V$, $y, z \in (X \cup V)^*$, $w \in (X \cup V)^+$,

che si legge: "A può essere sostituita con w nel contesto $y - z$" (contesto sinistro y e contesto destro z).

(2) $S \to \lambda$,

purché S non compaia nella parte destra di alcuna produzione.

■

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Definizione 3.4 (Linguaggio dipendente da contesto)
Un linguaggio L è *dipendente da contesto* se può essere generato da una grammatica dipendente da contesto. ■

3.3 Relazione tra linguaggi C.F. e C.S.

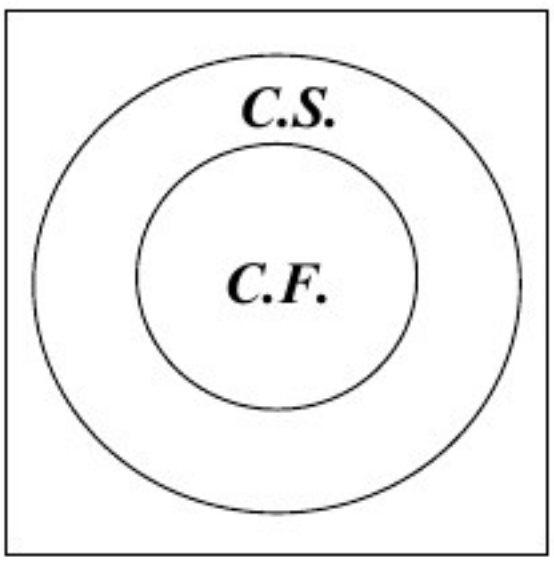


Figura 3.1

La relazione tra l'insieme dei linguaggi C.F. e l'insieme dei linguaggi C.S. è illustrato nella Figura 3.1.
Tale relazione sussiste perché le regole di produzione C.S. sono una generalizzazione di quelle C.F. Le produzioni C.F. sono un caso particolare delle produzioni di tipo (1) delle grammatiche C.S., che si verifica quando:

$y = z = \lambda$ contesto destro e sinistro equivalenti alla parola vuota.

con la seguente **eccezione**:
osservando con attenzione la definizione di grammatica C.F. si nota che $w \in (X \cup V)^*$, mentre nella definizione di grammatica

C.S. $w \in (X \cup V)^+$. Dunque le grammatiche C.F. ammettono produzioni del tipo $A \to \lambda$, con A che può anche non essere il simbolo iniziale, mentre le grammatiche C.S. non ammettono tali produzioni.

Chiameremo tutte le produzioni del tipo $A \to \lambda$ *λ-produzioni* o *λ-regole*.

Esempi di produzioni contestuali

$bC \to bc$

$baAChA \to baAabA$

Esempio di grammatica contestuale

$S \to \lambda \mid bC$

$bC \to bc$

$S \to \lambda$ è una produzione C.S. ed S non compare a destra di un'altra produzione.

Esempio di produzione non C.S. (né C.F.)

$CB \to BC$ non è né C.S. né C.F. È una produzione *monotona* perché del tipo $v \to w$ con $|v| \le |w|$

3.4 Grammatiche e linguaggi monotoni

Definizione 3.5 (Grammatica monotona)

Una grammatica $G = (X, V, S, P)$ è *monotona* se ogni sua produzione è monotona, cioè se $\forall v \to w \in P : |v| \le |w|$. ■

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Definizione 3.6 (Linguaggio monotono)

Un linguaggio L è *monotono* se può essere generato da una grammatica monotona. ■

Esempio 3.1

i) $AB \to CDEF$

ii) $CB \to BC$

sono produzioni monotone.

Una produzione monotona può essere sostituita da una sequenza di produzioni contestuali senza alterare il linguaggio generato.

Esempio 3.2

i) $AB \to CDEF$ può essere sostituita dalle seguenti produzioni contestuali:

i.1) $AB \to AG$

i.2) $AG \to CG$

i.3) $CG \to CDEF$

ii) $CB \to BC$ diventa:

ii.1)	$CB \to XB$		ii.1')	$CB \to X_1B$
ii.2)	$XB \to XC$	oppure	ii.2')	$X_1B \to X_1X_2$
ii.3)	$XC \to BC$		ii.3')	$X_1X_2 \to X_1C$
			ii.4')	$X_1C \to BC$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Difatti, vale la seguente:

Proposizione 3.1

La classe dei linguaggi contestuali coincide con la classe dei linguaggi monotoni.

Tale proposizione deriva immediatamente dal seguente risultato:

Teorema 3.1

Sia G una grammatica monotona, cioè tale che ogni produzione di G è della forma $v \to w$, con $|v| \leq |w|$, eccetto che ci può essere un'unica λ-produzione $S \to \lambda$ se S non appare alla destra di una produzione. Esiste allora una grammatica C.S. G' equivalente a G, cioè tale che $L(G) = L(G')$.

Il teorema precedente può essere enunciato anche in una forma leggermente differente:

Teorema 3.2

Un linguaggio L è dipendente da contesto se e solo se esiste una grammatica G tale che $L = L(G)$ ed ogni produzione di G nella forma

$$u \to v$$

ha la proprietà che: $0 < |u| \leq |v|$, con una sola eccezione: se $\lambda \in L(G)$ allora $S \to \lambda$ è una produzione di G ed in tal caso S non può comparire nella parte destra di altre produzioni.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Dimostrazione 3.2

$\Rightarrow$) Banale.

Se L è dipendente da contesto allora, per definizione, esiste G dipendente da contesto tale che $L = L(G)$.

$$L \text{ è C.S.} \overset{def}{\Leftrightarrow} \exists G \text{ C.S.}: \; L = L(G).$$

Allora ogni produzione di G è in una delle due forme:

(1) $yAz \to ywz$,

con $A \in V$, $y, z \in (X \cup V)^*$, $w \in (X \cup V)^+$,

(2) $S \to \lambda$,

con S che non compare nella parte destra di alcuna produzione.

Dunque, ogni produzione di G verifica la condizione $u \to v$, con $0 < |u| \le |v|$, se è del tipo (1), mentre se è del tipo (2) con S che non compare a destra di alcuna produzione, ricade nell'eccezione. Pertanto G è la grammatica cercata.

$\Leftarrow$) Sia G una grammatica in cui ogni produzione è nella forma $u \to v$, con $0 < |u| \le |v|$. Senza ledere la generalità della dimostrazione, possiamo supporre che una generica produzione di G abbia il formato: $A_1 A_2 ... A_m \to B_1 B_2 ... B_n$ $m \le n$ ove $A_i \in V$, $i = 1, 2, ..., m$.

È legittimo fare questa assunzione in quanto, se A_j fosse un terminale potremmo sostituirlo nella produzione con un

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

nuovo nonterminale A'_j ed aggiungere la nuova produzione $A'_j \to A_j$.

Denotiamo con C_1, C_2, …, C_m m simboli nonterminali non presenti in G. Utilizziamo le C_k, $k = 1, 2, \ldots, m$ per costruire nuove regole contestuali che riscrivono la stringa $A_1A_2...A_m$ con $B_1B_2...B_n$.

Le nuove regole sono:

$$\left.\begin{array}{l} A_1A_2...A_m \to C_1A_2...A_m \\ C_1A_2...A_m \to C_1C_2A_3...A_m \\ ... \\ C_1C_2...C_{m-1}A_m \to C_1C_2...C_{m-1}C_mB_{m+1}...B_n \\ C_1C_2...C_{m-1}C_mB_{m+1}...B_n \to C_1...C_{m-1}B_mB_{m+1}...B_n \\ ... \\ C_1B_2...B_n \to B_1B_2...B_n \end{array}\right\} \begin{array}{c} 2m \\ \text{produzioni} \end{array}$$

La nuova grammatica G' che incorpora queste produzioni è contestuale e si può dimostrare che

$$L(G) = L(G') .$$

Lasciamo per esercizio tale dimostrazione. *c.v.d.*

<u>Esempio 3.3</u>

Consideriamo il linguaggio:

$$L = \{a^n b^n c^n \mid n > 0\}$$

Determiniamo una grammatica che genera tale linguaggio.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Il linguaggio somiglia ad un linguaggio già visto, $\{a^n b^n \mid n > 0\}$, generato dalla grammatica $S \to aSb \mid ab$.

Dunque le produzioni saranno del tipo:

$$S \to \overset{(1)}{aSBC} \mid \overset{(2)}{aBC}$$

il NT B per generare le b

il NT C per generare le c

Se applichiamo una volta la (1) e poi la (2), abbiamo però:

$$S \underset{(1)}{\Rightarrow} aSBC \underset{(2)}{\Rightarrow} aaBCBC$$

che non è nella forma desiderata in quanto le b e le c risulterebbero alternate.

Generalizzando, se applichiamo n-1 volte la (1) e poi la (2), si ha:

$$S \overset{n-1}{\underset{(1)}{\Rightarrow}} a^{n-1} S \underbrace{BCBC...BC}_{n-1} \underset{(2)}{\Rightarrow} a^n \underbrace{BCBC...BC}_{n} = a^n (BC)^n$$

Abbiamo dunque bisogno di una produzione che riporti le B e le C in posizione corretta:

$$CB \overset{(3)}{\to} BC$$

con cui:

$$S \underset{(1)}{\Rightarrow} aSBC \underset{(2)}{\Rightarrow} aaBCBC \underset{(3)}{\Rightarrow} aaBBCC = a^2 B^2 C^2$$

e

$$S \overset{n-1}{\underset{(1)}{\Rightarrow}} a^{n-1} S (BC)^{n-1} \underset{(2)}{\Rightarrow} a^n \underbrace{BCBC...BC}_{n} \underset{(3)}{\Rightarrow} a^n BBCC \underbrace{BC...BC}_{n-2} \underset{(3)}{\Rightarrow}$$

$$\underset{(3)}{\Rightarrow} a^n BBCBCC...BC \underset{(3)}{\Rightarrow} a^n BBBCCC \underbrace{BC...BC}_{n-3} \overset{*}{\underset{(3)}{\Rightarrow}} a^n B^n C^n$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Quante volte abbiamo applicato la (3)? $\sum_{i=1}^{n-1}(n-i)$ volte.

Ora abbiamo bisogno delle produzioni che generano i terminali b e c.

Consideriamo la produzione $B \to b$. Non va bene. Perché? Perché altrimenti potremmo applicarla all'inizio di una derivazione, ottenendo stringhe del tipo:

$$a^n bCbC...bC$$

che impediscono di applicare la (3) (ed hanno bisogno di ulteriori produzioni per scambiare le b con le C) e quindi di trasformare le B in b solo dopo che le B hanno raggiunto la posizione corretta. Dunque dobbiamo considerare produzioni contestuali che trasformino le B in b solo dopo che hanno raggiunto la posizione corretta:

$aB \xrightarrow{(4)} ab$ per la prima occorrenza delle B

$bB \xrightarrow{(5)} bb$ per le restanti occorrenze delle B

Analogamente per le C:

$bC \xrightarrow{(6)} bc$ per la prima occorrenza delle C

$cC \xrightarrow{(7)} cc$ per le restanti occorrenze delle C

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Quindi, la grammatica G che genera $L = \{a^n b^n c^n \mid n > 0\}$ è:

$$S \xrightarrow{(1)} aSBC \mid \overset{(2)}{aBC}$$

$$CB \xrightarrow{(3)} BC$$

$$aB \xrightarrow{(4)} ab$$

$$bB \xrightarrow{(5)} bb$$

$$bC \xrightarrow{(6)} bc$$

$$cC \xrightarrow{(7)} cc$$

La grammatica è monotona, ma è facilmente determinabile una grammatica C.S. equivalente.

$$CB \xrightarrow{(3)} BC \qquad\qquad CB \xrightarrow{(3.a)} XB$$

$$XB \xrightarrow{(3.b)} XC$$

$$XC \xrightarrow{(3.c)} BC$$

La grammatica è comunque *non deterministica* perché non è univoca la sostituzione da operare in una forma di frase, come evidenziato dal successivo esempio.

<u>Esempio 3.5</u>

$$S \underset{(1)}{\Rightarrow} aSBC \underset{(2)}{\Rightarrow} aaBCBC \underset{(4)}{\Rightarrow} aabCBC \underset{(6)}{\Rightarrow} a^2bcBC$$

e a questo punto non possiamo più andare avanti nella derivazione perché non ci sono produzioni che possiamo applicare.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Si può dimostrare formalmente che questa grammatica genera solo stringhe di tipo $a^n\ b^n\ c^n$.

$$L = L(G)$$

Non forniamo l'intera dimostrazione, ma ci limitiamo ad osservare che:

1) $L \subseteq L(G)$ in quanto

$S \overset{n-1}{\underset{(1)}{\Rightarrow}} a^{n-1}S(BC)^{n-1}$ usando la (1) n-1 volte

$a^{n-1}S(BC)^{n-1} \underset{(2)}{\Rightarrow} a^n(BC)^n$ usando la (2) 1 volta

$a^n(BC)^n \overset{f(n)}{\underset{(3)}{\Rightarrow}} a^nB^nC^n$ usando la (3) $f(n)$ volte con

$$f(n) = \begin{cases} f(0) = 1 \\ f(n) = f(n-1) + n - 1 \end{cases}$$

o anche $f(n) = \sum_{i=1}^{n-1}(n-i)$

$a^nB^nC^n \underset{(4)}{\Rightarrow} a^nbB^{n-1}C^n$ usando la (4) 1 volta

$a^nbB^{n-1}C^n \overset{n-1}{\underset{(5)}{\Rightarrow}} a^nb^nC^n$ usando la (5) n-1 volte

$a^nb^nC^n \underset{(6)}{\Rightarrow} a^nb^ncC^{n-1}$ usando la (6) 1 volta

$a^nb^ncC^{n-1} \overset{n-1}{\underset{(7)}{\Rightarrow}} a^nb^nc^n$ usando la (7) n-1 volte

Dunque:

$$\forall n,\ n > 0 : a^nb^nc^n \in L(G)$$

2) $L(G) \subseteq L$ per esercizio.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Dunque: $L = \{a^n b^n c^n \mid n > 0\}$ è un linguaggio C.S. (e monotono), per il teorema che stabilisce una relazione tra grammatiche C.S. e grammatiche monotone, mentre $L = \{a^n b^n \mid n > 0\}$ è C.F.

3.5 Esercizi proposti

Esercizio 1

Sia dato il linguaggio:

$$L = \{a^m b^{3m} c^{2n} : m, n \geq 1\}.$$

Determinare una grammatica libera da contesto G che genera L. Dimostrare formalmente che G è corretta per L ($L = L(G)$).

Esercizio 2

Stabilire se il seguente linguaggio è libero da contesto e giustificare formalmente la risposta:

$$L = \{a^n w : w \in \{b, c\}^*, |w| = n, n > 0\}.$$

76 - LINGUAGGI LIBERI DA CONTESTO E LINGUAGGI DIPENDENTI DA CONTESTO

4. Linguaggi liberi da contesto

4.1 Alberi di derivazione

Le derivazioni in una grammatica libera da contesto possono essere rappresentate da *alberi* (detti *alberi di derivazione*).
La sequenza delle regole sintattiche utilizzate per generare una stringa w da una grammatica G di simbolo iniziale S definisce la *struttura* di w, che dunque potrebbe essere rappresentata da una delle derivazioni $S \overset{*}{\Rightarrow} w$. Al fine di disporre di una rappresentazione univoca, si preferisce ricorrere agli alberi di derivazione.

<u>Definizioni preliminari</u>

Un albero è un grafo orientato, aciclico, connesso e avente al massimo un lato entrante in ciascun nodo.
La definizione rigorosa di "albero" è la seguente:

<u>Definizione 4.1</u> (Albero)

Un *albero* T è un insieme finito, non vuoto, di vertici (detti *nodi*) tali che:

- esiste un vertice speciale, detto *radice* dell'albero;
- esiste una partizione $T_1, T_2, \ldots, T_m$, con $m \geq 0$, degli altri vertici, tale che esista un ordinamento $T_1, T_2, \ldots, T_m$, e ogni sottoinsieme di vertici T_i, $1 \leq i \leq m$, sia a sua volta un albero. ■

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Questa definizione è di tipo ricorsivo, ma non è circolare in quanto ogni sottoalbero T_i contiene meno vertici dell'albero T.
$T_1, T_2, ..., T_m$ sono detti *sottoalberi* di T.
I vertici privi di discendenti sono le *foglie* dell'albero, gli altri sono i *nodi interni*.
La stringa dei simboli che etichettano le foglie di un albero T, letti nell'ordine da sinistra a destra, prende il nome di *frontiera* di T.

Definizione 4.2 (Partizione)
Gli insiemi (non vuoti) $T_1, T_2, ..., T_m$, $m \geq 0$, costituiscono una *partizione* di un insieme (non vuoto) T se:

i) $T_i \cap T_j = \varnothing$ per $i \neq j,\ i, j = 1, 2, ..., m$;

ii) $\bigcup_{i=1}^{m} T_i = T$ ■

Definizione 4.3 (Lunghezza di un cammino)
Dato un albero di derivazione, la *lunghezza di un cammino* dalla radice ad una foglia è data dal numero di nonterminali su quel cammino. ■

Definizione 4.4 (Altezza o profondità)
L'*altezza* di un albero è data dalla lunghezza del suo cammino più lungo. ■

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

<u>Definizione 4.5</u> (Albero di derivazione)

Sia $G = (X, V, S, P)$ una grammatica libera da contesto e $w \in X^*$ una stringa derivabile da S in G, $S \overset{*}{\Rightarrow} w$.

Dicesi *albero di derivazione* l'albero T avente le seguenti proprietà:

(1) la radice è etichettata con il simbolo iniziale S;

(2) ogni nodo interno (nodo non foglia) è etichettato con un simbolo di V (un nonterminale);

(3) ogni nodo foglia è etichettato con un simbolo di X (un terminale) o con λ;

(4) se un nodo N è etichettato con A, ed N ha k discendenti diretti $N_1, N_2, ..., N_k$, etichettati con i simboli $A_1, A_2, ..., A_k$, rispettivamente, allora la produzione:

$$A \to A_1 A_2 ... A_k$$

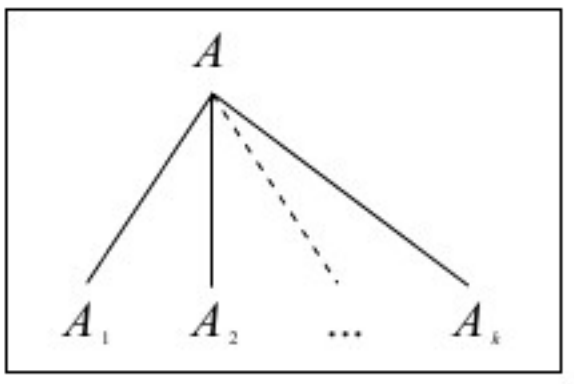


Figura 4.1

deve appartenere a P (Figura 4.1);

(5) la stringa w può essere ottenuta leggendo (e concatenando) le foglie dell'albero da sinistra a destra. ■

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Osservazione 4.1

Un albero di derivazione non impone alcun ordine sull'applicazione delle produzioni in una derivazione. In altri termini, *data una derivazione, esiste uno ed un solo albero di derivazione* che la rappresenta, mentre *un albero di derivazione rappresenta in generale più derivazioni* (in funzione dell'ordine col quale si espandono i nonterminali).

Esempio 4.1

Si consideri la seguente grammatica libera da contesto:

$$G_1 = (X, V, S, P)$$

dove:

$$X = \{a\} \qquad V = \{S, H\}$$

$$P = \left\{ S \overset{(1)}{\rightarrow} Ha,\ \ H \overset{(2)}{\rightarrow} HS,\ \ H \overset{(3)}{\rightarrow} a \right\}$$

La stringa $w = aaaa$ è derivabile da S in G_1 L'albero di derivazione di $w = aaaa$ è rappresentato in Figura 4.2.

Tale albero corrisponde sia alla *derivazione sinistra*:

$$S \underset{(1)}{\Rightarrow} Ha \underset{(2)}{\Rightarrow} HSa \underset{(3)}{\Rightarrow} aSa \underset{(1)}{\Rightarrow} aHaa \underset{(3)}{\Rightarrow} aaaa$$

sia alla *derivazione destra*:

$$S \underset{(1)}{\Rightarrow} Ha \underset{(2)}{\Rightarrow} HSa \underset{(1)}{\Rightarrow} HHaa \underset{(3)}{\Rightarrow} Haaa \underset{(3)}{\Rightarrow} aaaa$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

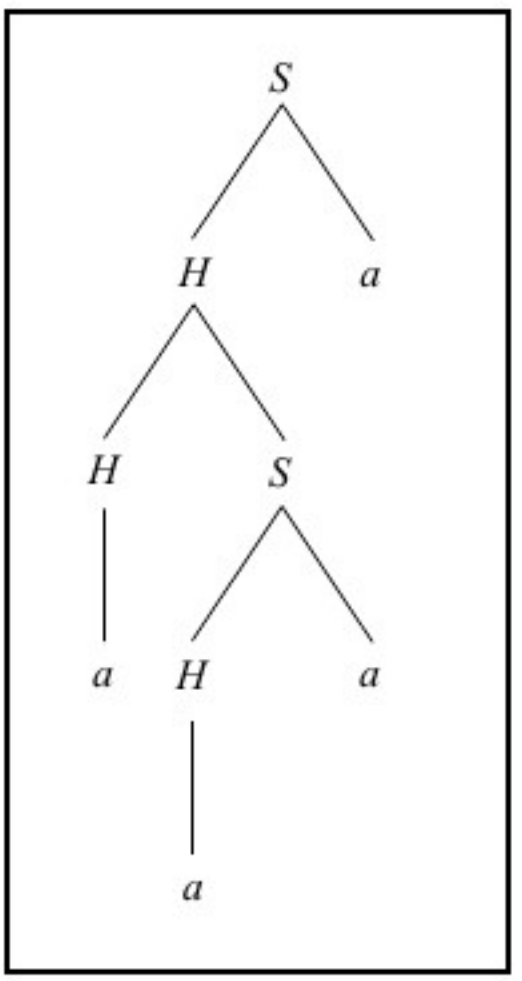


Figura 4.2

4.2 Principio di sostituzione di sottoalberi

<u>Definizione 4.6</u> (Derivazione destra/sinistra)

Data una grammatica $G = (X, V, S, P)$, diremo che una *derivazione* $S \overset{*}{\Rightarrow} w$, ove:

$$S \Rightarrow w_1 \Rightarrow w_2 \Rightarrow \ldots \Rightarrow w_n = w$$

$$w_i = y_i A z_i, \quad w_{i+1} = y_i w_i z_i, \quad i = 1, 2, \ldots, n-1$$

è *destra* (*sinistra*) se, per ogni i, $i = 1, 2, \ldots, n-1$, risulta:

$$z_i \in X^* \quad (y_i \in X^*)$$

In altre parole in una *derivazione destra* (*sinistra*) ad ogni passo si espande il nonterminale posto più a destra (sinistra). ■

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Esempio 4.2

Riconsideriamo la grammatica dell'Esercizio 2.2:

$$S \to 0B \mid 1A$$
$$A \to 0 \mid 0S \mid 1AA$$
$$B \to 1 \mid 1S \mid 0BB$$

che genera tutte e sole le stringhe che hanno un ugual numero di 1 e di 0.

Consideriamo la stringa 0011.

Una possibile derivazione è:

$$S \Rightarrow 0B \Rightarrow 00BB \Rightarrow 001B \Rightarrow 0011$$

Un'altra derivazione della stessa stringa è:

$$S \Rightarrow 0B \Rightarrow 00BB \Rightarrow 00B1 \Rightarrow 0011$$

Il non determinismo insito nella derivazione scompare quando si considera il relativo albero di derivazione (Figura 4.3).

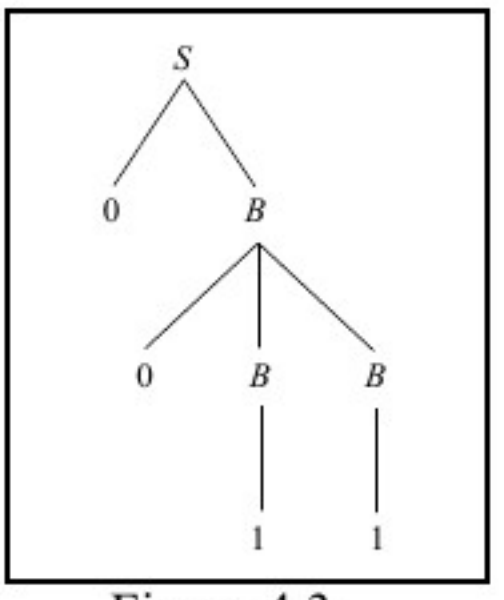


Figura 4.3

Consideriamo ora il sottoalbero con radice nel nodo di profondità minore etichettato con una B (ed individuato da un cerchio tratteggiato in Figura 4.4).

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

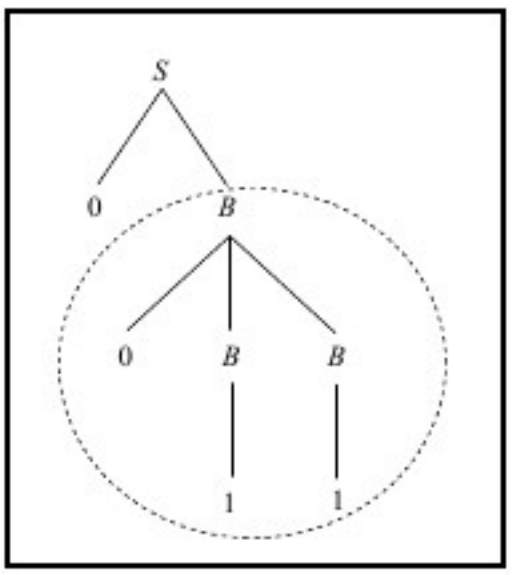


Figura 4.4

Cosa accade se un qualsiasi sottoalbero con radice in un nodo etichettato con una *B* viene sostituito con il sottoalbero ?
Il nuovo albero è ancora un albero di derivazione (ovviamente, per una stringa differente da 0011).
Consideriamo, ad esempio, il sottoalbero individuato dal cerchio pieno in Figura 4.5.

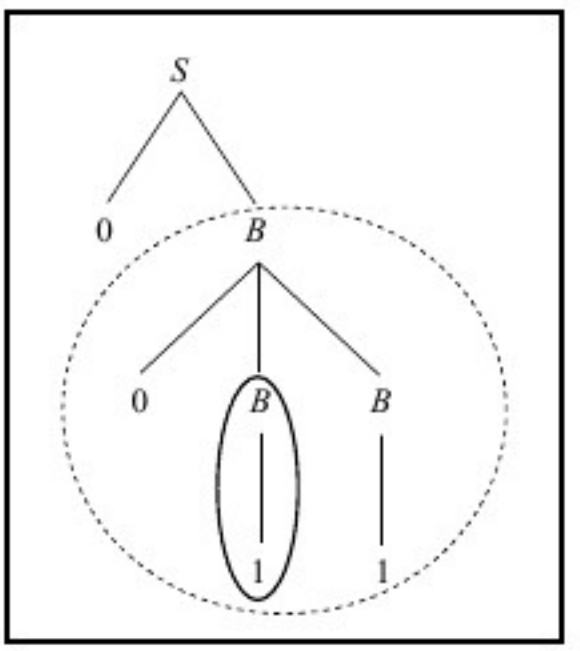


Figura 4.5

Il nuovo albero di derivazione è dato in Figura 4.6.

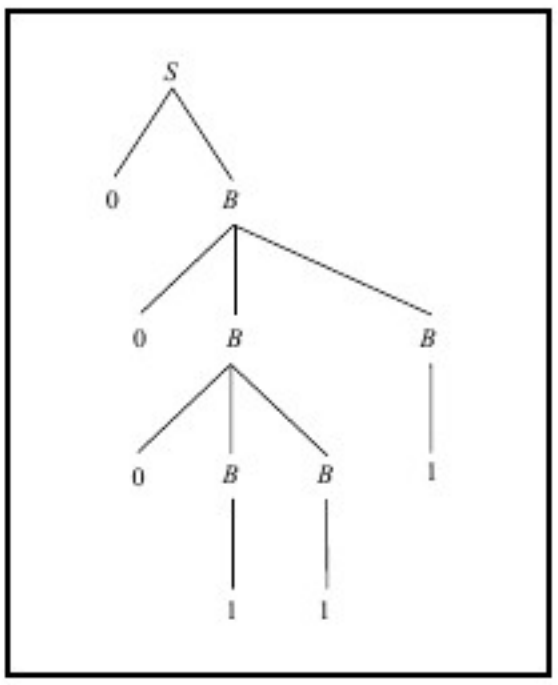


Figura 4.6

per cui:

$$S \overset{*}{\Rightarrow} 000111$$

Possiamo ripetere indefinitamente questo processo di sostituzione di sottoalberi, ottenendo parole del linguaggio di lunghezza crescente:

$$\begin{array}{ll} 0011 & |w| = 4 \\ 000111 & |w| = 6 \\ w = 00001111 & |w| = 8 \\ \vdots & \vdots \\ 00^n11^n & |w| = 2n+2 \quad n = 1, 2, \ldots \end{array}$$

La lunghezza cresce in maniera costante.

Nelle grammatiche C.F., dunque, possiamo sostituire alberi più piccoli con alberi di dimensioni maggiori, purché abbiano la stessa radice - più precisamente, purché i nodi radice siano etichettati con lo stesso *NT* - ottenendo ancora alberi di derivazione validi.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Una caratteristica dei linguaggi C.F., che discende direttamente dal processo descritto in precedenza, è che la lunghezza delle parole cresce in maniera costante.

Dunque, se una grammatica genera parole la cui lunghezza cresce in maniera esponenziale, allora il linguaggio generato non è libero.

Generalizziamo ora il discorso.

Supponiamo di avere un albero di derivazione T_z per una stringa z di terminali generata da una grammatica C.F. G, e supponiamo inoltre che il simbolo NT A compaia due volte su uno stesso cammino.

La situazione è rappresentata graficamente in Figura 4.7.

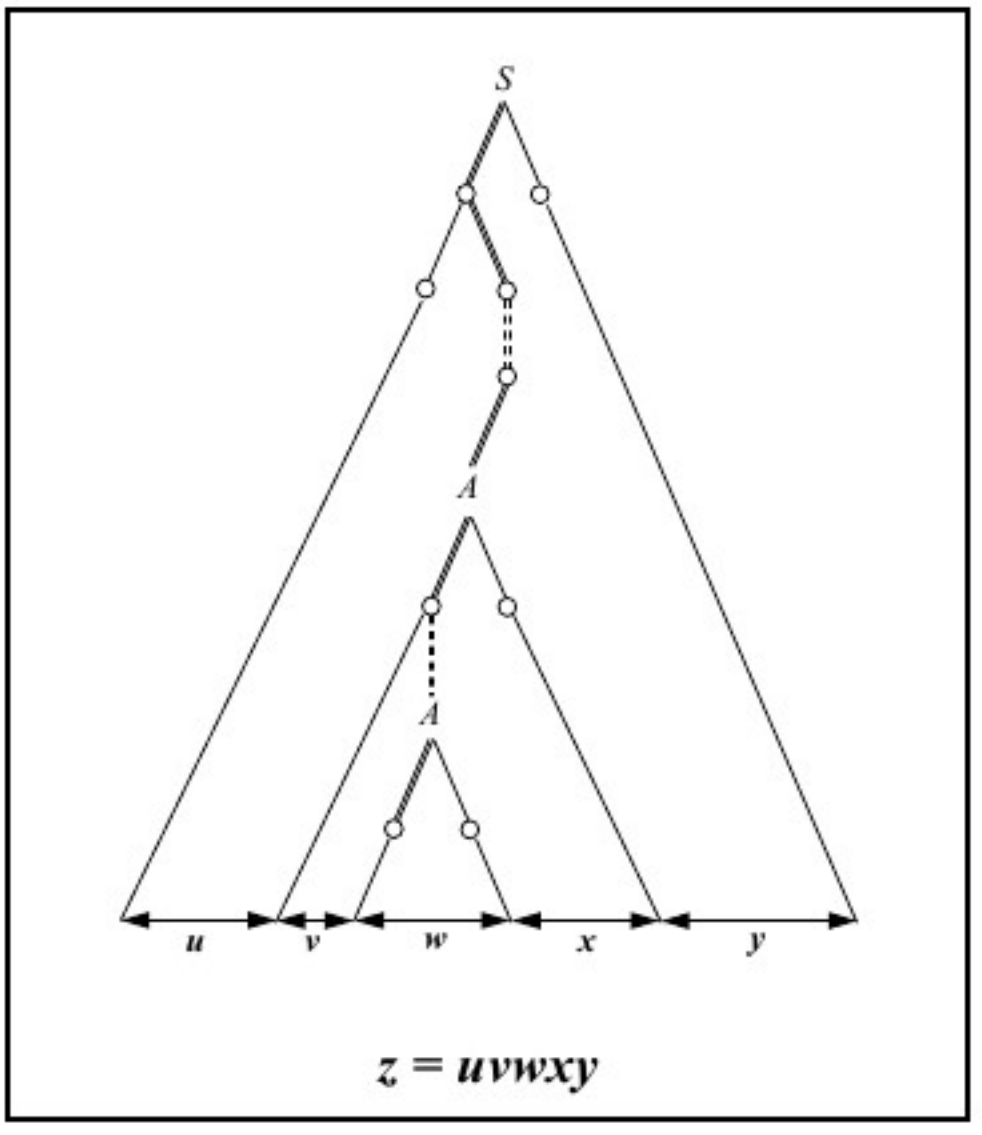


Figura 4.7

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Il sottoalbero più in basso con radice nel nodo etichettato con una A genera la sottostringa w, mentre quello più in alto genera la sottostringa vwx.

Poiché la G è C.F., sostituendo il sottoalbero più in alto con quello più in basso, si ottiene ancora una derivazione valida. Il *nuovo albero* genera la *stringa* uwy.

Se effettuiamo la sostituzione inversa (il sottoalbero più in basso viene rimpiazzato da quello più in alto), otteniamo un albero di derivazione lecito per la stringa $uvvwxxy$, ossia uv^2wx^2y. Ripetendo questa sostituzione un numero finito di volte, si ottiene l'insieme di stringhe:

$$\{uv^n wx^n y \mid n \geq 0\}$$

Proposizione 4.1

Ogni linguaggio C.F. infinito deve contenere almeno un sottoinsieme infinito di stringhe della forma:

$$uv^n wx^n y \qquad n \geq 0$$

Formalizziamo ora alcuni risultati connessi con il processo di sostituzione di sottoalberi.

Lemma 4.1

Sia $G = (X, V, S, P)$ una grammatica C.F. e supponiamo che:

$$m = \max\left\{ |v| \;\middle|\; A \to v \in P \right\}$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Sia T_w un albero di derivazione per una stringa w di $L(G)$. Se l'altezza di T_w è al più uguale ad un intero j, allora: $|w| \leq m^j$.

In formule: $height(T_w) \leq j \Rightarrow |w| \leq m^j$.

Dimostrazione 4.1

La dimostrazione vale per un albero di derivazione che abbia come radice un qualunque simbolo NT, non necessariamente S.

Procediamo per induzione su j.

Passo base

$j = 1$

T_w rappresenta un'unica produzione (Figura 4.8):

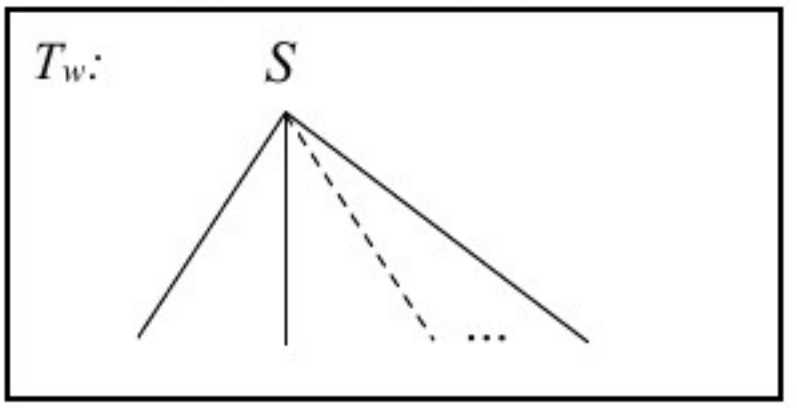


Figura 4.8

dunque la parola generata è composta al più da m caratteri terminali e si ha:

$$|w| \leq m$$

Passo induttivo

Supponiamo che il lemma valga per ogni albero di altezza al più uguale a j e la cui radice sia un simbolo NT e dimostriamolo per un albero di altezza j+1.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Supponiamo che il livello più alto dell'albero rappresenti la produzione $A \to v$ (Figura 4.9).

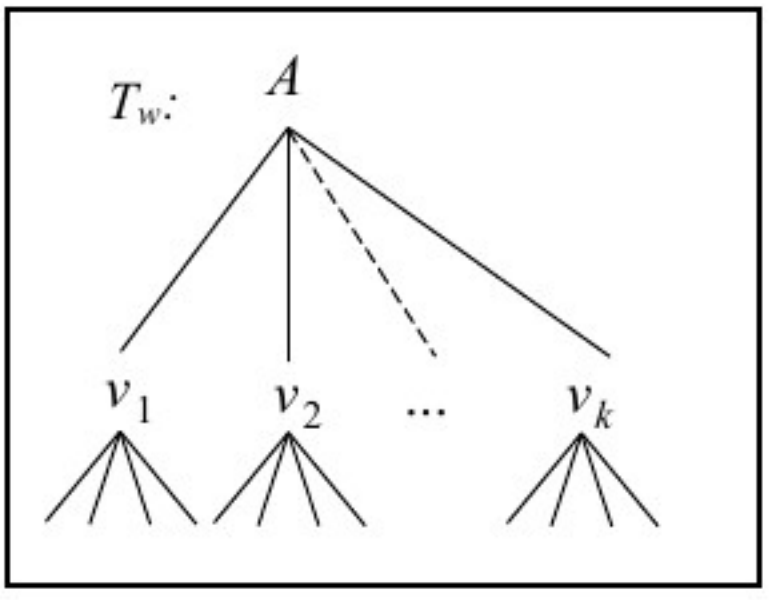


Figura 4.9

dove $v = v_1 v_2 ... v_k$, $|v| = k$, $k \le m$ (il sottoalbero di profondità 1 ha al più m figli).

Ogni simbolo v_i, $i = 1, 2,..., k$ di v può essere radice di un sottoalbero di altezza al più uguale a j, poiché T_w ha altezza uguale a j+1 (un v_i potrebbe anche essere un terminale).

Dunque, per ipotesi di induzione, ciascuno di questi alberi ha al più m^j foglie. Poiché $|v| = k \le m$, la stringa w, frontiera dell'albero T_w, ha lunghezza:

$$|w| \le \underbrace{m^j + m^j + ... + m^j}_{|v|=k \text{ volte}} = |v| \cdot m^j = k \cdot m^j \le m \cdot m^j = m^{j+1} \qquad c.v.d.$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

4.3 Pumping Lemma per i linguaggi liberi da contesto

Teorema 4.2 (Pumping Lemma per i linguaggi liberi o Teorema *uvwxy*)

Sia L un linguaggio libero da contesto. Allora esiste una costante p, che dipende solo da L, tale che se z è una parola di L di lunghezza maggiore di p ($|z| > p$), allora z può essere scritta come $uvwxy$ in modo tale che:

(1) $|vwx| \leq p$;

(2) al più uno tra v e x è la parola vuota ($vx \neq \lambda$);

(3) $\forall\, i,\ i \geq 0 : uv^i wx^i y \in L$.

Osservazione 4.2

(a) Dato un linguaggio generato da una grammatica che non è context-free, non possiamo escludere immediatamente che non esiste una grammatica C.F. che generi lo stesso linguaggio.

(b) Se un linguaggio infinito non obbedisce al Pumping Lemma, non può essere generato da una grammatica C.F.

Il Pumping Lemma per i linguaggi liberi fornisce una *condizione necessaria* affinché un linguaggio sia libero.

$$L \text{ libero} \Rightarrow \exists p, \ldots.$$

Dunque può essere utilizzato per dimostrare che un linguaggio non è libero (con una dimostrazione per assurdo - modus tollens).

$$\nexists p, \ldots \Rightarrow L \text{ non è libero}$$

Dimostrazione 4.2

Sia $G = (X, V, S, P)$ una grammatica C.F. che genera L.

Denotiamo con m il numero di simboli della più lunga parte destra di una produzione di G:

$$m = \max\left\{ |v| \;\middle|\; A \to v \in P \right\}$$

Denotiamo con k la cardinalità dell'alfabeto nonterminale di G:

$$k = |V|$$

Poniamo $p = m^{k+1}$ e sia $z \in L$, con $|z| > p$.

Per il lemma precedente (Lemma 4.1):

$$\left[\left(height(T_z) \le j \Rightarrow |z| \le m^j\right) \Leftrightarrow \left(|z| > m^j \Rightarrow height(T_z) > j\right)\right],$$

se $|z| > p = m^{k+1}$ allora ogni albero di derivazione per z deve avere altezza maggiore di k+1.

Dunque, in ogni albero di derivazione per z deve esistere almeno un cammino di lunghezza non inferiore a k+2.

Poiché $k = |V|$, almeno due NT devono comparire duplicati su quel cammino o un NT deve essere ripetuto 3 o più volte sul cammino.

Senza ledere la generalità della dimostrazione, denotiamo con A il NT che compare duplicato per ultimo su quel cammino (si faccia riferimento alla Figura 4.7).

Non vi sono pertanto altri NT ripetuti almeno due volte al di sotto della A più in alto (della coppia di A).

Questa considerazione implica che il cammino dalla A più in alto della coppia ad una foglia ha lunghezza al più k+1.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Indichiamo con vwx la sottostringa derivata dal sottoalbero avente radice nella A più alta della coppia, ove w è la sottostringa derivata dal sottoalbero avente radice nella A più bassa della coppia.

Dal Lemma 4.1, si ha:

$$|vwx| \leq m^{k+1} = p$$

per cui risulta dimostrata la (1) del Pumping Lemma.

Scriviamo z nella forma $uvwxy$, ed applichiamo il principio di sostituzione di sottoalberi, visto in precedenza.

Se sostituiamo al sottoalbero avente radice nella A più alta della coppia quello avente radice nella A più bassa, otteniamo un albero di derivazione per la stringa:

$$uwy = uv^0wx^0y$$

che è la (3) per $i = 0$.

Se operiamo la sostituzione inversa, otteniamo un albero di derivazione per la stringa:

$$uvvwxxy = uv^2wx^2y$$

che è la (3) per $i = 2$.

Se ripetiamo la suddetta sostituzione i-1 volte, la stringa derivata è:

$$u\underbrace{vv\ldots v}_{i \text{ volte}}w\underbrace{xx\ldots x}_{i \text{ volte}}y = uv^iwx^iy$$

per cui la (3) risulta dimostrata.

La (2) può essere dimostrata per assurdo.

Sia:

$$v = \lambda = x$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

La sostituzione del sottoalbero avente radice nella A più alta della coppia con quello avente radice nella A più bassa non provoca alcun cambiamento nella stringa z derivata dall'intero albero.
Ma tale sostituzione provoca la diminuzione della lunghezza del cammino che dalla A (in origine quella più in alto) porta ad una foglia.
Dunque, anche il cammino di lunghezza non inferiore a k+2 (su cui compariva la coppia di A) nell'albero di derivazione per z risulta accorciato.
In questo modo abbiamo ottenuto un albero di derivazione per z con altezza almeno uguale a k+1. Ma questo è assurdo per il Lemma 4.1. *c.v.d.*

4.4 Ambiguità di grammatiche e linguaggi

Definizione 4.7 (Grammatica ambigua)
Una grammatica G libera da contesto è *ambigua* se esiste una stringa x in $L(G)$ che ha due alberi di derivazione differenti. ■

In modo alternativo, G è ambigua se esiste una stringa x in $L(G)$ che ha due derivazioni sinistre (o destre) distinte.

Esempio 4.3
La seguente grammatica libera da contesto:

$$G_2 = (X, V, S, P)$$

$$X = \{a, +\}, \qquad V = \{S\}, \qquad P = \{S \to S + S,\ S \to a\}$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

è ambigua. Infatti la stringa $w = a + a + a$ in $L(G_2)$ ha due differenti alberi di derivazione (Figura 4.10).

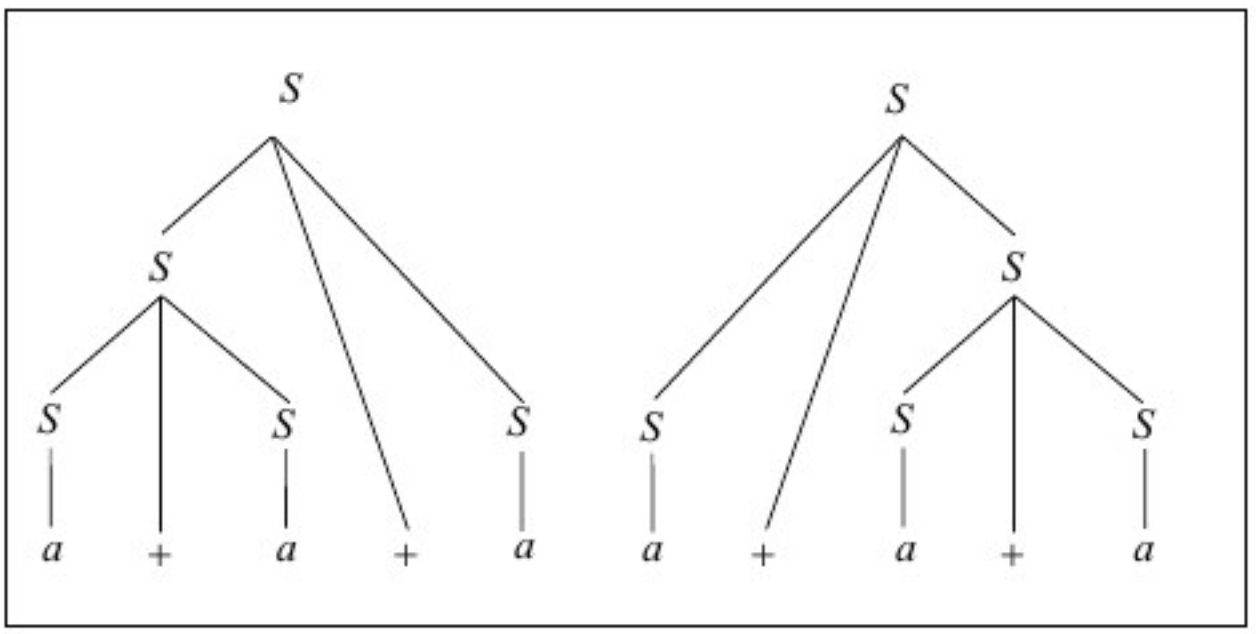


Figura 4.10

Nel linguaggio naturale, l'ambiguità (sintattica) è un fenomeno intrinseco, che si verifica frequentemente.

<u>Esempio 4.4</u>

"un dolce rosso"

"dolce" — sostantivo / aggettivo

"rosso" — aggettivo / sostantivo

"un dolce rosso" — "una torta di colore rosso" / "un uomo rosso dal carattere dolce"

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Agli effetti delle applicazioni ai linguaggi di programmazione, *l'ambiguità* è una *proprietà negativa*. Infatti, il *significato di una frase* può essere definito come una *funzione del suo albero di derivazione*.

Sicché, se esiste più di un albero di derivazione per una stessa frase, essa può avere un significato non univoco.

Si preferisce perciò cercare una grammatica differente che, pur generando lo stesso linguaggio, non sia ambigua.

L'unico vantaggio delle grammatiche ambigue risulta essere, in generale, il minore numero di regole che possono avere rispetto ad una grammatica non ambigua.

Esempio 4.5

Il linguaggio $L(G_2)$ (dell'Esempio 4.3) può essere generato anche dalla seguente grammatica:

$$G_2 = (X, V, S, P)$$

$$X = \{a, +\}, \qquad V = \{S\}, \qquad P = \{S \to S + a,\ S \to a\}$$

G_3 non è ambigua.

Inoltre: $L(G_2) = L(G_3) = \left\{ aw \mid w \in \{+a\}^* \right\} = a \cdot \{+a\}^*$.

Esistono però dei linguaggi, detti *inerentemente ambigui*, per i quali tutte le grammatiche che li generano risultano ambigue.

Definizione 4.8 (Linguaggio inerentemente ambiguo)

Un linguaggio L è *inerentemente ambiguo* se ogni grammatica che lo genera è ambigua.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

$$(\forall G)\ \ L = L(G) : G \text{ ambigua}$$ ■

Esempio 4.6

Un linguaggio inerentemente ambiguo è:

$$L = \{a^i b^j c^k \mid a^i b^j c^k,\ (i = j \vee j = k)\}$$

Intuitivamente, ci si rende conto di ciò pensando che in ogni grammatica che genera L devono esistere due diversi insiemi di regole per produrre le stringhe $a^i b^i c^k$ e le stringhe $a^i b^j c^j$. Le stringhe $a^i b^i c^i$ saranno necessariamente prodotte in due modi distinti, con distinti alberi di derivazione e conseguente ambiguità. Per esercizio: determinare una grammatica che genera L e provare questo risultato.

Esempio 4.7 (Linguaggi di programmazione)

Si consideri la seguente grammatica $G = (X, V, S, P)$:

$$X = \{\underline{\text{if}}, \underline{\text{then}}, \underline{\text{else}}, \text{a}, \text{b}, \text{p}, \text{q}\}$$

$$V = \{S, C\}$$

$$P = \{S \to \underline{\text{if}}\ C\ \underline{\text{then}}\ S\ \underline{\text{else}}\ S \mid \underline{\text{if}}\ C\ \underline{\text{then}}\ S \mid \text{a} \mid \text{b},\ C \to \text{p} \mid \text{q}\}$$

G è ambigua. Infatti la stringa:

$$w = \underline{\text{if}}\ \text{p}\ \underline{\text{then}}\ \underline{\text{if}}\ \text{q}\ \underline{\text{then}}\ \text{a}\ \underline{\text{else}}\ \text{b}$$

può essere generata in due modi, descritti in Figura 4.11 e 4.12:

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

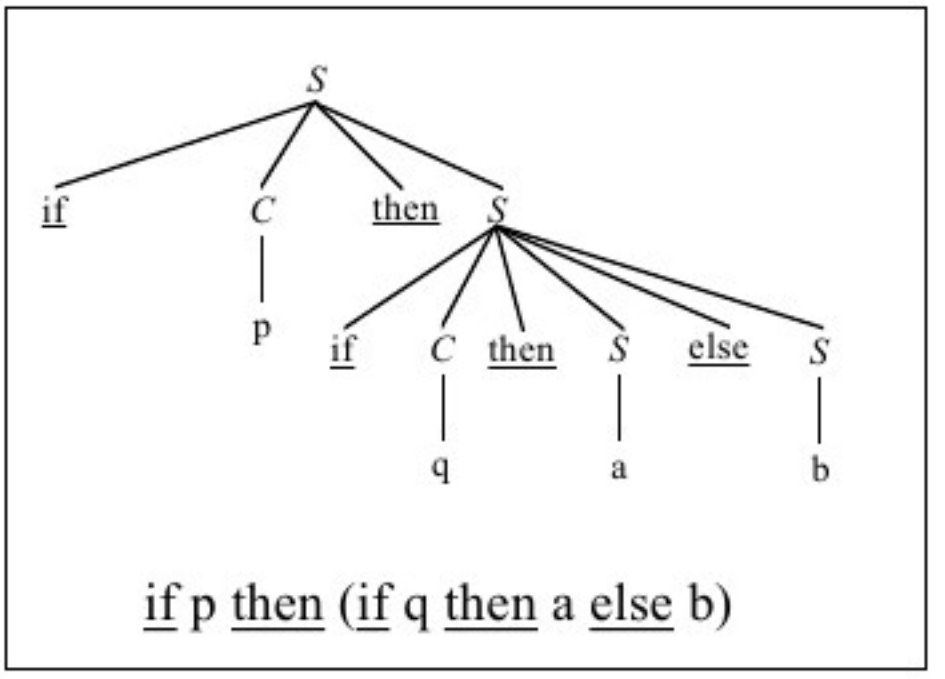


Figura 4.11

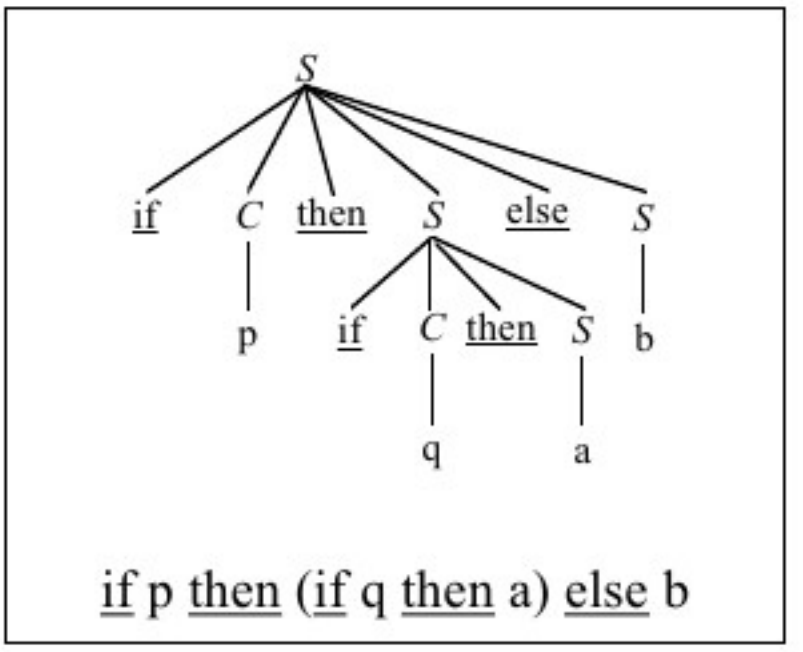


Figura 4.12

Per rendere non ambigua G si usa solitamente la convenzione di associare ogni istruzione else con l'istruzione if più vicina.

La grammatica che riflette questa convenzione è la seguente:

$$G' = (X, V, S, P)$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

$$X = \{\underline{\text{if}}, \underline{\text{then}}, \underline{\text{else}}, \text{a}, \text{b}, \text{p}, \text{q}\}$$

$$V = \{S, S_1, S_2, C, T\}$$

$$\underline{\text{if}}\ C\ \underline{\text{then}}\ S\ \underline{\text{else}}\ S \mid \underline{\text{if}}\ C\ \underline{\text{then}}\ S \mid \text{a} \mid \text{b},\ C \to \text{p} \mid \text{q}\}$$

$$\begin{aligned} P = \{\ & S \to S_1 \mid S_2, \\ & S_1 \to \underline{\text{if}}\ C\ \underline{\text{then}}\ S_1\ \underline{\text{else}}\ S_2 \mid T, \\ & S_2 \to \underline{\text{if}}\ C\ \underline{\text{then}}\ S \mid \underline{\text{if}}\ C\ \underline{\text{then}}\ S_1\ \underline{\text{else}}\ S_2 \mid T, \\ & C \to p \mid q, \\ & T \to a \mid b\ \} \end{aligned}$$

In G' la stringa w = <u>if</u> p <u>then</u> <u>if</u> q <u>then</u> a <u>else</u> b ha un unico albero di derivazione (Figura 4.13):

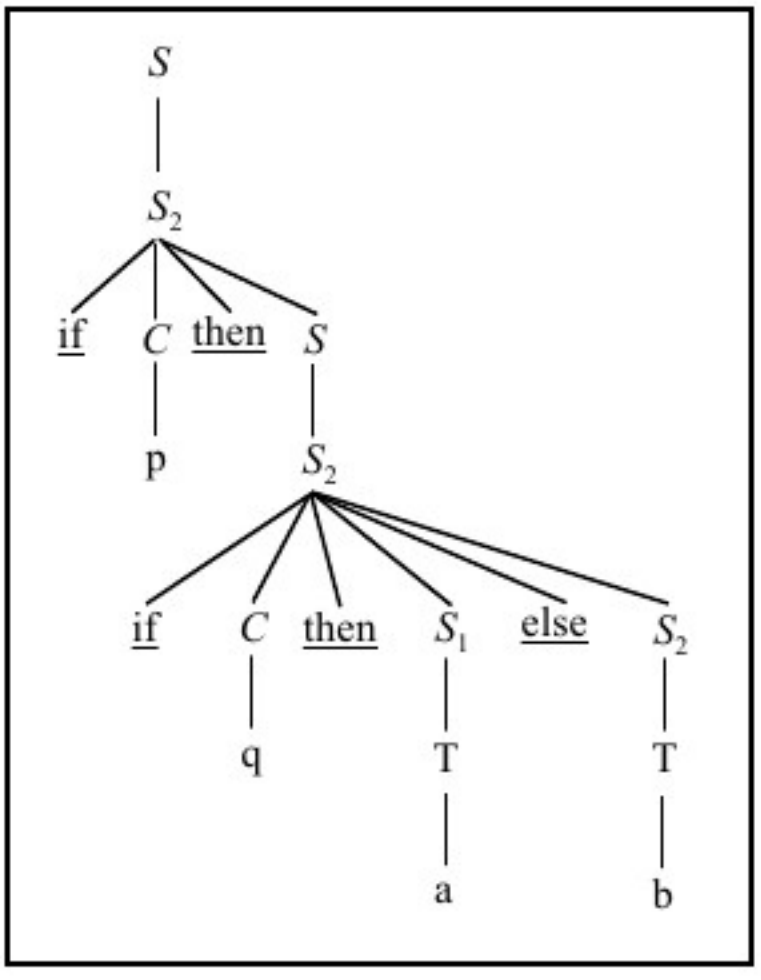


Figura 4.13

Ma è proprio vero che l'albero di derivazione per w è unico?

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Si tenga conto che l'ambiguità di un linguaggio prescinde dal nome delle variabili ma dipende dalla struttura.

4.5 Esercizi

Esercizio 4.1

1) Determinare la grammatica che genera il seguente linguaggio:

$$L = \{a^n b^n c^n \mid n > 0\}$$

e dimostrare la correttezza di tale grammatica.

2) Di che tipo è la grammatica che genera L?

3) Applicare il Pumping Lemma per dimostrare che L non è libero.

1)

$$G = (X, V, S, P)$$

$$X = \{a,b,c\} \qquad V = \{S,A,B,C\} \qquad P = \left\{ \begin{array}{l} S \overset{(1)}{\rightarrow} aBC,\ \ S \overset{(2)}{\rightarrow} aSBC \\ CB \overset{(3)}{\rightarrow} BC, \\ aB \overset{(4)}{\rightarrow} ab,\ \ bB \overset{(5)}{\rightarrow} bb, \\ bC \overset{(6)}{\rightarrow} bc,\ \ cC \overset{(7)}{\rightarrow} cc \end{array} \right\}$$

Vediamo come ricavare le produzioni di G.

Una derivazione della parola di lunghezza minima in L, $w = abc$, è la seguente:

$$S \underset{(1)}{\Rightarrow} aBC \underset{(4)}{\Rightarrow} abC \underset{(6)}{\Rightarrow} abc$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Le derivazioni (sinistre) delle altre parole in L sono del tipo:

$$S \underset{(2)}{\Rightarrow} aSBC \underset{(2)}{\Rightarrow} aaSBCBC$$

La struttura della stringa $aaSBCBC$ non è quella desiderata (che contraddistingue le parole di L).

È necessaria una produzione che effettui uno scambio di nonterminali:

$$CB \to BC$$ [6]

Grazie a questa regola di produzione si ha:

$$S \underset{(2)}{\Rightarrow} aSBC \underset{(1)}{\Rightarrow} aaBCBC \underset{(3)}{\Rightarrow} aaBBCC \underset{(4)}{\Rightarrow} aabBCC \underset{(5)}{\Rightarrow} aabbCC \underset{(6)}{\Rightarrow}$$
$$\underset{(6)}{\Rightarrow} aabbcC \underset{(7)}{\Rightarrow} aabbcc$$

Le derivazioni di parole più lunghe in L non hanno bisogno di ulteriori produzioni:

$$S \underset{(2)}{\Rightarrow} aSBC \overset{2}{\underset{(2)}{\Rightarrow}} aaaSBCBCBC \underset{(1)}{\Rightarrow} aaaaBCBCBCBC \underset{(3)}{\Rightarrow}$$
$$\underset{(3)}{\Rightarrow} aaaaBBCCBCBC \underset{(3)}{\Rightarrow} aaaaBBCCBCBC \underset{(3)}{\Rightarrow}$$
$$\underset{(3)}{\Rightarrow} aaaaBBCBCCBC \underset{(3)}{\Rightarrow} a^4B^3CCCBC \underset{(3)}{\Rightarrow}$$
$$\underset{(3)}{\Rightarrow} a^4B^3CCBCC \underset{(3)}{\Rightarrow} a^4B^4C^4 \underset{(4)}{\Rightarrow} a^4bB^3C^4 \underset{(5)}{\Rightarrow} a^4b^2B^2C^4 \overset{2}{\underset{(5)}{\Rightarrow}}$$
$$\overset{2}{\underset{(5)}{\Rightarrow}} a^4b^4C^4 \underset{(6)}{\Rightarrow} a^4b^4cC^3 \underset{(7)}{\Rightarrow} a^4b^4c^2C^2 \overset{2}{\underset{(7)}{\Rightarrow}} a^4b^4c^4$$

2) G è una grammatica monotona. Per il teorema di equivalenza delle grammatiche monotone e contestuali (Teorema 3.2), esiste una grammatica contestuale G' equivalente a G.

---

[6] È una produzione monotona.

3) Supponiamo, per assurdo, che L sia libero da contesto. Per il Pumping Lemma sui linguaggi liberi, esiste una costante p tale che:

$$\forall z \;\; z \in L, \; |z| > p \quad \Rightarrow \quad z = uvwxy$$

ed inoltre valgono le seguenti:

(1) $|vwx| \leq p$;

(2) $vx \neq \lambda$;

(3) $\forall i, \; i \geq 0 : uv^i wx^i y \in L$

Consideriamo una parola in L:

$$z = a^p b^p c^p .$$

Il Pumping Lemma può essere applicato a tale parola poiché $|z| = 3p > p$ e dunque z può essere scritta nella forma $z = uvwxy$, in modo tale che:

$$|vwx| \leq p$$

Poiché la stringa vwx ha lunghezza al più uguale a p, si hanno le seguenti possibilità:

(i) vwx è formata da sole a, cioè è del tipo $vwx = a^k$, $0 < k \leq p$;

(ii) vwx è formata da sole b, cioè è del tipo $vwx = b^k$, $0 < k \leq p$;

(iii) vwx è formata da sole c, cioè è del tipo $vwx = c^k$, $0 < k \leq p$;

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

(iv) vwx è a cavallo tra a e b, cioè è del tipo $vwx = a^k b^r$, $0 < k + r \leq p$ e $k, r > 0$;

(v) vwx è a cavallo tra b e c, cioè è del tipo $vwx = b^k c^r$, $0 < k + r \leq p$ e $k, r > 0$.

È immediato osservare che vwx non può essere formata da a, b e c (ossia non può essere contemporaneamente a cavallo tra a e b e tra b e c), perché non è sufficientemente lunga.

Consideriamo la stringa (*pompata*):

$$uv^2wx^2y$$

per ognuno dei casi (i) - (v).

Per la (3) del Pumping Lemma sui linguaggi liberi, si dovrebbe avere:

$$uv^2wx^2y \in L$$

Ma nel caso:

(i) aggiungiamo almeno una a, ed al più p a; $uv^2wx^2y = a^{p+t}b^pc^p$, $0 < t \leq p$;

(ii) aggiungiamo almeno una b, ed al più p b; $uv^2wx^2y = a^pb^{p+t}c^p$, $0 < t \leq p$;

(iii) aggiungiamo almeno una c, ed al più p c; $uv^2wx^2y = a^pb^pc^{p+t}$, $0 < t \leq p$;

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

(iv) per la (2) del Pumping Lemma, si hanno le seguenti possibilità:

(iv.a) $v \neq \lambda,\ x \neq \lambda$;

(iv.b) $v \neq \lambda,\ x = \lambda$;

(iv.c) $v = \lambda,\ x \neq \lambda$.

Osserviamo preliminarmente che, se $v \neq \lambda$, allora v è costituita da sole a. Infatti, se v fosse del tipo $v = a^k b^{r'}$, con $0 < r' \leq r$, si avrebbe $uv^2wx^2y = a^{p-k}a^k b^{r'} a^k b^{r'} b^s c^p \notin L$, con $p - r' \leq s \leq 2(r - r') + p - r$.

Analogamente, se $x \neq \lambda$, allora x è costituita da sole b. Infatti, se x fosse del tipo $x = a^{k'} b^r$, con $0 < k' \leq k$, si avrebbe $uv^2wx^2y = a^s a^{k'} b^r a^{k'} b^r b^{p-r} c^p \notin L$, con $p - k' \leq s \leq 2(k - k') + p - k$.

Per cui:

(iv.a) se $v \neq \lambda,\ x \neq \lambda$, per l'osservazione precedente $v = a^{k'}$, con $0 < k' \leq k$ e $x = b^{r'}$, $0 < r' \leq r$ e si ha che $uv^2wx^2y = a^{p+k'} b^{p+r'} c^p \notin L$, poiché $k',\ r' > 0$;

(iv.b) se $v \neq \lambda,\ x = \lambda$, si ha $v = a^{k'}$, con $0 < k' \leq k$ e $uv^2wx^2y = a^{p+k'} b^p c^p \notin L$, poiché $k' > 0$;

(iv.c) se $v = \lambda,\ x \neq \lambda$, si ha $x = b^{r'}$, con $0 < r' \leq r$ e $uv^2wx^2y = a^p b^{p+r'} c^p \notin L$, poiché $r' > 0$;

(v) Si lascia per esercizio.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

In ciascuno dei casi (i) - (v) $uv^2wx^2y \notin L$.

Assurdo. Ne segue che L non è un linguaggio libero da contesto, poiché è un linguaggio infinito per il quale non vale il Pumping Lemma.

Esercizio 4.2

Applicare il Pumping Lemma per dimostrare che il seguente linguaggio L non è libero da contesto:

$$L = \left\{ a^{n^2} \mid n \geq 0 \right\}$$

Analizziamo le parole che costituiscono L.

$$L = \left\{ \lambda, a, a^4, a^9, a^{16}, \ldots \right\}$$

Supponiamo, per assurdo, che L sia libero da contesto.

Per il Pumping Lemma sui linguaggi liberi, esiste un numero naturale p, dipendente solo da L, tale che se $z \in L$ e $|z| > p$, allora:

$$z = uvwxy$$

(1) $|vwx| \leq p$;

(2) $vx \neq \lambda$;

(3) $uv^iwx^iy \in L, \forall\ i \geq 0$.

Consideriamo la parola:

$$z = a^{p^2}$$

$z \in L$ ed inoltre $|z| = p^2 > p$.

Per il Pumping Lemma, possiamo scrivere:

$$z = uvwxy \text{ ove } |vwx| \leq p \ .$$

Consideriamo la stringa:

$$uv^2wx^2y$$

Per la (3) del Pumping Lemma, si deve avere:

$$uv^2wx^2y \in L$$

Ma:

$$\begin{aligned} |uv^2wx^2y| &= |uvwxy| + |vx| \leq |uvwxy| + |vwx| \leq \\ &\leq p^2 + p < p^2 + 2p + 1 = (p+1)^2 \end{aligned}$$

Dunque:

$$uv^2wx^2y \notin L$$

Assurdo. Ne segue che L non è un linguaggio libero da contesto.

<u>Esercizio 4.3</u>

Applicare il Pumping Lemma per dimostrare che il seguente linguaggio L non è libero da contesto:

$$L = \left\{ a^i b^j \mid i = 2^j, \ i, j \geq 0 \right\}$$

Analizziamo le parole che costituiscono L.

$$L = \left\{ a^{2^j} b^j \mid j \geq 0 \right\} = \left\{ a, a^2b, a^4b^2, a^8b^3, a^{16}b^4, \ldots \right\}$$

Per assurdo, supponiamo L libero da contesto. Vale, dunque, per L il Pumping Lemma sui linguaggi liberi.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Dunque, si ha:

$\exists p \in \aleph$, p dipendente solo da L, tale che se $z \in L$, $|z| > p$, allora:

$$z = uvwxy$$

(4) $|vwx| \leq p$;

(5) $vx \neq \lambda$;

(6) $uv^i wx^i y \in L$, $\forall\ i \geq 0$.

Consideriamo la parola:

$$z = a^{2^p} b^p$$

$z \in L$ ed inoltre $|z| = 2^p + p > p$.

Per il Pumping Lemma, possiamo scrivere:

$$z = uvwxy$$

ove $|vwx| \leq p$. Consideriamo la stringa:

$$uv^2wx^2y$$

Per la (3) del Pumping Lemma, si deve avere:

$$uv^2wx^2y \in L$$

Ma:

$$\begin{aligned} |uv^2wx^2y| &= |uvwxy| + |vx| \leq |uvwxy| + |vwx| \leq \\ &\leq 2^p + p + p \leq 2^p + 2^p + p = 2 \cdot 2^p + p = \\ &= 2^{p+1} + p < 2^{p+1} + p + 1 \end{aligned}$$

Dunque la stringa pompata uv^2wx^2y non è del tipo $a^{2^j} b^j$, ossia:

$$uv^2wx^2y \notin L$$

Assurdo. Ne segue che L non è un linguaggio libero da contesto.

Esercizio 4.4

Dimostrare che il seguente linguaggio

$$L = \left\{ a^k b^r \mid k > 0,\ r > k^2 \right\}$$

non è libero.

Analizziamo le parole che costituiscono L:

$$\begin{aligned} L = \{ & ab^2, ab^3, ab^4, \ldots \\ & a^2b^5, a^2b^6, a^2b^7, \ldots \\ & a^3b^{10}, a^3b^{11}, a^3b^{12}, \ldots \} \end{aligned}$$

Supponiamo, per assurdo, che L sia libero da contesto.

Per il Pumping Lemma sui linguaggi liberi, esiste un numero naturale p, dipendente da L, tale che, se $z \in L$, $|z| > p$, allora:

$$z = uvwxy$$

(7) $|vwx| \leq p$;

(8) $vx \neq \lambda$;

(9) $uv^iwx^iy \in L$, $\forall\ i \geq 0$.

Consideriamo la parola:

$$z = a^p b^{p^2+1}$$

$z \in L$ ed inoltre $|z| = p + p^2 + 1 > p$.

Per il Pumping Lemma, possiamo scrivere:

$$z = uvwxy.$$

Per la sottostringa vwx si hanno le seguenti possibilità:

(i) vwx è formata da sole a;

(ii) vwx è formata da sole b;

(iii) vwx è a cavallo tra a e b.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Nel caso (i), per la (1) e la (2) del Pumping Lemma, si ha:

$$0 < |vwx| \leq p \qquad \text{e} \qquad 0 < |vx| \leq p$$

Dunque vwx è formata da almeno una a ed al più p a. Consideriamo la parola:

$$uv^2wx^2y.$$

Tale parola ha almeno una ed al più p a in più di z. Dunque, uv^2wx^2y ha almeno p+1 a (ed al più 2p a), mentre il numero delle b non è mutato. Ossia, denotato con #(x) il numero di occorrenze del simbolo x nella parola uv^2wx^2y:

$$p+1 \leq \#(a) \leq 2p \qquad \text{e} \qquad \#(b) = p^2+1$$

Ma questo vuol dire che:

$$\#(b) = p^2+1 < (p+1)^2 \leq \#(a)^2 \leq 4p^2$$

Dunque $r < k^2$ e $uv^2wx^2y \notin L$.

Nel caso (ii), vwx è formata da almeno una b ed al più p b. Consideriamo la parola:

$$uv^0wx^0y.$$

Tale parola ha almeno una ed al più p b in meno di z, in quanto per la (1) e la (2) del Pumping Lemma:

$$0 < |vwx| \leq p.$$

Dunque, uv^0wx^0y ha al più p^2 b (ed almeno p^2+1-p b), mentre il numero delle a non è cambiato. Quindi si ha che:

$$p^2-p+1 \leq \#(b) \leq p^2 \qquad \text{e} \qquad \#(a) = p$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

da cui:

$$\#(b) \leq \#(a)^2 = p^2 .$$

Dunque si ha $r \leq k^2$ e $uv^0wx^0y \notin L$.

Nel caso (iii), *vwx* è formata sia da *a* sia da *b* e

$$0 < \#(a) + \#(b) \leq p$$

poiché per le (1) e (2) del Pumping Lemma:

$$0 < |vwx| \leq p$$

Se $v \neq \lambda$, allora v contiene solo a (altrimenti v^i, $i > 1$, conterrebbe delle *a* alternate a delle *b*). Analogamente, se $x \neq \lambda$ allora *x* contiene solo *b*.

Dunque, ci sono tre possibilità:

(iii.a) $v \neq \lambda,\ x = \lambda;$

(iii.b) $v = \lambda,\ x \neq \lambda;$

(iii.c) $v \neq \lambda,\ x \neq \lambda.$

Nel caso (iii.a), consideriamo la parola uv^2wx^2y. Come nel caso (i), tale parola ha almeno una *a* in più, rispetto a *z* (ed al più $p-1$ *a* in più di *z*, poiché ci deve essere almeno una *b* in *vwx* e questa deve essere necessariamente in *w*).

Il numero delle *b* in uv^2wx^2y non è mutato. Si ha che:

$$p+1 \leq \#(a) \leq p+p-1 = 2p-1 \qquad \text{e} \qquad \#(b) = p^2+1$$

e quindi poiché:

$$\#(b) = p^2+1 < (p+1)^2 \leq \#(a)^2$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

si ha che:

$$uv^2wx^2y \notin L\,.$$

Nel caso (iii.b), consideriamo la parola uv^0wx^0y. Come nel caso (ii), tale parola ha almeno una b in meno rispetto a z (ed al più $p-1$ b in meno di z, poiché ci deve essere almeno una a in vwx e questa deve essere necessariamente in w), mentre il numero delle a in uv^0wx^0y non cambia. Si ha che:

$$p^2+1-(p-1) \le \#(b) \le p^2 \qquad \text{e} \qquad \#(a) = p$$

e quindi poiché:

$$\#(b) \le \#(a)^2 = p^2$$

si ha che:

$$uv^0wx^0y \notin L\,.$$

Nel caso (iii.c), consideriamo la parola uv^2wx^2y.

Poiché $v \neq \lambda$, v contiene almeno una a. Dunque v^2 contiene almeno due a e uv^2wx^2y contiene almeno $p+1$ a.

Ma:

$$\left|uv^2wx^2y\right| = \left|uvwxy\right| + \left|vx\right| = \left|z\right| + \left|vx\right| \le \left|z\right| + \left|vwx\right| \le p + p^2 + 1 + p =$$
$$= p^2 + 2p + 1 = (p+1)^2 < p + 1 + (p+1)^2 + 1$$

Dunque uv^2wx^2y ha almeno $p+1$ a, ma la sua lunghezza è strettamente minore della lunghezza della parola di L con almeno $p+1$ a e di lunghezza minima. Ne consegue che:

$$uv^2wx^2y \notin L\,.$$

In ciascuno dei casi (e sottocasi di) (i), (ii) e (iii) risulta violata la (3) del Pumping Lemma per i linguaggi liberi.
Assurdo.
L'assurdo deriva dall'aver assunto L libero da contesto. Dunque L non è un linguaggio libero.

Esercizio 4.5

Sia dato il linguaggio:

$$L = \{a^t \mid t \text{ numero primo}\}$$

Stabilire se L è libero e giustificare formalmente la risposta.

Analizziamo le parole che costituiscono L.

$$L = \{a, a^2, a^3, a^5, a^7, a^{11}, a^{13}, a^{17}, \ldots\}$$

$\lvert a\rvert = 1$	$\lvert a^{11}\rvert = 11$	$\lvert a^{29}\rvert = 29$
$\lvert a^2\rvert = 2$	$\lvert a^{13}\rvert = 13$	$\lvert a^{31}\rvert = 31$
$\lvert a^3\rvert = 3$	$\lvert a^{17}\rvert = 17$	…
$\lvert a^5\rvert = 5$	$\lvert a^{19}\rvert = 19$	$\lvert a^n\rvert = n$
$\lvert a^7\rvert = 7$	$\lvert a^{23}\rvert = 23$	…

La lunghezza delle parole di L non cresce in maniera costante, né è possibile determinare un sottoinsieme infinito di L le cui parole hanno lunghezze che crescono in maniera costante. Dunque L non è un linguaggio libero da contesto.
Proviamo formalmente ciò.
Supponiamo per assurdo che L sia libero da contesto.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Per il Pumping Lemma per i linguaggi liberi deve esistere una costante intera p, dipendente unicamente da L, tale che:

$$\forall z \;\; z \in L, \; |z| > p \;\; \Rightarrow \;\; z = uvwxy$$

ed inoltre valgono le seguenti:

(1) $|vwx| \leq p$;

(2) $vx \neq \lambda$;

(3) $\forall i, \; i \geq 0 : uv^i wx^i y \in L$

Consideriamo la seguente parola di L:

$$z = a^{f(p+1)}$$

ove f è una funzione definita in $\aleph$ ed a valori in $\aleph$, che associa ad ogni intero t il t-esimo numero primo.

$$f \; : \aleph \rightarrow \aleph$$

$\forall t \in \aleph : f(t) = t-$esimo numero primo .

Pertanto z è la parola di L ottenuta concatenando un numero di a pari al p+1-esimo numero primo.

È immediato osservare che valgono le seguenti relazioni:

$$\begin{aligned} \forall t \in \aleph : \quad & f(t) \geq t \qquad\qquad (f(t) > t \quad \forall t, \; t > 3) \\ & f(t+1) > t \\ & f(t+1) > f(t) \end{aligned}$$

Pertanto risulta:

$$|z| = f(p+1) > p$$

e z può essere scritta nella forma $z = uvwxy$ in modo tale che valgano le (1), (2) e (3).

Consideriamo la parola:

$$uv^{f(p+1)+1}wx^{f(p+1)+1}y$$

Per la (3), con $i = f(p+1)+1$, si deve verificare che tale parola è un elemento di L.

D'altra parte, se valutiamo la lunghezza della parola $uv^{f(p+1)+1}wx^{f(p+1)+1}y$ si ha:

$$\begin{aligned}\left|uv^{f(p+1)+1}wx^{f(p+1)+1}y\right| &= \left|uvv^{f(p+1)}wxx^{f(p+1)}y\right| = \\ &= \left|uvwxy\right| + \left|v^{f(p+1)}\right| + \left|x^{f(p+1)}\right|\end{aligned} \qquad \text{(a)}$$

Poiché:

$$\left|\alpha^i\right| + \left|\beta^i\right| = i \cdot \left|\alpha\right| + i \cdot \left|\beta\right| = i \cdot \left(\left|\alpha\right| + \left|\beta\right|\right) = i \cdot \left|\alpha\beta\right|$$

la (a) può essere scritta nella forma:

$$\begin{aligned}\left|z\right| + f(p+1) \cdot \left|vx\right| &= f(p+1) + f(p+1) \cdot \left|vx\right| = \\ &= f(p+1) \cdot \left(1 + \left|vx\right|\right)\end{aligned} \qquad \text{(b)}$$

Ora denotiamo con k la lunghezza della stringa vx:

$$k = \left|vx\right|$$

Per la (2), $k > 0$ e per la (1) si ha che $k \le \left|vwx\right| \le p$.

Dunque:

$$0 < k \le p$$

Ma allora la (b) diventa:

$$f(p+1) \cdot \left(1 + \left|vx\right|\right) = f(p+1) \cdot (1+k) \qquad \text{(c)}$$

e, per la (3), deve risultare che:

$$f(p+1) \cdot (1+k)$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

sia un numero primo. Ma questo è un assurdo, poiché $k+1 \neq 1$ per ogni k.

Dunque:

$$uv^{f(p+1)+1}wx^{f(p+1)+1}y \notin L$$

contro la (3).

L'assurdo è derivato dall'aver supposto che L fosse libero da contesto. Possiamo concludere che L non è libero da contesto, in quanto L è infinito ma non verifica il Pumping Lemma per i linguaggi liberi.

Inoltre, se L fosse libero da contesto, il seguente sarebbe un teorema dell'aritmetica formale (*teoria assiomatica dei numeri*):

$$\begin{array}{lcll} t \text{ numero primo} & \Rightarrow & t+k\cdot i & \text{numero primo} \\ t \text{ numero primo} & \Rightarrow & t+2k\cdot i & \text{numero primo} \end{array} \qquad 0<k\leq p,\ \forall i,\ i\geq 0$$

Il teorema precedente è palesemente falso. Esso sarebbe un corollario del Pumping Lemma per i linguaggi liberi, nell'ipotesi che L fosse libero. Il perché è semplice.

È sufficiente considerare

$$z = a^t,\ t \text{ numero primo e } t > p\ .$$

Consideriamo le parole:

$$uv^{i+1}wx^{i+1}y \quad \forall i,\ i\geq 0$$

Per la (3),

$$uv^{i+1}wx^{i+1}y \in L \quad \forall i,\ i\geq 0$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Ma si ha:

$$\left|uv^{i+1}wx^{i+1}y\right| = \left|uvwxy\right| + \left|v^i x^i\right| = t + i \cdot \left|vx\right| \qquad \text{(a')}$$

Posto $k = \left|vx\right|$ e $0 < k \leq p$, per la (1) e la (2) si ha che (a′) diventa:

$$t + i \cdot \left|vx\right| = t + i \cdot k \quad \forall i,\ i \geq 0 \qquad \text{(b')}$$

e $t + i \cdot k$ deve essere un numero primo per un certo k, $0 < k \leq p$, e per tutti gli interi non negativi i ($i \geq 0$). Ma questo è assurdo.

4.6 Esercizi proposti

Esercizio 1

Stabilire se i seguenti linguaggi sono liberi da contesto e giustificare formalmente la risposta:

a) $\{ a^i b^j c^k : j = \min\{i, k\},\ i, j, k \geq 0 \}$

b) $\{ 0^i 1^j 2^i : 0 \leq i \leq j \}$

c) $\{ a^i b^j c^k : 0 \leq i < j < k \}$

d) $\{ a^n b^n c a^n b^n : n \text{ intero}, 0 \leq n \}$

e) $\{ a^i b^j c a^i b^j : 0 \leq i < j \}$

f) $\{ wcw : w \in \{a, b\}^* \}$

g) $\{ ww : w \in \{a, b, c\}^* \}$

h) $\{ a^r b^k : r > k^3,\ r, k > 0 \}$

i) $\{ a^i b^j c^k : 0 \leq i \leq j \leq k \}$

j) $\{ w : w \in \{a, b, c\}^*$, w ha lo stesso numero di a, di b e di c $\}$

k) $\{ w : w \in \{a, b, c\}^*$, w ha lo stesso numero di a, di b e di c $\} \cup \{a, b\}^*$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

l) $\{ a^i b^j c^k : j = \min\{i, k\}, i, j, k \geq 0 \} \cup \{ a^n b^n : n \geq 0 \}$

m) $\{ ww : w \in \{ a, b, c \}^* \} \cup \{ a, b \}^*$

n) $\{ a^i b^j : i = j = n^2, n \geq 0 \}$

o) $\{ a^m b^n c^{2n} : m < n, m, n$ interi positivi $\}$

p) $\{ a^n b^n c^m : n \leq m \leq 2n, m, n$ interi positivi $\}$

Esercizio 2

Stabilire se il seguente linguaggio:

$$L = \{ (ab)^n c^n d^n : n > 0 \}$$

è libero da contesto.

Giustificare formalmente la precedente risposta.

Determinare una grammatica che genera L.

Esercizio 3

Sia data la seguente grammatica:

$G = (X, V, S, P)$

ove $X = \{a, b, c\}$

$V = \{S, A, B, C\}$

$$\begin{aligned} P = \{ \quad & S \to AB, \\ & A \to CC \mid a, \\ & B \to b, \\ & C \to AA \mid c \quad \} \end{aligned}$$

Costruire l'albero di derivazione per la parola $w = caab$.

Descrivere in maniera succinta un sottoinsieme di cardinalità infinita di $L(G)$.

116 - LINGUAGGI LIBERI DA CONTESTO

5. Grammatiche e macchine

5.1 Classificazione delle grammatiche secondo Chomsky

Definizione 5.1 (Gerarchia di Chomsky (1956, 1959))

Sia $G = (X, V, S, P)$ una grammatica. Dalla definizione di grammatica, si ha:

$$P = \left\{ v \to w \;\middle|\; \begin{array}{l} v \in (X \cup V)^+ \text{ e } v \text{ contiene almeno un } NT, \\ w \in (X \cup V)^* \end{array} \right\}.$$

A seconda delle restrizioni imposte sulle regole di produzione, si distinguono le varie classi di grammatiche.

Tipo '0' Quando le stringhe che appaiono nella produzione $v \to w$ non sono soggette ad alcuna limitazione.

Tipo '1' **Dipendente da contesto**, quando le produzioni sono limitate alle forme:

(1) $yAz \to ywz$,

con $A \in V$, $y, z \in (X \cup V)^*$, $w \in (X \cup V)^+$;

(2) $S \to \lambda$,

purché S non compaia nella parte destra di alcuna produzione.

Tipo '2' **Libera da contesto**, quando le produzioni sono limitate alla forma:

$v \to w$ con $v \in V$.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Tipo '3' **Lineare destra**, quando le produzioni sono limitate alle forme:

(1) $A \to bC$ con $A, C \in V$ e $b \in X$;

(2) $A \to b$ con $A \in V$ e $b \in X \cup \{\lambda\}$.

Una grammatica di tipo '3' è detta lineare destra perché il *NT*, se c'è, compare a destra (nella parte destra della produzione). Un linguaggio generato da una tale grammatica è detto *di tipo '3'* o *lineare a destra*. ■

<u>Esempi di linguaggi di tipo '3'</u>

- Consideriamo la grammatica G_1:

$$S \to \lambda \mid 0 \mid 0S \mid 1 \mid 1S$$

  G_1 è lineare destra. $L(G_1) = \{0, 1\}^*$ (l'insieme di tutte le stringhe binarie).

- Consideriamo la grammatica G_2:

$$S \to \lambda \mid 0S \mid 1T$$
$$T \to 0T \mid 1S$$

G_2 è lineare destra.

$L(G_2) = \left\{ w \in \{0, 1\}^* \mid w \text{ ha un numero pari di } 1 \right\}$.

5.2 Teorema della Gerarchia

Dimostriamo formalmente che le quattro classi di linguaggi viste costituiscono una gerarchia (Figura 5.1).

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

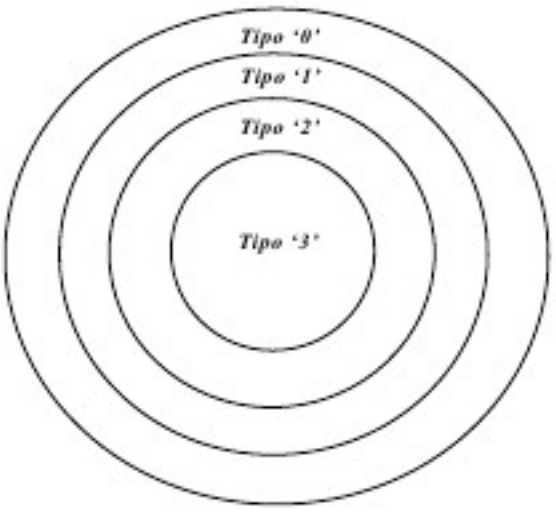


Figura 5.1

Vale infatti il seguente:

<u>Teorema 5.1</u> (della Gerarchia)

Denotiamo con $\mathcal{L}_i$ il seguente insieme:

$\mathcal{L}_i = \{L \subset X^* \mid L = L(G),\ G \text{ di tipo } i\}$ (classe dei linguaggi di tipo i).

La gerarchia di Chomsky è una gerarchia in senso stretto di classi di linguaggi:

$$\mathcal{L}_3 \underset{\neq}{\subset} \mathcal{L}_2 \underset{\neq}{\subset} \mathcal{L}_1 \underset{\neq}{\subset} \mathcal{L}_0$$

<u>Dimostrazione 5.1</u>

1) $\mathcal{L}_3 \underset{\neq}{\subset} \mathcal{L}_2$

Per dimostrare che la classe di linguaggi $\mathcal{L}_3$ è *inclusa propriamente* nella classe di linguaggi $\mathcal{L}_2$ si deve dimostrare che ogni linguaggio di tipo '3' può essere generato da una grammatica di tipo '2' e che esiste almeno un linguaggio C.F. (di tipo '2') che non può essere generato da una grammatica di tipo '3'.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

$\mathcal{L}_3 \subseteq \mathcal{L}_2$ discende direttamente dalle definizioni di linguaggio di tipo '3' e di grammatica di tipo '2'. Infatti, si osserva facilmente che ogni grammatica di tipo '3' è anche una grammatica di tipo '2'.

$\mathcal{L}_3 \neq \mathcal{L}_2$: posponiamo questa dimostrazione.

Mostreremo (si veda Esercizio 7.8) che: $L = \{a^k b^k \mid k > 0\}$ non è di tipo '3' (abbiamo già determinato una grammatica di tipo '2' che genera L).

2) $\mathcal{L}_2 \underset{\neq}{\subset} \mathcal{L}_1$

Abbiamo già osservato che le produzioni C.F. sono un caso particolare delle produzioni C.S. con l'unica eccezione rappresentata dalle produzioni:

$$A \to \lambda, \qquad A \neq S$$

che sono C.F. ma ***non*** C.S.

Dunque:

$$\forall L : L \in \mathcal{L}_2 \overset{def}{\Leftrightarrow} \exists G, G \text{ è C.F.} : L = L(G).$$

Se $A \to \lambda$, $A \in V \setminus \{S\}$ non è una produzione di G, allora G è anche C.S. (di tipo '1') e l'asserto è dimostrato. Il problema sorge se G ha almeno una λ-produzione. In tal caso, ci avvaliamo del seguente risultato:

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

5.3 Lemma della stringa vuota

Lemma 5.2 (della stringa vuota)

Sia $G = (X, V, S, P)$ una grammatica C.F. con almeno una λ-produzione. Allora esiste una grammatica C.F. G' tale che:

i) $L(G) = L(G')$ (G' è equivalente a G);

ii) se $\lambda \notin L(G)$ allora in G' non esistono produzioni del tipo $A \to \lambda$;

iii) se $\lambda \in L(G)$ allora in G' esiste un'unica produzione $S' \to \lambda$, ove S' è il simbolo iniziale di G' ed S' non compare nella parte destra di alcuna produzione di G'.

Dimostrazione 5.2

Si rimanda la dimostrazione di questo lemma alla trattazione della forma normale di Greibach per le grammatiche libere da contesto (Paragrafo 8.4).

Riprendiamo la dimostrazione di $\mathcal{L}_2 \underset{\neq}{\subset} \mathcal{L}_1$

$$\forall L : L \in \mathcal{L}_2 \overset{def}{\Leftrightarrow} \exists G, G \text{ è di tipo '2'} : L = L(G).$$

Se G ha almeno una λ-produzione, utilizziamo il Lemma della stringa vuota per determinare una grammatica C.F. G' equivalente a G, ma priva di λ-produzioni (al più, in G' compare la produzione $S' \to \lambda$, ed S' non compare nella parte

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

destra di alcuna produzione di G'). G' è di tipo '1'. Questo dimostra che $\mathcal{L}_2 \subseteq \mathcal{L}_1$.

Dimostriamo che $\mathcal{L}_2 \neq \mathcal{L}_1$.

Dall'Esercizio 4.2, segue immediatamente che:

$$L = \{a^n b^n c^n \mid n > 0\}$$

è di tipo '1' ma non di tipo '2'. Si osservi che, per asserire che L è di tipo '1', ci siamo avvalsi del teorema che stabilisce l'equivalenza delle classi di linguaggi contestuali e monotoni.

3) $\mathcal{L}_1 \underset{\neq}{\subset} \mathcal{L}_0$

Non lo dimostriamo formalmente. La dimostrazione comporta la conoscenza degli automi limitati lineari e delle macchine di Turing (che riconoscono linguaggi di tipo '0' o ricorsivamente enumerabili). Ci limitiamo ad osservare che $\mathcal{L}_1 \subseteq \mathcal{L}_0$ discende direttamente dalle definizioni di linguaggio di tipo '1' e di grammatica di tipo '0'. *c.v.d.*

5.4 Operazioni sui linguaggi

<u>Definizione 5.2</u> (Operazioni sui linguaggi)

Siano L_1 ed L_2 due linguaggi definiti su uno stesso alfabeto X ($L_1, L_2 \subseteq X^*$).

(i) L'*unione* di L_1 ed L_2 è:

$$L_1 \cup L_2 = \{w \mid w \in L_1 \vee w \in L_2\} ;$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

(ii) La *concatenazione* di L_1 ed L_2 (anche detta il *prodotto* di L_1 ed L_2 o "L_1 *punto* L_2") è:

$$L_1 \cdot L_2 = \{w \mid w = w_1 w_2,\ \ w_1 \in L_1,\ \ w_2 \in L_2\};$$

(iii) L'*iterazione* di L_1 (o *chiusura riflessiva e transitiva* di L_1 rispetto all'operazione di concatenazione, anche detta *stellatura* di L_1 o "L_1 *star*" o *chiusura di Kleene*) è:

$$L_1^* = \{w \mid w = w_1 w_2 ... w_n,\ \ n \geq 0 \ \text{ e } \ \forall i : w_i \in L_1\};$$

(iv) Il *complemento* di L_1 è:

$$\overline{L_1} = X^* - L_1$$

(v) L'*intersezione* di L_1 ed L_2 è:

$$L_1 \cap L_2 = \{w \mid w \in L_1 \wedge w \in L_2\}$$ ■

<u>Proprietà 5.1</u>

Dati $L_1, L_2, L_3 \subseteq X^*$ $\quad (\equiv L_1, L_2, L_3 \in 2^{X^*})$, risulta:

1) $(L_1 \cdot L_2) \cdot L_3 = L_1 \cdot (L_2 \cdot L_3)$ (proprietà associativa);

2) $L_1 \cdot L_2 \neq L_2 \cdot L_1$;

3) $L_1 \cdot \{\lambda\} = \{\lambda\} \cdot L_1 = L_1$, ($\{\lambda\}$ è l'elemento neutro rispetto all'operazione di concatenazione di linguaggi).

Dunque anche $\left(2^{X^*},\ \cdot\right)$ è un monoide;

4) $L_1 \cdot \varnothing = \varnothing \cdot L_1 = \varnothing$ ($\varnothing$ è l'elemento assorbente);

5) Se $\lambda \in L_1$: $\quad L_2 \subseteq L_1 \cdot L_2$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

$$L_2 \subseteq L_2 \cdot L_1$$

6) Se $\lambda \in L_2$: $\quad L_1 \subseteq L_1 \cdot L_2$

$$L_1 \subseteq L_2 \cdot L_1$$

<u>Esempio 5.1</u>

Si considerino i linguaggi:

$$L_1 = \{a^{2n} \mid n \geq 0\} \qquad L_2 = \{b, cc\}$$

$$L_1 \cdot L_2 = \{b, cc, a^2b, a^2cc, a^4b, a^4cc, ...\}$$

$$L_2 \cdot L_1 = \{b, cc, ba^2, cca^2, ba^4, cca^4, ...\}$$

Dunque:

$$L_2 \subset L_1 \cdot L_2 \qquad \text{e} \qquad L_2 \subset L_2 \cdot L_1$$

mentre:

$$L_1 \not\subseteq L_1 \cdot L_2 \qquad \text{e} \qquad L_1 \not\subseteq L_2 \cdot L_1$$

$$L_1 \cup L_2 = L_2 \cup L_1 = \{\lambda, b, cc, a^2, a^4, a^6, ...\}$$

$$L_2^* = \{\lambda, b, cc, bb, bcc, ccb, bbb, cccc, ...\}$$

$$L_1^* = \{\lambda, a^2, a^4, a^6, ...\} = L_1$$

<u>Definizione 5.3</u> (Potenza di un linguaggio)

Sia L un linguaggio definito su un alfabeto X. Dicesi *potenza n-esima* di L, e si denota con L^n, $n \geq 0$, il seguente linguaggio:

$$L^n = \begin{cases} \{\lambda\} & \text{se } n = 0 \\ L^{n-1} \cdot L & \text{altrimenti} \end{cases}$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Posto:

$$L^+ = \bigcup_{i \geq 1} L^i$$

si ha:

$$L^* = \{\lambda\} \cup L^+ = L^0 \cup L^+ = \bigcup_{i \geq 0} L^i$$

L^+ è detta chiusura transitiva rispetto alla operazione di concatenazione. ■

Dunque, si ha:

$$L^0 = \{\lambda\}$$
$$L^1 = L^0 \cdot L = L$$
$$L^2 = L^1 \cdot L = (L^0 \cdot L) \cdot L$$
$$L^3 = L^2 \cdot L = (L^1 \cdot L) \cdot L = \big((L^0 \cdot L) \cdot L\big) \cdot L$$

<u>Proposizione 5.3</u>

Sia L un linguaggio definito su un alfabeto X. Si ha:

$$L^* = \bigcup_{i \geq 0} L^i = \{\lambda\} \cup L \cup L^2 \cup L^3 \cup \ldots$$

<u>Esempio 5.2</u>

Si consideri nuovamente il linguaggio $L_2 = \{b, cc\}$. Si ha:

$$L_2^{\ 0} = \{\lambda\},$$
$$L_2^{\ 1} = L_2,$$
$$L_2^{\ 2} = \{bb, bcc, ccb, cccc\},$$
$$L_2^{\ *} = L_2^{\ 0} \cup L_2^{\ 1} \cup L_2^{\ 2} \cup \ldots = \{\lambda, b, cc, bb, bcc, ccb, cccc, bbb, \ldots\}.$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

5.5 Proprietà di chiusura delle classi di linguaggi

Definizione 5.4 (Linguaggio - Classe di linguaggi)

1) *L linguaggio* definito su $X \overset{def}{\Leftrightarrow} L \subseteq X^* \Leftrightarrow L \in 2^{X^*}$;

2) $\mathcal{L}$ *classe di linguaggi* su $X \overset{def}{\Leftrightarrow} \mathcal{L} \subseteq 2^{X^*} \Leftrightarrow \mathcal{L} \in 2^{2^{X^*}}$. ■

Definizione 5.5 (Chiusura)

Sia $\mathcal{L}$ una classe di linguaggi su X.

Sia α un'operazione binaria sui linguaggi di $\mathcal{L}$:

$$\alpha : 2^{X^*} \times 2^{X^*} \to 2^{X^*}, \qquad (L_1, L_2) \mapsto \alpha(L_1, L_2).$$

Sia β un'operazione unaria sui linguaggi di $\mathcal{L}$:

$$\beta : 2^{X^*} \to 2^{X^*}, \qquad L \mapsto \beta(L).$$

$\mathcal{L}$ è *chiusa* rispetto ad $\alpha \overset{def}{\Leftrightarrow} \forall\, L_1, L_2 \in \mathcal{L} : \alpha(L_1, L_2) \in \mathcal{L}$.

$\mathcal{L}$ è *chiusa* rispetto a $\beta \overset{def}{\Leftrightarrow} \forall\, L_1 \in \mathcal{L} : \beta(L_1) \in \mathcal{L}$. ■

Esempio 5.3

$\mathcal{L}$ è chiusa rispetto all'iterazione se $\forall\, L_1 \in \mathcal{L} : L_1^* \in \mathcal{L}$.

Teorema 5.3

La classe dei linguaggi di tipo i, $i = 0, 1, 2, 3$, è chiusa rispetto alle operazioni di *unione, concatenazione* ed *iterazione*.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Dimostrazione 5.3

La dimostrazione di questo teorema è *costruttiva*.

Siano L_1 ed L_2 due linguaggi:

$$L_1 = L(G_1) \qquad G_1 = (X, V_1, S_1, P_1)$$

$$L_2 = L(G_2) \qquad G_2 = (X, V_2, S_2, P_2)$$

Assumiamo che: [7]

$$V_1 \cap V_2 = \varnothing \qquad S \notin V_1 \cup V_2$$

Poniamo:

$$V = V_1 \cup V_2 \cup \{S\}$$

Lo schema generale della dimostrazione è il seguente:

- consideriamo un'operazione alla volta (denotata con α);
- date G_1 e G_2, costruiamo G;
- si dimostra che, se G_1 e G_2 sono di tipo i, allora G è di tipo i;
- si dimostra che $L(G) = \alpha(L_1, L_2)$ e dunque la classe di linguaggi $\mathcal{L}_i$ è chiusa rispetto alla operazione α.

UNIONE

Costruiamo la grammatica:

$$G_3 = (X, V, S, P_3)$$

ove:

$$P_3 = \{S \to S_1,\ S \to S_2\} \cup P_1 \cup P_2$$

---

[7] Nel caso in cui tale assunzione non sia vera, ridenominiamo i nonterminali in comune.

Osserviamo che, se G_1 e G_2 sono entrambe di tipo i, $i = 0, 1, 2$, lo è anche G_3.

In ciascuno di questi casi, si ha:

$$L(G_3) = L_1 \cup L_2$$

Infatti una derivazione da S in G_3 deve necessariamente iniziare o con:

$S \Rightarrow S_1$ ed in tal caso può generare unicamente parole di $L(G_1)$

o con

$S \Rightarrow S_2$ ed in tal caso genera una parola di $L(G_2)$.

Dunque, risulta dimostrato che <u>$\mathcal{L}_0$, $\mathcal{L}_1$ e $\mathcal{L}_2$, sono chiuse rispetto all'unione</u>.

Però, se G_1 e G_2 sono di tipo '3', G_3 non è lineare destra, perché le produzioni $S \to S_1$ ed $S \to S_2$ non sono ammesse.

Per avere ancora produzioni lineari destre che simulino il passo iniziale di una derivazione in G_1 ed anche il passo iniziale di una derivazione in G_2, costruiamo pertanto la grammatica:

$$G_4 = (X, V, S, P_4)$$

ove:

$$P_4 = \{S \to w \mid S_1 \to w \in P\} \cup \{S \to w \mid S_2 \to w \in P_2\} \cup P_1 \cup P_2$$

G_4 è lineare destra se G_1 e G_2 lo sono e inoltre:

$$L(G_4) = L_1 \cup L_2$$

ed <u>$\mathcal{L}_3$ è chiusa rispetto all'unione</u>.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Esempio 5.4

$$G_1 \text{ con } P_1 = \{S_1 \to aA,\ A \to b\}$$
$$G_2 \text{ con } P_2 = \{S_2 \to cC,\ C \to b\}$$
$$V_1 \cap V_2 = \varnothing \quad \text{mentre} \quad X_1 = \{a,b\} \text{ ed } X_2 = \{b,c\}$$

Consideriamo

$$X = \{a,b,c\} = X_1 \cup X_2$$

e

$$G_4 \text{ con } P_4 = \{S \to aA,\ S \to cC,\ S_1 \to aA,\ A \to b,\ S_2 \to cC,\ C \to b\}$$
$$L(G_1) = \{ab\} \qquad L(G_2) = \{cb\} \qquad L(G_4) = \{ab, cb\}$$

dunque

$$L(G_4) = L(G_1) \cup L(G_2)$$

e

$$G_4 = \left(X = \{a,b,c\},\ V = V_1 \cup V_2 \cup \{S\} = \{S, S_1, S_2, A, C\},\ S,\ P_4\right).$$

Osserviamo che S_1, S_2 e le produzioni $S_1 \to aA$, $S_2 \to cC$ non sono di alcuna utilità, in quanto non possono essere utilizzate in alcuna derivazione da S in G_4. S_1 ed S_2 sono detti *nonterminali inutili* (Definizione 8.12) perché non compariranno mai all'interno di una forma di frase generata da S in G_4.

La grammatica G_4, per effetto della eliminazione dei nonterminali inutili, diventa:

$$G_4' = \left(X' = \{a,b,c\},\ V' = \{S, A, C\},\ S,\ P_4' = \{S \to aA \mid cC,\ A \to b,\ C \to b\}\right)$$

e $L(G_4') = L(G_4)$ (dimostrazione per esercizio).

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

CONCATENAZIONE

Costruiamo la grammatica:

$$G_5 = (X, V, S, P_5)$$

ove:

$$P_5 = \{S \to S_1 S_2\} \cup P_1 \cup P_2$$

Osserviamo che, se G_1 e G_2 sono entrambe di tipo i, $i = 0, 1, 2$, lo è anche G_5.

Se G_1 e G_2 sono C.F. (tipo '2'), allora si ha:

$$L(G_5) = L_1 \cdot L_2$$

in quanto ogni derivazione da S in G_5 ha la seguente "struttura":

$$S \Rightarrow S_1 S_2 \overset{*}{\underset{G_1}{\Rightarrow}} w_1 S_2 \overset{*}{\underset{G_2}{\Rightarrow}} w_1 w_2$$

ove, evidentemente, si ha:

$$S_1 \overset{*}{\Rightarrow} w_1 \qquad\qquad S_2 \overset{*}{\Rightarrow} w_2$$

Il seguente controesempio mostra che la condizione che G_1 e G_2 siano di tipo '2' è fondamentale per la validità di $L(G_5) = L_1 \cdot L_2$.

<u>Esempio 5.5</u> (Controesempio)

Consideriamo G_1 con $P_1 = \{S_1 \to b\}$ e G_2 con $P_2 = \{bS_2 \to bb\}$.

Da cui $L_1 = L(G_1) = \{b\}$ ed $L_2 = L(G_2) = \varnothing$.

Dunque:

$$L_1 \cdot L_2 = \varnothing .$$

Se costruiamo G_5, si ha:

$$P_5 = \{S \to S_1 S_2,\ S_1 \to b,\ bS_2 \to bb\}$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

ed $L(G_5) \neq \varnothing$ in quanto $S \overset{*}{\underset{G_5}{\Rightarrow}} bb$ attraverso la seguente derivazione:

$$S \Rightarrow S_1 S_2 \Rightarrow bS_2 \Rightarrow bb\ .$$

Dunque la G_5 va bene solo per grammatiche di tipo '2' e risulta dimostrato che $\mathcal{L}_2$ è chiusa rispetto alla concatenazione. La dimostrazione fatta non va bene per grammatiche di tipo '0' e di tipo '1', in quanto entrambe queste classi di grammatiche presentano produzioni dipendenti da contesto.

In presenza di grammatiche di tali tipi è necessario impedire che le derivazioni da S_2 si servano di precedenti derivazioni da S_1 e/o viceversa. In altri termini, è necessario evitare interferenze tra derivazioni da S_1 e derivazioni da S_2 nella definizione dei contesti. È possibile ottenere ciò considerando copie distinte di X in derivazioni distinte (da S_1 e da S_2), in modo che nessun NT derivato da S_1 possa far uso di parte di una forma di frase derivata da S_2 come contesto per l'applicazione di una produzione (e viceversa).

Siano pertanto:

$$X' = \{x' \mid x \in X\} \qquad \text{e} \qquad X'' = \{x'' \mid x \in X\}$$

due copie distinte di X tali che:

$$X' \cap X'' = \varnothing \qquad X' \cap X = \varnothing \qquad X' \cap V = \varnothing$$
$$X'' \cap X = \varnothing \qquad X'' \cap V = \varnothing$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

e sia P_1' l'insieme delle produzioni ottenute da P_1 sostituendo ogni occorrenza di un terminale x in X con il corrispondente nonterminale x' in X':

$$P_1' = P_1\left[x'/_x\right]$$

Similmente, sia P_2'' l'insieme delle produzioni ottenute da P_2 sostituendo ogni occorrenza di un terminale x in X con il corrispondente x'' in X'':

$$P_2'' = P_2\left[x''/_x\right]$$

In questo modo evitiamo l'interferenza tra contesti.

Costruiamo ora la grammatica:

$$G_6 = (X,\ V \cup X' \cup X'',\ S,\ P_6)$$

ove:

$$P_6 = \{S \to S_1S_2\} \cup P_1' \cup P_2'' \cup \{x' \to x \mid x \in X\} \cup \{x'' \to x \mid x \in X\}$$

Se G_1 e G_2 sono entrambe di tipo i, $i = 0, 1$, lo è anche G_6. Inoltre, si ha:

$$L(G_6) = L_1 \cdot L_2$$

Il controesempio visto in precedenza non dà più problemi:

$$P_1' = \{S_1 \to b'\} \qquad\qquad P_2'' = \{b''S_2 \to b''b''\}$$

e

$$G_6 = (X,\ V \cup \{b'\} \cup \{b''\},\ S,\ P_6)$$

dove

$$P_6 = \{S \to S_1S_2,\ S_1 \to b',\ b''S_2 \to b''b'',\ b' \to b,\ b'' \to b\}$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

ed $L(G_6) = \varnothing$ in quanto $S \overset{*}{\underset{G_6}{\not\Rightarrow}} bb$, $S \Rightarrow S_1S_2 \Rightarrow b'S_2 \Rightarrow bS_2$.

Risulta così dimostrato che $\mathcal{L}_0$ ed $\mathcal{L}_1$ sono chiuse rispetto alla concatenazione. È immediato osservare che, se G_1 e G_2 sono di tipo '3' né G_5 né G_6 sono di tipo '3', per la presenza della produzione $S \to S_1S_2$.

Dobbiamo simulare l'effetto della produzione $S \to S_1S_2$, che determina la concatenazione delle parole generate da S_1 e da S_2. A tale scopo osserviamo che, data una grammatica di tipo '3', ogni forma di frase derivata dal simbolo iniziale di tale grammatica ha due peculiarità:

1) in essa compare al più un *NT*;
2) se in essa compare un *NT*, questo è il simbolo più a destra

$$\left(S \Rightarrow x_1A \Rightarrow x_1x_2B \overset{*}{\Rightarrow} x_1x_2x_3...x_nN \Rightarrow x_1x_2x_3...x_nx_{n+1} \right).$$

Modifichiamo pertanto ogni produzione del tipo $A \to b$ in G_1 in modo che essa non costituisca l'ultima produzione applicata in una derivazione da S_1 in G_1, ma possa innescare una derivazione da S_2 in G_2. Poniamo dunque a destra della b il simbolo iniziale S_2 di G_2.

Le produzioni del tipo $A \to b$ in G_1 vengono trasformate in: $A \to bS_2$.

Costruiamo dunque la grammatica:

$$G_7 = (X,\ V - \{S\},\ S_1,\ P_7)$$

ove:

$$\begin{aligned} P_7 = & \{A \to bB \mid A \to bB \in P_1\} \cup \\ & \cup \{A \to bS_2 \mid A \to b \in P_1,\ b \neq \lambda\} \cup \\ & \cup \{A \to bS_2 \mid A \to bB \in P_1,\ B \to \lambda \in P\} \cup \quad (*) \\ & \cup \{S_1 \to w \mid S_2 \to w \in P_2,\ S_1 \to \lambda \in P_1\} \cup \quad (**) \\ & \cup P_2 \end{aligned}$$

Le (*) e (**) sono state introdotte per garantire la correttezza della grammatica generata, anche in presenza di λ-produzioni in G_1.

G_7 è di tipo '3' se G_1 e G_2 lo sono ed inoltre:

$$L(G_7) = L_1 \cdot L_2$$

ed <u>$\mathcal{L}_3$ è chiusa rispetto alla concatenazione</u>.

ITERAZIONE

Costruiamo la grammatica:

$$G_8 = (X,\ V_1 \cup \{S\},\ S,\ P_8)$$

ove:

$$P_8 = \{S \to \lambda,\ S \to S_1 S\} \cup P_1$$

Data la grammatica $G_1 = (X,\ V_1,\ S_1,\ P_1)$ che genera L_1, la grammatica G_8 genera la parola vuota λ e tutte le parole che si possono ottenere per concatenazione di parole generate da S_1 in G_1.

Si ha infatti:

$$S \overset{n}{\underset{G_8}{\Rightarrow}} \underbrace{S_1 S_1 \ldots S_1}_{n} S \Rightarrow S_1 S_1 \ldots S_1 \overset{*}{\Rightarrow} w_1 w_2 \ldots w_n$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

con $w_i \in L_1,\ i = 1, 2, ..., n$.

Vediamo per quali classi di grammatiche la G_8 è la grammatica che stiamo cercando:

- se G_1 è di tipo '3', G_8 non è di tipo '3', perché $S \to S_1S$ non è lineare destra;
- se G_1 è di tipo '2', G_8 è di tipo '2' e si ha:

  $$L(G_8) = L_1^*$$

  e risulta dimostrato che <u>$\mathcal{L}_2$ è chiusa rispetto all'iterazione.</u>
- se G_1 è di tipo '1', G_8 non è di tipo '1', perché S compare nella parte destra della produzione $S \to S_1S$ ed $S \to \lambda$ è una produzione in P_8; possiamo trasformare facilmente G_8 nella grammatica equivalente G_8', definita come segue:

  $$G_8' = (X,\ V_1 \cup \{S, S'\},\ S,\ P_8')$$
  $$P_8' = \{S \to \lambda \mid S',\ S' \to S_1 \mid S_1S'\} \cup P_1$$

  ma G_8' incorre nello stesso problema di interferenza nella definizione dei contesti visto nella dimostrazione per la concatenazione.
- se G_1 è di tipo '0', lo è anche G_8, ma si incorre nello stesso problema di interferenza nella definizione dei contesti visto nella dimostrazione per la concatenazione.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Il problema dell'interferenza dei contesti, comune agli ultimi due casi - G_1 di tipo i, $i = 0, 1$ - esiste ancora come mostrato dal seguente controesempio:

<u>Esempio 5.6</u> (Controesempio)

Consideriamo G_1 con $P_1 = \{S_1 \to b,\ bS_1 \to bc\}$ allora $L_1 = L(G) = \{b\}$ e $L_1^* = \{b\}^*$.

Del resto $P_8 = \left\{ S \to \overset{(1)}{\lambda} \mid \overset{(2)}{S'},\ S' \to \overset{(3)}{S_1} \mid \overset{(4)}{S_1 S'},\ S_1 \overset{(5)}{\to} b,\ bS_1 \overset{(6)}{\to} bc \right\}$.

Se consideriamo la derivazione

$$S \underset{(2)}{\Rightarrow} S' \underset{(4)}{\Rightarrow} S_1 S' \underset{(3)}{\Rightarrow} S_1 S_1 \underset{(5)}{\Rightarrow} bS_1 \underset{(6)}{\Rightarrow} bc$$

si ottiene che $bc \in L(G_8')$ e quindi $L(G_8') \neq L_1^*$.

Il problema dell'interferenza dei contesti può essere quindi risolto come segue.

Eliminiamo dapprima le produzioni del tipo $S_1 \to \lambda$ (si veda il Lemma della stringa vuota - Lemma 5.2).

Utilizziamo gli insiemi ausiliari di NT X' e X'' per evitare interferenze:

$$X' = \{x' \mid x \in X\} \qquad X'' = \{x'' \mid x \in X\}$$

$$X' \cap X'' = \varnothing \qquad X' \cap X = \varnothing \qquad X' \cap V_1 = \varnothing$$

$$X'' \cap X = \varnothing \qquad X'' \cap V_1 = \varnothing$$

Costruiamo due copie di G_1:

$$G_1' = (X,\ V_1 \cup X',\ S_1,\ P_1')$$

$$G_1'' = (X,\ V_1 \cup X'',\ S_2,\ P_1'')$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

ove:

$$P_1' = P_1\left[x'/x\right] \qquad P_1'' = P_1\left[x''/x\right]$$

Infine, combiniamo G_1', G_1'' con produzioni che costruiscono sequenze finite di copie di S_1 ed S_2 che si alternano, in modo da ottenere L_1^*.

Dunque, la grammatica che otteniamo è:

$$G_9 = (X,\ V_1 \cup X' \cup X'' \cup \{S, S_1', S_2, S_2'\},\ S,\ P_9)$$

ove:

$$P_9 = \{S \to \lambda \mid S_1' \mid S_2',\ S_1' \to S_1 \mid S_1 S_2',\ S_2' \to S_2 \mid S_2 S_1'\} \cup$$
$$\cup P_1' \cup P_2' \cup \{x' \to x \mid x \in X\} \cup \{x'' \to x \mid x \in X\}$$

Se G_1 è di tipo i, $i = 0,1$, lo è anche G_9 e si ha:

$$L(G_9) = L_1^*$$

e risulta dimostrato che <u>$\mathcal{L}_0$ e $\mathcal{L}_1$ sono chiuse rispetto all'iterazione.</u>

Resta da dimostrare che $\mathcal{L}_3$ è chiusa rispetto all'iterazione.

Per costruire la nuova grammatica, introduciamo dapprima un nuovo simbolo iniziale S e la produzione $S \to \lambda$ che genera la stringa vuota.

Inoltre, eliminiamo da P_1, se c'era, la produzione $S_1 \to \lambda$ ed aggiungiamo una produzione $S \to w$ per ogni produzione "iniziale" $S_1 \to w$ in P_1.

Infine, per ogni produzione la cui parte destra contiene solo un terminale, del tipo $A \to b$, nell'insieme delle produzioni che stiamo costruendo, aggiungiamo la produzione $A \to bS$ in modo

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

che, avendo derivato una forma di frase del tipo $w_1 w_2 ... w'_j A$ tale che ogni sottostringa $w_1, w_2, ..., w_j = w'_j b$ è una parola di L_1, abbiamo la possibilità di terminare la derivazione con $w_1 w_2 ... w_j$ o di continuarla generando la forma di frase $w_1 w_2 ... w_j S$, che consente di generare una parola più lunga di L_1^*.

Si noti che, per garantire la correttezza della grammatica generata anche in presenza di λ-produzioni in G_1, occorre aggiungere una produzione $A \to bS$ per ogni produzione $A \to bB$ <u>nell'insieme delle produzioni che stiamo costruendo</u>, quando $B \to \lambda$ è pure una produzione di tale insieme, alla stregua di quanto fatto per la concatenazione in $\mathcal{L}_3$.

Costruiamo dunque la grammatica:

$$G_{10} = (X,\ V_1 \cup \{S\},\ S,\ P_{10})$$

ove:

$$\begin{aligned} P_{10} = \{S \to \lambda\} \cup \big(P_1 - \{S_1 \to \lambda\}\big) \cup \{S \to w \mid S_1 \to w \in P_1\} \cup \\ \cup \{A \to bS \mid A \to b \in P_{10},\ b \neq \lambda\} \cup \\ \cup \{A \to bS \mid A \to bB \in P_{10},\ B \to \lambda \in P_{10}\} \end{aligned}$$

Si noti che la definizione di P_{10} è ricorsiva. Se G_1 è di tipo '3', lo è anche G_{10} e si ha:

$$L(G_{10}) = L_1^*$$

e risulta dimostrato che <u>$\mathcal{L}_3$ è chiusa rispetto all'iterazione.</u> *c.v.d.*

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Per la loro importanza pratica, riassumiamo le modalità di costruzione delle grammatiche che generano i linguaggi *unione/concatenazione/iterazione* di L_1 ed L_2 nella Tavola 1 che segue.

	UNIONE	***CONCATENAZIONE***	***ITERAZIONE***
$\mathcal{L}_0$			Eliminiamo le produzioni del tipo: $S_1 \to \lambda$ $X' = \{x' \mid x \in X\} \quad X'' = \{x'' \mid x \in X\}$ $X' \cap X'' = \varnothing \quad X' \cap X = \varnothing \quad X' \cap V_1 = \varnothing$ $X'' \cap X = \varnothing \quad X'' \cap V_1 = \varnothing$ Costruiamo due copie di G_1: $G_1' = (X, V_1 \cup X', S_1, P_1')$ $G_1'' = (X, V_1 \cup X'', S_2, P_1'')$ $P_1' = P_1\left[{}^{x'}/_{x}\right] \quad P_1'' = P_1\left[{}^{x''}/_{x}\right]$ $G_9 = (X, V_1 \cup X' \cup X'' \cup \{S, S_1', S_2, S_2'\}, S, P_9)$ $P_9 = \{S \to \lambda \mid S_1' \mid S_2',\ S_1' \to S_1 \mid S_1 S_2',\ S_2' \to S_2 \mid S_2 S_1'\} \cup$ $\cup P_1' \cup P_2' \cup \{x' \to x \mid x \in X\} \cup \{x'' \to x \mid x \in X\}$
$\mathcal{L}_1$	$G_3 = (X, V, S, P_3)$ $P_3 = \{S \to S_1,\ S \to S_2\} \cup$ $\cup P_1 \cup P_2$	$X' = \{x' \mid x \in X\} \quad X'' = \{x'' \mid x \in X\}$ $X' \cap X'' = \varnothing \quad X' \cap X = \varnothing \quad X' \cap V = \varnothing$ $X'' \cap X = \varnothing \quad X'' \cap V = \varnothing$ $P_1' = P_1\left[{}^{x'}/_{x}\right] \quad P_2'' = P_2\left[{}^{x''}/_{x}\right]$ $G_6 = (X, V \cup X' \cup X'', S, P_6)$ $P_6 = \{S \to S_1 S_2\} \cup P_1' \cup P_2' \cup$ $\cup \{x' \to x \mid x \in X\} \cup$ $\cup \{x'' \to x \mid x \in X\}$	$G_8 = \{X, V_1 \cup \{S\}, S, P_8\}$ $P_8 = \{S \to \lambda \mid S_1 S\} \cup P_1$
$\mathcal{L}_2$		$G_5 = (X, V, S, P_5)$ $P_5 = \{S \to S_1 S_2\} \cup P_1 \cup P_2$	
$\mathcal{L}_3$	$G_4 = (X, V, S, P_4)$ $P_4 = \{S \to w \mid S_1 \to w \in P_1\} \cup$ $\cup \{S \to w \mid S_2 \to w \in P_2\} \cup$ $\cup P_1 \cup P_2$	$G_7 = (X, V - \{S\}, S_1, P_7)$ $P_7 = \{A \to bB \mid A \to bB \in P_1\} \cup$ $\cup \{A \to bS_2 \mid A \to b \in P_1,\ b \neq \lambda\} \cup$ $\cup \{A \to bS \mid A \to bB \in P_1,\ B \to \lambda \in P_1\} \cup$ $\cup \{S_1 \to w \mid S_2 \to w \in P_2,\ S_1 \to \lambda \in P_1\} \cup P_2$	$G_{10} = (X, V_1 \cup \{S\}, S, P_{10})$ $P_{10} = \{S \to \lambda\} \cup (P_1 - \{S_1 \to \lambda\}) \cup$ $\cup \{S \to w \mid S_1 \to w \in P_1\} \cup$ $\cup \{A \to bS \mid A \to b \in P_{10},\ b \neq \lambda\} \cup$ $\cup \{A \to bS \mid A \to bB \in P_{10},\ B \to \lambda \in P_{10}\}$

Tavola 1. Proprietà di Chiusura

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Teorema 5.4

1) La classe dei linguaggi lineari destri (tipo '3') è chiusa rispetto al complemento ed all'intersezione.
2) La classe dei linguaggi non contestuali (tipo '2') non è chiusa rispetto al complemento ed all'intersezione.
3) La classe dei linguaggi contestuali (tipo '1') è chiusa rispetto al complemento (e dunque anche rispetto all'intersezione).
4) La classe dei linguaggi di tipo '0' non è chiusa rispetto al complemento.

Dimostrazione 5.4

1) Dimostreremo prossimamente la chiusura di $\mathcal{L}_3$ rispetto al complemento. Dimostriamola rispetto all'intersezione:

$$L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}} \qquad \text{(Leggi di De Morgan)}$$

La chiusura di $\mathcal{L}_3$ rispetto all'intersezione discende direttamente da questo risultato.

2) Consideriamo i linguaggi:

$$L_1 = \{a^n b^n c^m \mid n,m > 0\} \qquad L_2 = \{a^n b^m c^m \mid n,m > 0\}$$

L_1 e L_2 sono linguaggi liberi (perché? dimostrarlo), mentre $L_1 \cap L_2$ non lo è:

$$L_1 \cap L_2 = \{a^k b^k c^k \mid k > 0\}\,.$$

Il complemento è: $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$.

Dunque, se L_1, $L_2 \in \mathcal{L}_2$ e se $\mathcal{L}_2$ fosse chiusa rispetto al complemento, …

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

3) Un risultato recente ha stabilito che $\mathcal{L}_1$ è chiusa rispetto al complemento (e quindi all'intersezione). Non se ne conosce la dimostrazione.
4) Non lo dimostriamo. *c.v.d.*

	$\overline{L}$	$L_1 \cap L_2$	L^R	$L_1 \cap Q \ (Q \in \mathcal{L}_3)$
$\mathcal{L}_0$	non r. e. (no)	tipo 0 (sì)	tipo 0	tipo 0
$\mathcal{L}_1$	tipo 1 (sì)	tipo 1 (sì)	tipo 1	tipo 1
$\mathcal{L}_2$	tipo 1 (no)	tipo 1 (no)	tipo 2	tipo 2
$\mathcal{L}_3$	tipo 3 (sì)	tipo 3 (sì)	tipo 3	tipo 3

5.6 L'operazione di riflessione

Definizione 5.6 (Stringa riflessa)

Sia w una parola su un alfabeto $X = \{x_1, x_2, ..., x_k\}$, $w = x_{i_1} x_{i_2} ... x_{i_{n-1}} x_{i_n}$. Dicesi *stringa riflessa* (o *riflessione*) di w la stringa:

$$w^R = x_{i_n} x_{i_{n-1}} ... x_{i_2} x_{i_1} \qquad \blacksquare$$

Definizione 5.7 (Operazione di riflessione)

Sia w una parola su un alfabeto $X = \{x_1, x_2, ..., x_k\}$ e sia w^R la stringa riflessa di w. L'operazione che trasforma w in w^R è detta *operazione di riflessione*. $\blacksquare$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Definizione 5.8 (Parola palindromica)

Un *palindromo* (o *parola palindromica*) è una parola la cui lettura a ritroso riproduce la parola di partenza:

$$w \text{ palindromo } \overset{def}{\Leftrightarrow} w = w^R$$

Un palindromo è dunque una parola che coincide con la sua riflessione. ■

Esempio 5.7

Alcuni palindromi sull'alfabeto $\{a, b, ..., z\}$ sono:

- *a*;
- *ii* (plurale di io?);
- *non, ala, ara, ici*;
- *osso, alla, arra*;
- *radar, alalà, arerà* (ignorando l'accento);
- *ossesso, ingegni*;
- *avallava, ovattavo*;
- *onorarono*;
- *accavallavacca*, *accumolomucca*;
- *feci nulla all'Unicef*, *ogni tela male tingo* (ignorando spazi bianchi, punteggiatura e differenza tra maiuscole e minuscole).

I palindromi (su un qualunque alfabeto) sono di due tipi:

- palindromi di lunghezza pari;
- palindromi di lunghezza dispari.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

I primi hanno un "*asse di simmetria*" costituito dalla parola vuota; i secondi hanno un "*asse di simmetria*" costituito da uno dei simboli dell'alfabeto.

Più precisamente, si ha la seguente caratterizzazione (senza dimostrazione):

Teorema 5.5

Sia w una parola su un alfabeto X.

w è palindromo se e solo se $w = \alpha x \alpha^R,\ x \in X \cup \{\lambda\}$.

Teorema 5.6

La classe dei linguaggi non contestuali (tipo '2') è chiusa rispetto all'operazione di riflessione.

Dimostrazione 5.6

Sia $G_1 = (X, V_1, S_1, P_1)$ una grammatica non contestuale. Dobbiamo dimostrare che:

$$L(G_1)\ \text{non contestuale} \Rightarrow \left(L(G_1)\right)^R = \{w^R \mid L(G_1)\}\ \text{è non contestuale.}$$

Costruiamo la grammatica:

$$G_{11} = (X, V_1, S_1, P_{11})$$

ove:

$$P_{11} = \{A \to w^R \mid A \to w \in P_1\}$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Risulta allora:

$$L(G_{11}) = \left(L(G_1)\right)^R$$

Dimostrare per esercizio.

Quindi se in $P_{\underline{1}}$ abbiamo la produzione:

$$A \to BaC$$

in $P_{\underline{11}}$ avremo la produzione:

$$A \to CaB \qquad\qquad c.v.d.$$

5.7 Esercizi

<u>Esercizio 5.1</u>

Dimostrare che il linguaggio:

$$L = \{a^n b^n c^m \mid n, m > 0\}$$

è non contestuale.

Consideriamo i linguaggi:

$L_1 = \{a^n b^n \mid n > 0\}$ $\qquad$ $L_2 = \{c^m \mid m > 0\} = \{c\}^+ = \{c\}^* - \{\lambda\}$

Si ha:

$$L = L_1 \cdot L_2$$

L_1 è un linguaggio non contestuale ed L_2 è un linguaggio lineare destro. Si ha infatti:

$$L_2 = \{c\}^* - \{\lambda\} = \{c\}^* \cap \overline{\{\lambda\}}$$

e $\{c\}^*$ è lineare destro (per esercizio determinare la grammatica che genera $\{c\}^*$).

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Poiché $\{\lambda\}$ è lineare destro, per la chiusura dei linguaggi di tipo '3' rispetto al complemento si ha che:

$$\overline{\{\lambda\}}$$

è lineare destro. Dunque L_2 è lineare destro e

$$L = L_1 \cdot L_2$$

è non contestuale, poiché si ottiene per concatenazione di due linguaggi non contestuali ($L_2 \in \mathcal{L}_3 \subset \mathcal{L}_2$).

La grammatica G_1 che genera L_1 è data da:

$$G_1 = (X_1, V_1, S_1, P_1)$$

$$X_1 = \{a,b\} \qquad V_1 = \{S_1\} \qquad P_1 = \left\{S_1 \underset{(1)}{\rightarrow} aS_1b,\ S_1 \underset{(2)}{\rightarrow} ab\right\}$$

La grammatica G_2 che genera L_2 è data da:

$$G_2 = (X_2, V_2, S_2, P_2)$$

$$X_2 = \{c\} \qquad V_2 = \{C\} \qquad S_2 = C \qquad P_2 = \left\{C \underset{(1)}{\rightarrow} cC,\ C \underset{(2)}{\rightarrow} c\right\}$$

La grammatica G che genera $L = L_1 \cdot L_2$ è data da:

$$G = (X, V, S, P)$$

$$X = X_1 \cup X_2 = \{a,b,c\} \qquad V = V_1 \cup V_2 \cup \{S\} = \{S, S_1, C\}$$

$$P = \{S \rightarrow S_1C\} \cup P_1 \cup P_2 = \left\{\begin{array}{l} S \underset{(1)}{\rightarrow} S_1C,\ S_1 \underset{(2)}{\rightarrow} aS_1b,\ S_1 \underset{(3)}{\rightarrow} ab, \\ C \underset{(4)}{\rightarrow} cC,\ C \underset{(5)}{\rightarrow} c \end{array}\right\}$$

G è non contestuale.

<u>Esercizio 5.2</u>

Siano $L_1 = \{a^n b^n \mid n \geq 0\}$ ed $L_2 = \{a\}^* \cdot \{bb\}^*$. Dimostrare che il linguaggio $L = L_1 \cap L_2$ è non contestuale.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Il linguaggio L_1 può essere riguardato come l'unione di due insiemi:

$$L_1 = \{a^n b^n \mid n > 0\} \cup \{\lambda\}$$

e tale linguaggio è non contestuale.

La grammatica che lo genera è:

$$G_1 = (X_1, V_1, S_1, P_1)$$

$$X_1 = \{a, b\} \qquad V_1 = \{S_1\} \qquad P_1 = \left\{S_1 \underset{(1)}{\rightarrow} aS_1b,\ S_1 \underset{(2)}{\rightarrow} ab,\ S_1 \underset{(3)}{\rightarrow} \lambda\right\}$$

$L_2 = \{a\}^* \cdot \{bb\}^*$ è lineare destro. Infatti, il linguaggio $L_3 = \{a\}^*$ è lineare destro e la grammatica che lo genera è:

$$G_3 = (X_3, V_3, S_3, P_3)$$

$$X_3 = \{a\} \qquad V_3 = \{A\} \qquad S_3 = A$$

$$P_3 = \left\{A \underset{(1)}{\rightarrow} aA,\ A \underset{(2)}{\rightarrow} a,\ A \underset{(3)}{\rightarrow} \lambda\right\}.$$

Ma anche il linguaggio $L_4 = \{bb\}^*$ è lineare destro e la grammatica che lo genera è:

$$G_4 = (X_4, V_4, S_4, P_4)$$

$$X_4 = \{b\} \qquad V_4 = \{B, B_1\} \qquad S_4 = B$$

$$P_4 = \left\{B \underset{(1)}{\rightarrow} bB_1,\ B \underset{(2)}{\rightarrow} \lambda,\ B_1 \underset{(3)}{\rightarrow} bB,\ B_1 \underset{(4)}{\rightarrow} b\right\}.$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Dunque la grammatica che genera $L_2 = L_3 \cdot L_4$ è:

$$G_2 = (X_2, V_2, S_2, P_2)$$

$$X_2 = X_3 \cup X_4 = \{a,b\} \qquad V_2 = V_3 \cup V_4 = \{A, B, B_1\} \qquad S_2 = S_3 = A$$

$$\begin{aligned} P_2 = {} & \{A \to aA\} \cup \{A \to aB\} \cup \\ & \cup \{B \to bB_1,\ B \to \lambda,\ B_1 \to bB,\ B_1 \to b\} \cup \\ & \cup \{A \to bB_1,\ A \to \lambda\} = \\ = & \left\{ \begin{array}{l} A \underset{(1)}{\to} aA,\ A \underset{(2)}{\to} aB,\ A \underset{(3)}{\to} bB_1,\ A \underset{(4)}{\to} \lambda, \\ B \underset{(5)}{\to} bB_1,\ B \underset{(6)}{\to} \lambda,\ B_1 \underset{(7)}{\to} bB,\ B_1 \underset{(8)}{\to} b \end{array} \right\} \end{aligned}$$

$L = L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$, per la legge di De Morgan. Non possiamo però dire nulla sulla classe di linguaggi cui L appartiene, dato che L_1 è non contestuale e la classe dei linguaggi non contestuali non è chiusa rispetto al complemento.

Procediamo in modo diverso.

Il linguaggio $L = L_1 \cap L_2$ non è altro che:

$$L = L_1 \cap L_2 = \left\{ w \,\middle|\, w \in \{a,b\}^*, \quad \begin{array}{l} w = a^n b^n,\ n \geq 0 \ \text{ AND} \\ w = a^k (bb)^m,\ k, m \geq 0 \end{array} \right\}$$

Poiché necessariamente si deve avere $k = n$, si ha:

$$\begin{aligned} L &= \left\{ w \,\middle|\, w \in \{a,b\}^*,\ w = a^n b^n,\ n \geq 0 \ \text{ AND } \ w = a^n (bb)^m,\ m \geq 0 \right\} = \\ &= \left\{ w \,\middle|\, w \in \{a,b\}^*,\ w = a^n b^n,\ n \geq 0 \ \text{ AND } \ w = a^n b^{2m},\ m \geq 0 \right\} \end{aligned}$$

ma, poiché si deve avere che $n = 2m$, si ha:

$$L = \left\{ w \,\middle|\, w \in \{a,b\}^*,\ w = a^{2m} b^{2m},\ m \geq 0 \right\} = \left\{ a^{2m} b^{2m} \,\middle|\, m \geq 0 \right\}$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

L è un linguaggio non contestuale e la grammatica che genera L è:

$$G = (X, V, S, P)$$

ove:

$$X = \{a, b\} \qquad V = \{S\} \qquad P = \left\{S \underset{(1)}{\rightarrow} aaSbb,\ S \underset{(2)}{\rightarrow} aabb,\ S \underset{(3)}{\rightarrow} \lambda\right\}.$$

<u>Esercizio 5.3</u>

Si utilizzi la proprietà di chiusura della classe $\mathcal{L}_2$ rispetto all'unione per dimostrare che ciascuno dei seguenti linguaggi è non contestuale:

(a) $L = \{a^i b^j \mid i \neq j,\ i, j \geq 0\}$
(b) $L = \left\{ w \in \{a,b\}^* \mid w = w^R \right\}$
(c) $L = \{a,b\}^* - \{a^i b^i \mid i \geq 0\}$

(a) Analizziamo le parole del linguaggio:

$$L = \{a^i b^j \mid i \neq j,\ i, j \geq 0\} =$$

$$= \left\{ \begin{array}{l} b, b^2, \ldots, b^n, \ldots, a, ab^2, \ldots, ab^n, \\ a^2, a^2b, a^2b^3, \ldots, a^2b^n, \ldots, a^3b, a^3b^2, a^3b^4, \ldots, a^3b^n, \ldots \end{array} \right\} =$$

$$= \{a^n b^m \mid 0 \leq n < m\} \cup \{a^m b^n \mid 0 \leq n < m\}$$

Poniamo:

$$L_1 = \{a^n b^m \mid 0 \leq n < m\}$$
$$L_2 = \{a^m b^n \mid 0 \leq n < m\}$$

Si ha:

$$L = L_1 \cup L_2 .$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

L_1 è generato da:

$$G_1 = (X_1, V_1, S_1, P_1)$$

ove:

$$X_1 = \{a,b\} \qquad V_1 = \{S_1, B\}$$
$$P_1 = \left\{S_1 \underset{(1)}{\rightarrow} aS_1b,\ S_1 \underset{(2)}{\rightarrow} Bb,\ B \underset{(3)}{\rightarrow} Bb,\ B \underset{(4)}{\rightarrow} \lambda\right\}.$$

Analogamente, L_2 è generato da:

$$G_2 = (X_2, V_2, S_2, P_2)$$

ove:

$$X_2 = \{a,b\} \qquad V_2 = \{S_2, A\}$$
$$P_2 = \left\{S_2 \underset{(1)}{\rightarrow} aS_2b,\ S_2 \underset{(2)}{\rightarrow} aA,\ A \underset{(3)}{\rightarrow} aA,\ A \underset{(4)}{\rightarrow} \lambda\right\}.$$

L_1 ed L_2 sono non contestuali perché generati da grammatiche non contestuali.

Poiché la classe dei linguaggi non contestuali è chiusa rispetto all'unione, anche:

$$L = L_1 \cup L_2$$

è non contestuale. La grammatica che genera L è:

$$G = (X, V, S, P)$$

ove:

$$X = \{a,b\} \qquad V = V_1 \cup V_2 \cup \{S\} = \{S, S_1, S_2, A, B\}$$
$$\begin{aligned} P &= \{S \rightarrow S_1,\ S \rightarrow S_2\} \cup P_1 \cup P_2 = \\ &= \{S \rightarrow S_1,\ S \rightarrow S_2, \\ &\quad S_1 \rightarrow aS_1b \mid Bb,\ B \rightarrow Bb \mid \lambda, \\ &\quad S_2 \rightarrow aS_2b \mid aA,\ A \rightarrow aA \mid \lambda\}. \end{aligned}$$

Si provi a svolgere lo stesso esercizio *escludendo* la possibilità che i e j assumano valore zero.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

(b) $L = \left\{ w \in \{a,b\}^* \mid w = w^R \right\}$

Analizziamo le parole che costituiscono L:

$$L = \left\{ w \in \{a,b\}^* \mid w = w^R \right\} =$$
$$= \{\lambda, a, b, aa, bb, aaa, aba, bab, bbb, aaaa, abba, baab, bbbb, ...\}$$

L è l'insieme dei palindromi sull'alfabeto $\{a,b\}$.

Dunque, per il teorema di caratterizzazione, si ha che:

$$L = \left\{ \alpha x \alpha^R \mid \alpha \in \{a,b\}^*,\ x \in \{a,b,\lambda\} \right\} =$$
$$= \left\{ \alpha\alpha^R \mid \alpha \in \{a,b\}^* \right\} \cup \left\{ \alpha a \alpha^R \mid \alpha \in \{a,b\}^* \right\} \cup \left\{ \alpha b \alpha^R \mid \alpha \in \{a,b\}^* \right\}$$

Poniamo:

$$L_1 = \left\{ \alpha\alpha^R \mid \alpha \in \{a,b\}^* \right\}$$
$$L_2 = \left\{ \alpha a \alpha^R \mid \alpha \in \{a,b\}^* \right\}$$
$$L_3 = \left\{ \alpha b \alpha^R \mid \alpha \in \{a,b\}^* \right\}$$

La grammatica che genera L_1 è:

$$G_1 = (X, V_1, S_1, P_1)$$

ove:

$$X = \{a,b\} \qquad V_1 = \{S_1\} \qquad P_1 = \{S_1 \to aS_1a \mid bS_1b \mid \lambda\}.$$

La grammatica che genera L_2 è:

$$G_2 = (X, V_2, S_2, P_2)$$

ove:

$$X = \{a,b\} \qquad V_2 = \{S_2\} \qquad P_2 = \{S_2 \to aS_2a \mid bS_2b \mid a\}.$$

La grammatica che genera L_3 è:

$$G_3 = (X, V_3, S_3, P_3)$$

ove:

$$X = \{a,b\} \qquad V_3 = \{S_3\} \qquad P_3 = \{S_3 \to aS_3a \mid bS_3b \mid b\}.$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

L_1, L_2 ed L_3 sono linguaggi non contestuali perché G_1, G_2 e G_3 sono grammatiche non contestuali. $L = L_1 \cup L_2 \cup L_3$ è un linguaggio non contestuale dato che la classe dei linguaggi di tipo '2' è chiusa rispetto all'unione.

La grammatica che genera L è:

$$G = (X, V, S, P)$$

ove:

$$X = \{a, b\} \qquad V = V_1 \cup V_2 \cup V_3 \cup \{S\} = \{S, S_1, S_2, S_3\}$$

$$P = \{S \to S_1,\ S \to S_2,\ S \to S_3\} \cup P_1 \cup P_2 \cup P_3 =$$

$$= \{S \to S_1 \mid S_2 \mid S_3,\ S_1 \to aS_1a \mid bS_1b \mid \lambda,$$

$$S_2 \to aS_2a \mid bS_2b \mid a,\ S_3 \to aS_3a \mid bS_3b \mid b\}.$$

(c) Risolvere per esercizio.

<u>Esercizio 5.4</u>

Dimostrare che il linguaggio:

$$L = \{a^n b^m \mid n \neq m,\ n, m > 0\}$$

non è lineare destro, utilizzando le proprietà di chiusura di questa classe di linguaggi.

Se L fosse lineare destro, allora, per le proprietà di chiusura dei linguaggi lineari destri, anche il linguaggio:

$$L_1 = \{a^n b^n \mid n > 0\}$$

sarebbe un linguaggio lineare destro. Infatti, si ha:

$$L_1 = L_2 - L$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

ove $L_2 = \{a^n b^m \mid n, m > 0\}$. L_2 è un linguaggio lineare destro. Inoltre abbiamo che:

$$L_1 = L_2 - L = L_2 \cap \overline{L}$$

Se L fosse lineare destro, $\overline{L}$ sarebbe lineare destro (poiché $\mathcal{L}_3$ è chiusa rispetto al complemento).

Di conseguenza, anche L_1 sarebbe lineare destro in quanto L_2 è lineare destro ed $\mathcal{L}_3$ è chiusa anche rispetto all'operazione di intersezione. Ma L_1 è un linguaggio libero da contesto e non è lineare destro (si veda la dimostrazione del Teorema 5.1). Dunque L non è lineare destro.

Riassumiamo lo schema di ragionamento seguito nel precedente esercizio in Tavola 2.

SCHEMA DI RAGIONAMENTO PER UTILIZZO PROPRIETÀ DI CHIUSURA	
Siano L, L_1 ed L_2 tre linguaggi tali che: $L = \alpha(L_1, L_2)$ ove $\alpha = \cup, \cdot$	
ESATTO	***ERRATO***
Supponiamo che $L_2 \in \mathcal{L}_i$. Se $L \notin \mathcal{L}_i$ allora $L_1 \notin \mathcal{L}_i$.	Se $\underset{j=1,2}{L_j} \notin \mathcal{L}_i$ allora $L \notin \mathcal{L}_i$

Tavola 2

titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

5.8 Esercizi proposti

Esercizio 1

Stabilire se i seguenti linguaggio sono liberi da contesto e giustificare formalmente la risposta:

a) $\{a^i b^k c^j : k = i + j, i, j, k > 0\}$

b) $\{wb^n w' : w, w' \in \{ a, c \}^*, |ww'| = n, n > 0\}$

c) $\{wb^n : w \in \{a, c\}^*, |w| = n, n \geq 0 \} \cdot \{ b^n w : w \in \{a, c\}^*, |w| = n, n \geq 0\}$

6. Automi a stati finiti (deterministici e non deterministici)

6.1 Automi a stati finiti deterministici

Definizione 6.1 (Automa a stati finiti o accettore a stati finiti o FSA)

Sia X un alfabeto. Un *automa a stati finiti* (*FSA*) è una quadrupla:

$$M = (Q, \delta, q_0, F)$$

ove:

- X è detto *alfabeto di ingresso*;
- Q è un insieme finito e non vuoto di *stati*;
- δ è una funzione da $Q \times X$ in Q, detta *funzione di transizione*:

$$\delta : Q \times X \to Q$$

- q_0 è lo *stato iniziale*;
- $F \subseteq Q$ è l'insieme degli *stati di accettazione o finali*. ■

Talora i valori della funzione di transizione δ non sono definiti per tutte le coppie (stato-simbolo di ingresso) (q, x). In tal caso, si dice che δ è una funzione parziale o definita parzialmente. Questo significa che la lettura di x dà luogo in q ad un comportamento dell'automa che non si ritiene utile descrivere ai fini del riconoscimento (nel senso che produrrebbe stringhe non accettate).

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Evidentemente questo fatto può essere descritto in modo equivalente, seguendo la definizione data di automa a stati finiti, passando da q, per effetto di x, in uno stato q' dal quale non si possa mai raggiungere uno stato finale (*stato pozza*).

Un FSA può essere rappresentato mediante:

i) *Grafo degli Stati* o *Diagramma di Transizione* o *Diagramma di Stato*

È una rappresentazione grafica in cui:

- ogni stato $q \in Q$ è rappresentato da un cerchio (*nodo*) con etichetta q;
- lo stato iniziale (nodo q_0) ha un arco orientato entrante libero (ossia, che non proviene da nessun altro nodo);
- per ogni stato $q \in Q$ e per ogni simbolo x dell'alfabeto di ingresso, $x \in X$, se $\delta(q,x) = q'$, esiste un arco orientato etichettato con x uscente dal nodo q ed entrante nel nodo q' (Figura 6.1).

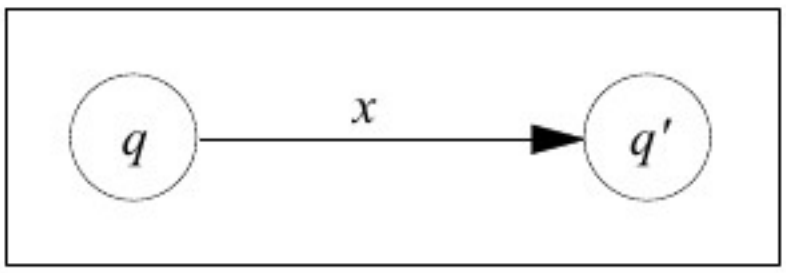


Figura 6.1

ii) *Tavola di Transizione*

È una tabella in cui sono riportati gli stati sulle righe e i simboli dell'alfabeto di ingresso sulle colonne.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Per ogni coppia (stato-ingresso) si legge nella tavola lo stato successivo:

δ	x_1	x_2	...	....	x_n
q_0	q_0^1	q_0^2	...	...	q_0^n
q_1	q_1^1	q_1^2	...	...	q_1^n
...	...	...	...	...	...
...	...	...	...	...	...
q_m	q_m^1	q_m^2	...	...	q_m^n

dove l'alfabeto di ingresso e l'insieme degli stati sono rispettivamente:

$X = \{x_1, x_2, ..., x_n\}$ e $Q = \{q_0, q_1, ..., q_m\}$. Quindi si ha che:

$$\delta(q_i, x_j) = q_i^j \text{ con } q_i, q_i^j \in Q,\ x_j \in X .$$

Si può definire un'estensione della funzione di transizione δ come segue:

<u>Definizione 6.2</u> (δ^* per FSA)

Dato un FSA $M = (Q, \delta, q_0, F)$ con alfabeto di ingresso X, definiamo per induzione la funzione:

$$\delta^* : Q \times X^* \to Q$$

tale che $\delta^*(q, w)$, per $q \in Q$ e $w \in X^*$, sia lo stato in cui M si porta avendo in ingresso la parola w su X e partendo dallo stato q.

$$\begin{cases} \delta^*(q, \lambda) = q \\ \delta^*(q, wx) = \delta\big(\delta^*(q, w), x\big) \end{cases} \qquad \text{per ogni } q \in Q,\ x \in X,\ w \in X^*$$

■

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Si veda la Figura 6.2 per una descrizione grafica della definizione precedente.

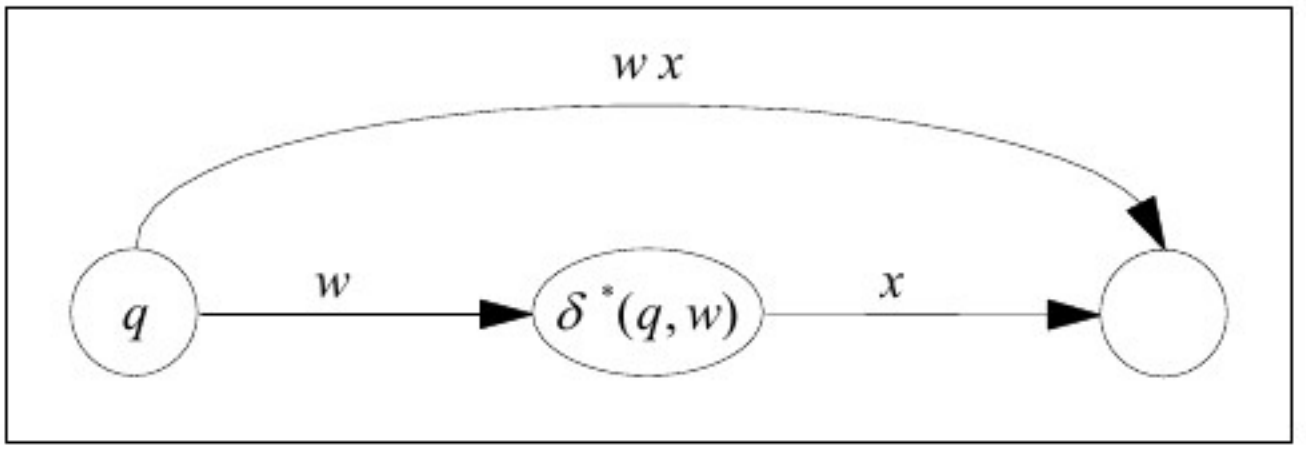


Figura 6.2

<u>Definizione 6.3</u> (Parola accettata o riconosciuta da un FSA)

Sia $M = (Q, \delta, q_0, F)$ un FSA con alfabeto di ingresso X.

Una *parola* $w \in X^*$ è *accettata* (o *riconosciuta*) da M se, partendo dallo stato iniziale q_0, lo stato q in cui l'automa si porta alla fine della sequenza di ingresso w è uno stato finale.

$$w \text{ accettata } \overset{def}{\Leftrightarrow} \delta^*(q_0, w) \in F$$

■

<u>Definizione 6.4</u> (Linguaggio accettato o riconosciuto da un FSA)

Sia $M = (Q, \delta, q_0, F)$ un FSA con alfabeto di ingresso X.

Il *linguaggio accettato* o *riconosciuto* da M è il seguente sottoinsieme di X^*:

$$T(M) = \{w \in X^* \mid \delta^*(q_0, w) \in F\}$$

(l'insieme delle parole accettate da M).

■

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Definizione 6.5 (FSA equivalenti)

Sia $M_1 = (Q_1, \delta_1, q_1, F_1)$ ed $M_2 = (Q_2, \delta_2, q_2, F_2)$ due FSA di alfabeto di ingresso X. M_1 ed M_2 si dicono *equivalenti* se:

$$T(M_1) = T(M_2)$$ ■

6.2 Linguaggi a stati finiti

Definizione 6.6 (Linguaggio a stati finiti, classe dei linguaggi a stati finiti)

Dato un alfabeto X, un linguaggio L su X è un *linguaggio a stati finiti* (o FSL - Finite State Language) se esiste un FSA M con alfabeto di ingresso X tale che $L = T(M)$.

Risulta così definita la *Classe dei Linguaggi a Stati Finiti*:

$$\mathcal{L}_{FSL} = \left\{ L \in 2^{X^*} \mid \exists M,\ M \text{ è un FSA} : L = T(M) \right\}$$ ■

Proposizione 6.1

I linguaggi a stati finiti sono chiusi rispetto al complemento.

Dimostrazione 6.1

Sia $L \in \mathcal{L}_{FSL}$ un linguaggio a stati finiti sull'alfabeto X.

Dalla definizione di linguaggio a stati finiti, $L = T(M)$, ove $M = (Q, \delta, q_0, F)$.[8]

[8] M è un automa stati finiti con funzione di transizione δ **definita totalmente** sul proprio dominio.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Consideriamo il complemento di L: $\overline{L} = X^* - L$ e l'automa a stati finiti $\overline{M} = (Q, \delta, q_0, Q - F)$. Si ha: $\overline{L} = T(\overline{M})$.

(Per esercizio, dimostrare $\overline{L} = T(\overline{M})$ per induzione sulla lunghezza di una parola w). *c.v.d.*

6.3 Automi a stati finiti non deterministici

Definiamo ora una nuova classe di macchine a stati finiti.

<u>Definizione 6.7</u> (Automa a stati finiti non deterministico o accettore a stati finiti non deterministico)

Un *automa a stati finiti non deterministico* (*NDA*) con alfabeto di ingresso X è una quadrupla:

$$M = (Q, \delta, q_0, F)$$

ove:

- per Q, q_0 ed F valgono le definizioni date per gli FSA;
- $\delta : Q \times X \rightarrow 2^Q$ è la funzione di transizione che assegna ad ogni coppia (stato-simbolo di ingresso) (q, x) un insieme $\delta(q, x) \subseteq Q$ di possibili stati successivi. ■

Si può osservare che un NDA è un FSA con l'unica eccezione che, in corrispondenza di una coppia (stato-simbolo di ingresso) (q, x), vi è un insieme di stati in cui l'automa può transitare (*stati successivi possibili*).

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Inoltre, come per gli automi a stati finiti deterministici (FSA), possiamo estendere la funzione di transizione δ al dominio $2^Q \times X^*$ attraverso la funzione δ^* definita come segue:

<u>Definizione 6.8</u> (δ^* per NDA)

Dato un NDA $M = (Q, \delta, q_0, F)$ con alfabeto di ingresso X, definiamo per induzione la funzione:

$$\delta^* : 2^Q \times X^* \to 2^Q$$

$$\begin{cases} \delta^*(p,\lambda) = p \\ \delta^*(p,wx) = \bigcup_{q \in \delta^*(p,w)} \delta(q,x) \end{cases} \quad \text{per ogni } p \in 2^Q \ (p \subset Q),\ x \in X,\ w \in X^*$$

■

Analogamente a quanto fatto per gli FSA, si dovrebbero riformulare le definizioni di parola accettata e di linguaggio accettato da un NDA.

La complicazione, rinveniente dalla computazione non deterministica dello stato successivo in cui un NDA transita, comporta che una stessa parola può indurre cammini multipli attraverso un NDA, alcuni che terminano in stati di accettazione, altri che terminano in stati di non accettazione.

Esempio 6.1

Sia dato il seguente NDA $M = (Q, \delta, q_0, F)$ (Figura 6.3):

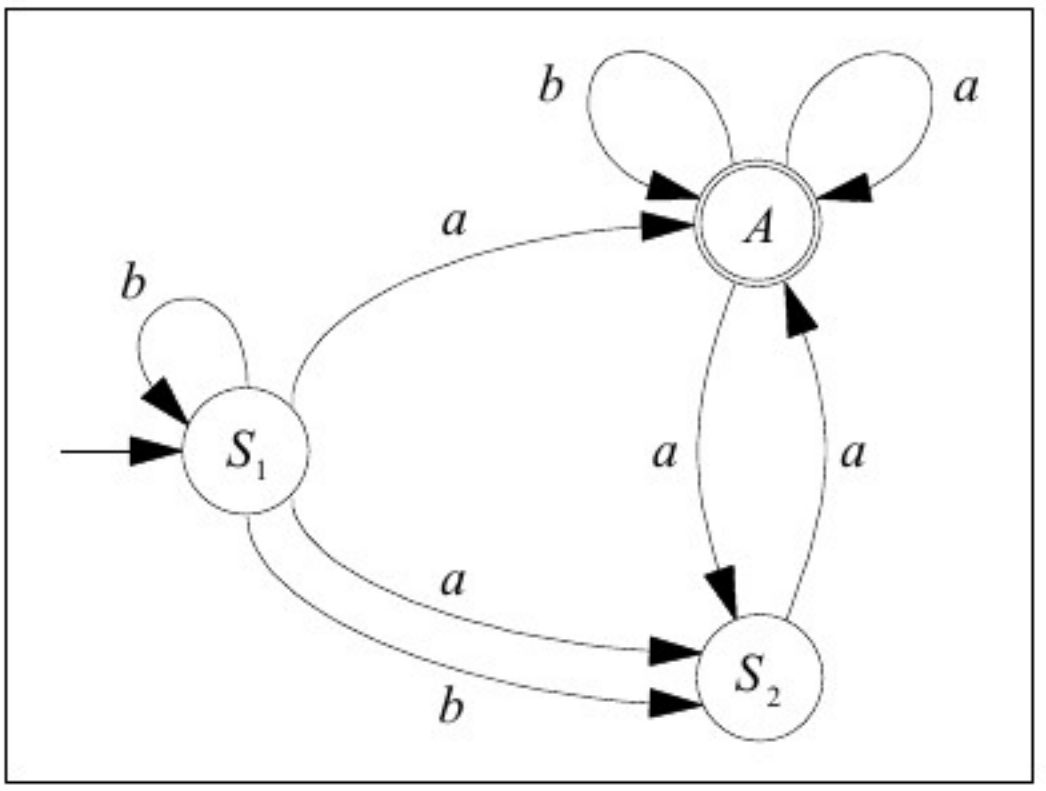


Figura 6.3

$$\delta(\{S_1\}, a) = \{A, S_2\} \qquad \delta(\{S_1\}, b) = \{S_1, S_2\}$$

... ...

... ...

Supponiamo di avere la parola $w = aba$. Vogliamo stabilire se w è riconosciuta dall'automa (si veda la Figura 6.4).

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Partiamo da S_1:

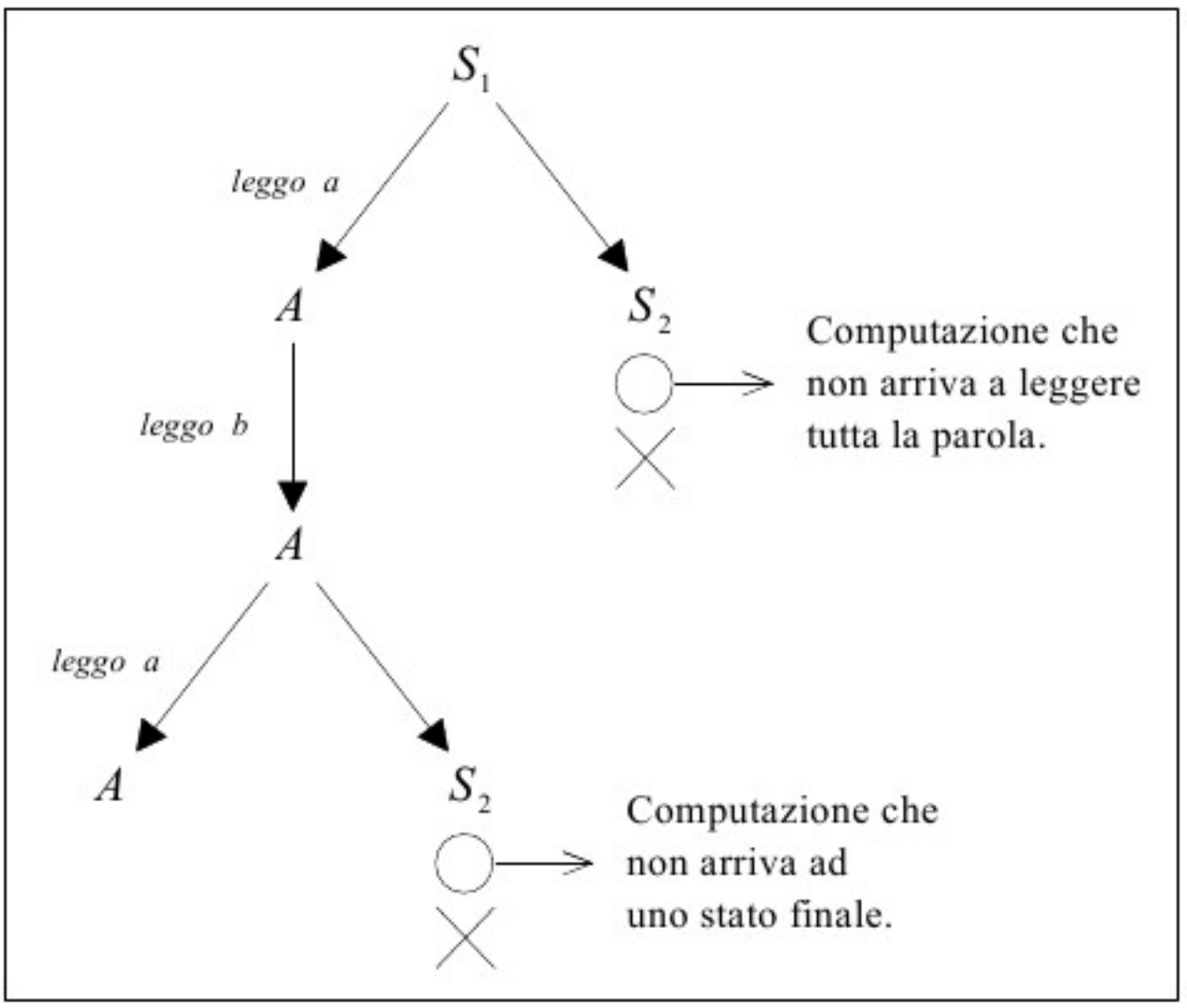


Figura 6.4

Quanto visto in Figura 6.4 è l'albero di tutte le possibili computazioni per la parola $w = aba$. Poiché almeno una computazione arriva ad uno stato finale, la stringa è riconosciuta[9].

Possiamo ora formulare le definizioni di parola accettata e di linguaggio accettato da un NDA.

[9] Può succedere che si abbiano computazioni che non leggono tutta la parola in ingresso (nel caso δ sia definita parzialmente) e computazioni che, pur leggendo per intero la parola in ingresso, non arrivano ad uno stato finale (non determinismo).

Definizione 6.9 (Parola accettata o riconosciuta da un NDA)

Sia $M = (Q, \delta, q_0, F)$ un NDA con alfabeto di ingresso X.

Una *parola* $w \in X^*$ è *accettata* (o *riconosciuta*) da M se, partendo dallo stato iniziale q_0, esiste almeno un modo per M di portarsi in uno stato di accettazione alla fine della sequenza di ingresso w. In formule:

$$w \text{ accettata } \overset{def}{\Leftrightarrow} \exists\, p : p \in \delta^*(\{q_0\}, w) \cap F \Leftrightarrow$$
$$\Leftrightarrow \delta^*(\{q_0\}, w) \cap F \neq \varnothing$$

■

Esempio 6.2

Sia dato l'NDA dell'esempio 6.1. Calcoliamo $\delta^*(\{S_1\}, aba)$.

$$\delta^*(\{S_1\}, aba) = \bigcup_{q \in \delta^*(\{S_1\}, ab)} \delta(q, a)$$

$$\delta^*(\{S_1\}, ab) = \bigcup_{q' \in \delta^*(\{S_1\}, a)} \delta(q', b)$$

$$\delta^*(\{S_1\}, a) = \bigcup_{q'' \in \delta^*(\{S_1\}, \lambda)} \delta(q'', a)$$

$$\delta^*(\{S_1\}, \lambda) = \{S_1\}$$

$$\delta^*(\{S_1\}, a) = \delta(S_1, a) = \{A, S_2\}$$

$$\delta^*(\{S_1\}, ab) = \delta(A, b) \cup \delta(S_2, b) = \{A\} \cup \varnothing = \{A\}$$

$$\delta^*(\{S_1\}, aba) = \delta(A, a) = \{A, S_2\}$$

Poiché $F = \{A\}$, si ha:

$$\delta^*(\{S_1\}, aba) \cap F = \{A\} \neq \varnothing$$

e $w = aba$ è accettata da M_1.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

6.4 Linguaggi accettati da automi non deterministici

Definizione 6.10 (Linguaggio accettato o riconosciuto da un NDA)

Sia $M = (Q, \delta, q_0, F)$ un NDA con alfabeto di ingresso X.

Il *linguaggio accettato* o *riconosciuto* da M è l'insieme delle parole su X accettate da M:

$$T(M) = \left\{ w \in X^* \mid \delta^*(\{q_0\}, w) \cap F \neq \varnothing \right\}$$

(è l'insieme delle parole w per le quali esiste almeno un cammino, etichettato con lettere di w nell'ordine da sinistra a destra, attraverso il diagramma degli stati che porta M dallo stato iniziale ad uno degli stati di accettazione). ■

Definizione 6.11 (NDA equivalenti)

Siano $M_1 = (Q_1, \delta_1, q_1, F_1)$ ed $M_2 = (Q_2, \delta_2, q_2, F_2)$ due NDA di alfabeto di ingresso X. M_1 ed M_2 si dicono *equivalenti* se:

$$T(M_1) = T(M_2)$$ ■

Risulta così definita, a partire da un NDA, la seguente classe di linguaggi:

$$\mathcal{L}_{NDL} = \left\{ L \in 2^{X^*} \mid \exists M,\ M \text{ è un NDA} : L = T(M) \right\}$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

6.5 Equivalenza delle classi di linguaggi accettati da automi a stati finiti deterministici e non deterministici

Teorema 6.2

Le classi dei linguaggi $\mathscr{L}_{FSL}$ e $\mathscr{L}_{NDL}$ sono equivalenti (1ª formulazione).

Sia L un linguaggio su X. L è un linguaggio a stati finiti se e solo se $L = T(M)$ per qualche NDA M (2ª formulazione).

Dimostrazione 6.2[10]

$\Rightarrow$) Sia $L \in 2^{X^*}$ ed $L \in \mathscr{L}_{FSL}$. Dalla definizione di linguaggio a stati finiti, si ha:

$$\exists M_1 : M_1 \text{ è un FSA, } M_1 = (Q_1, \delta_1, q_1, F_1)$$

con alfabeto di ingresso X : $L = T(M_1)$.

Sulla base di M_1, definiamo il seguente NDA con alfabeto di ingresso X:

$$M_2 = (Q_2, \delta_2, q_2, F_2)$$

ove:

- $Q_2 = Q_1$;
- δ_2 è così definita:

$$\delta_2 : Q_2 \times X \to 2^{Q_2},$$
$$\delta_2(q, x) = \{\delta_1(q, x)\} \quad \forall q \in Q_2 = Q_1,\ x \in X;$$

- $q_2 = q_1$;

[10] Fatta sulla base della 2ª formulazione.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

- $F_2 = F_1$.

M_2 è un NDA ed inoltre accetta lo stesso linguaggio accettato da M_1, ossia si ha:

$$T(M_2) = T(M_1)$$

(Dimostrarlo, per esercizio, per induzione sulla lunghezza di una parola w).

$\Leftarrow$) Sia $L \in 2^{X^*}$ ed $L \in \mathcal{L}_{NDL}$. Dalla definizione della classe di linguaggi $\mathcal{L}_{NDL}$, si ha:

$$\exists M,\ M \text{ è un NDA},\ M = (Q, \delta, q_0, F)$$

con alfabeto di ingresso X : $L = T(M)$.

Si può definire allora il seguente algoritmo per la costruzione di un FSA equivalente all'NDA M:

<u>Algoritmo 6.1</u> (Trasformazione di un automa a stati finiti non deterministico in un automa deterministico equivalente)

Sia $M = (Q, \delta, q_0, F)$ un automa accettore a stati finiti non deterministico di alfabeto di ingresso X. M può essere trasformato in un automa deterministico M' di alfabeto di ingresso X come segue:

$$M' = (Q', \delta', q_0', F')$$

ove:

i) $Q' = 2^Q$;

ii) $q_0' = \{q_0\}$;

iii) $F' = \{p \subseteq Q \mid p \cap F \neq \varnothing\}$;

iv) $\delta' : Q' \times X \to Q' \quad \ni' \quad \forall q' = \{q_1, q_2, ..., q_i\} \in Q',\ \forall x \in X$:

$$\delta'(q', x) = \delta'(\{q_1, q_2, ..., q_i\}, x) = \bigcup_{j=1}^{i} \delta(q_j, x) = \bigcup_{q \in q'} \delta(q, x).$$

Si può dimostrare che M' è equivalente a M, ossia che $T(M') = T(M)$. Ossia, dobbiamo dimostrare che:

$$\forall w \in X^* : w \in T(M') \Leftrightarrow w \in T(M)$$

Procediamo per induzione sulla lunghezza della parola w: $|w|$.

<u>Passo base</u>: $|w| = 0,\ w = \lambda$

$$\lambda \in T(M') \Rightarrow \delta'^*(q_0', \lambda) \in F'$$

Dobbiamo dimostrare che: $\delta^*(q_0, \lambda) \cap F \neq \varnothing$.

Calcoliamo:

$$\delta'^*(q_0', \lambda) = \delta'^*(\{q_0\}, \lambda) = \{q_0\} = \delta(q_0, \lambda) = \delta^*(q_0, \lambda)$$

Ma $\{q_0\} \in F' \overset{def}{\Leftrightarrow} \{q_0\} \cap F \neq \varnothing$.

<u>Passo induttivo</u>: $w' = wa$

L'ipotesi di induzione è che:

$$\delta'^*(\{q_0\}, w) = \delta^*(q_0, w)$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Poiché:

$$\delta'^*(q_0', wa) \overset{\text{def di } q_0'}{=} \delta'^*(\{q_0\}, wa) \overset{\text{def di } \delta'^*}{=} \delta'\big(\delta'^*(\{q_0\}, w), a\big) \overset{\substack{\text{per ipotesi}\\ \text{di}\\ \text{induzione}}}{=}$$

$$\overset{\substack{\text{per ipotesi}\\ \text{di}\\ \text{induzione}}}{=} \delta'\big(\delta^*(q_0, w), a\big) \overset{\text{def di } \delta'}{=} \bigcup_{q' \in \delta^*(q_0, w)} \delta(q', a) \overset{\text{def di } \delta^*}{=}$$

$$\overset{\text{def di } \delta^*}{=} \delta^*(q_0, wa)$$

cioè:

$$\delta^*(q_0, wa) = \delta'^*(q_0', wa)$$

e poiché per ipotesi:

$$\delta'^*(q_0', wa) \in F'$$

si ha:

$$\delta^*(q_0, wa) \cap F \neq \varnothing \qquad c.v.d.$$

6.6 Proprietà di chiusura della classe dei linguaggi a stati finiti

Per i linguaggi a stati finiti oltre alla proposizione 6.1 vale il seguente teorema:

<u>Teorema 6.3</u>

La classe dei linguaggi a stati finiti è chiusa rispetto alle operazioni di *unione, concatenazione* ed *iterazione*.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Dimostrazione 6.3

La dimostrazione di questo teorema è simile a quella riportata per $\mathcal{L}_3$.

Siano L_1 ed L_2 due linguaggi $\in \mathcal{L}_{FSL}$ allora esistono M_1 e M_2 tali che:

$$T(M_1) = L_1 \qquad M_1 = (Q_1, q_0^1, \delta_1, F_1)$$

$$\text{con} \quad M_2 = (Q_2, q_0^2, \delta_2, F_2)$$

$$T(M_2) = L_2$$

Si assume che $X_1 = X_2 = X$[11] e $Q_1 \cap Q_2 = \varnothing$.

Si vuole dimostrare che anche $L_1 \cup L_2$, $L_1 \cdot L_2$ e L_1^* sono linguaggi a stati finiti: la dimostrazione è costruttiva in quando consente di determinare per ognuna delle tre operazioni un automa a stati finiti $M = (Q, \delta, q_0, F)$ per cui $T(M) = T(M_1) \cup T(M_2)$, $T(M) = T(M_1) \cdot T(M_2)$ e $T(M) = (T(M_1))^*$.

UNIONE

Costruiamo M nel seguente modo:

$$Q = Q_1 \cup Q_2 \cup q_0$$

$$F = F_1 \cup F_2 \cup \{q_0 \mid \text{se } \lambda \in T(M_1) \cup T(M_2)\}$$

$$\delta(q,x) = \begin{cases} \delta_1(q,x) & \text{se } q \in Q_1 \\ \delta_2(q,x) & \text{se } q \in Q_2 \end{cases} \quad (\text{o anche } \delta(q,x) = \delta_1(q,x) \cup \delta_2(q,x))$$

$$\delta(q_0,x) = \delta_1(q_0^1,x) \cup \delta_2(q_0^2,x)$$

---

[11] Questa assunzione non limitativa in quanto è sufficiente considerare $X = X_1 \cup X_2$.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Esempio 6.3

Consideriamo i seguenti automi M_1 (Figura 6.5) e M_2 (Figura 6.6):

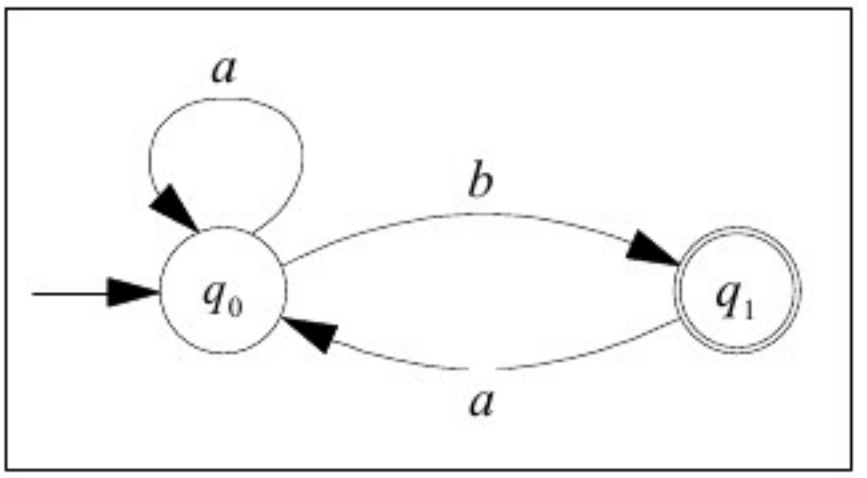


Figura 6.5

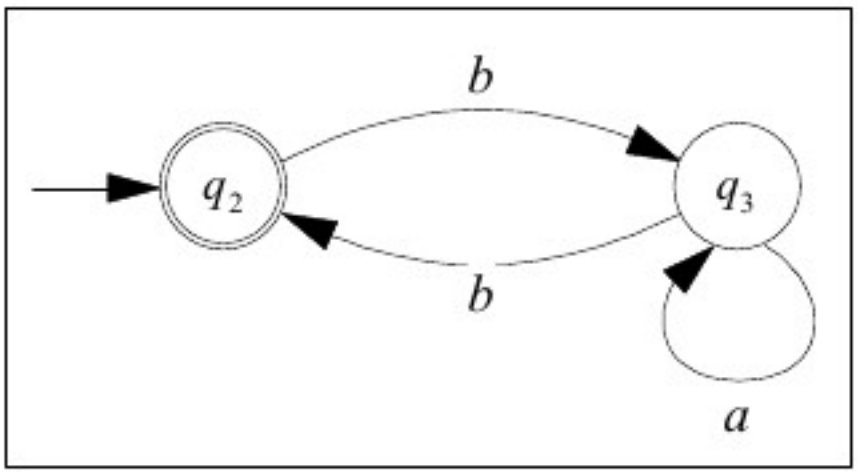


Figura 6.6

E' immediato verificare che M (Figura 6.7) è tale che $T(M) = T(M_1) \cup T(M_2)$.

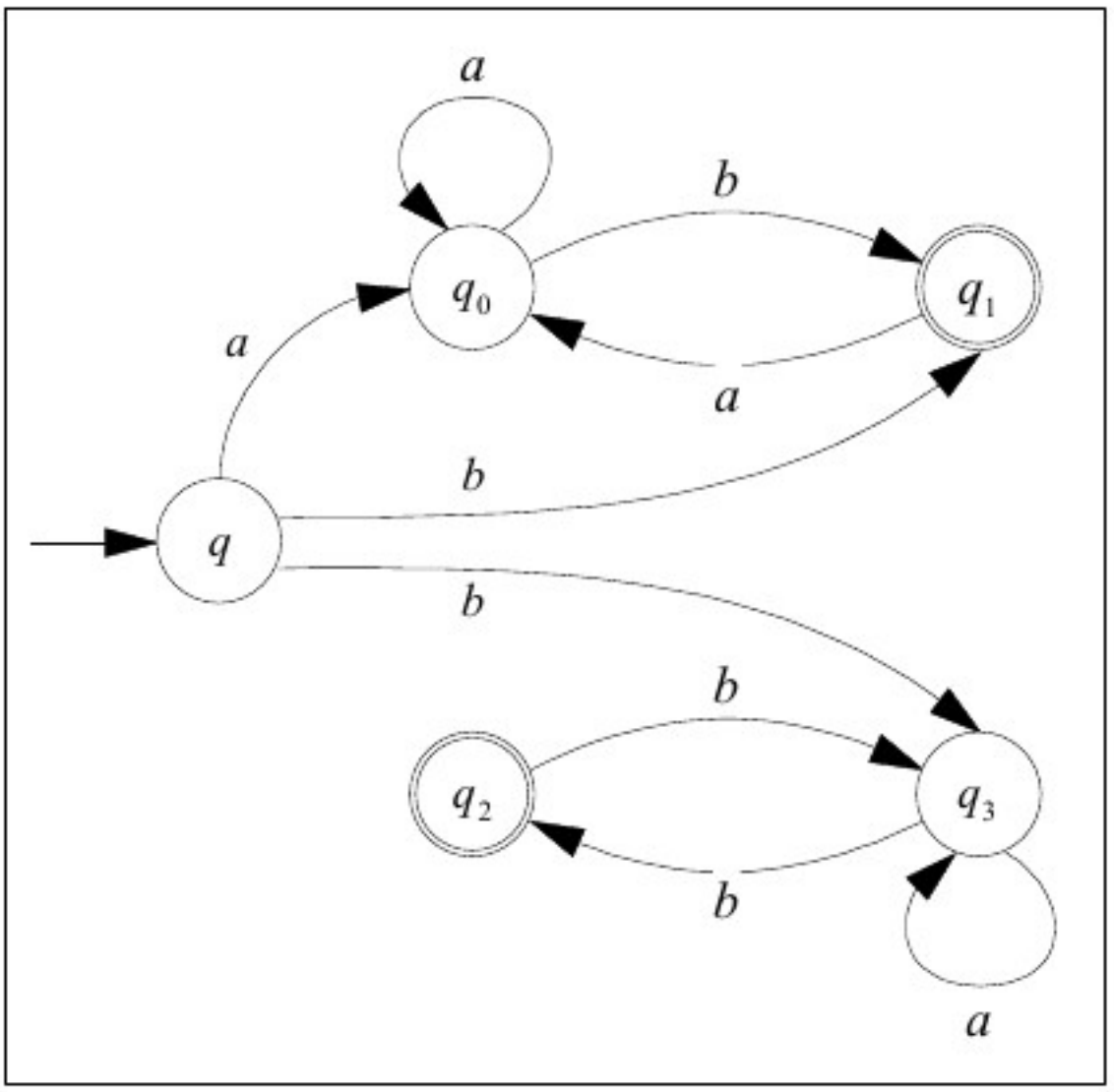


Figura 6.7

CONCATENAZIONE

Costruiamo M nel seguente modo:

$$Q = Q_1 \cup Q_2$$

$$q_0 = q_0^1$$

$$F = \begin{cases} F_2 & \text{se } \lambda \notin T(M_2) \\ F_1 \cup F_2 & \text{altrimenti} \end{cases}$$

$$\delta(q,x) = \begin{cases} \delta_1(q,x) & \text{se } q \in Q_1 \setminus F_1 \\ \delta_1(q,x) \cup \delta_2(q,x) & \text{se } q \in F_1 \\ \delta_2(q,x) & \text{se } q \in Q_2 \end{cases}$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Esempio 6.4

Consideriamo gli automi M_1 (Figura 6.5) e M_2 (Figura 6.6), è immediato verificare che M (Figura 6.8) è tale che $T(M) = T(M_2) \cdot T(M_1)$.

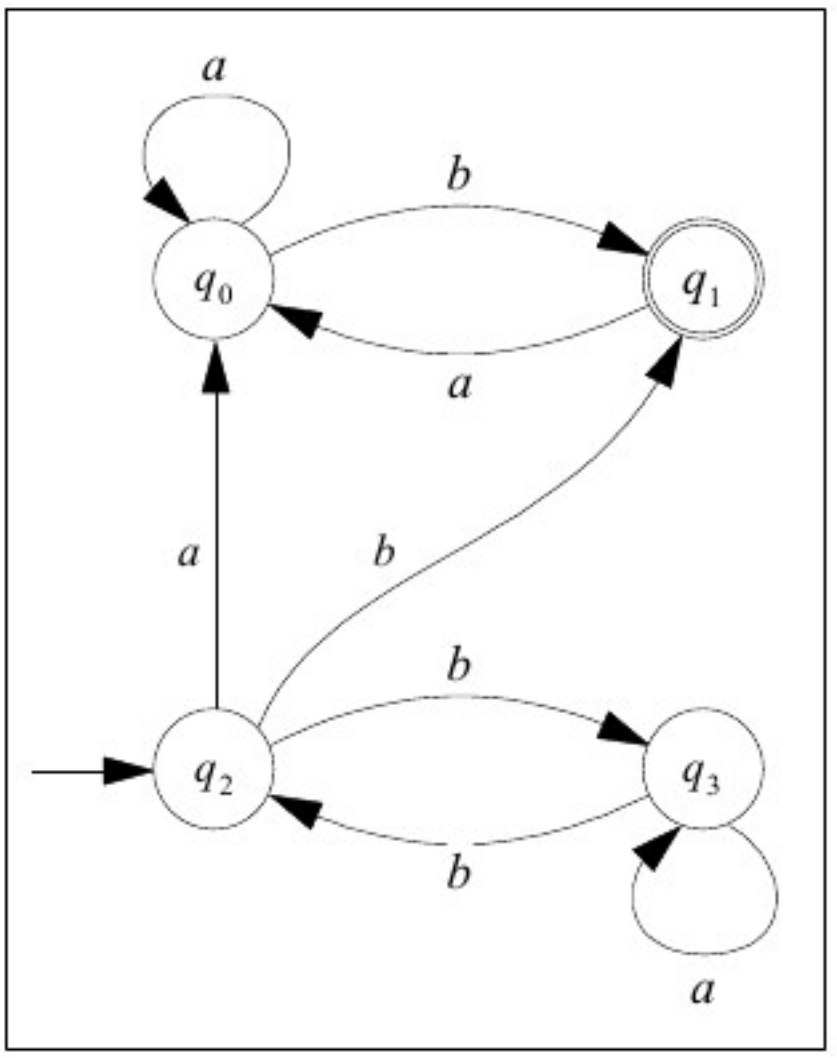


Figura 6.8

ITERAZIONE

Costruiamo M nel seguente modo:

$$Q = Q_1 \cup \{q_0\}$$

$$F = F_1 \cup \{q_0\}$$

$$\delta(q,x) = \begin{cases} \delta_1(q,x) & \text{se } q \in Q_1 \setminus F_1 \\ \delta_1(q_0^1,x) \cup \delta_1(q,x) & \text{se } q \in F_1 \end{cases}$$

$$\delta(q_0,x) = \delta(q_0^1,x)$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Esempio 6.5

Consideriamo l'automa M_1 (Figura 6.9):

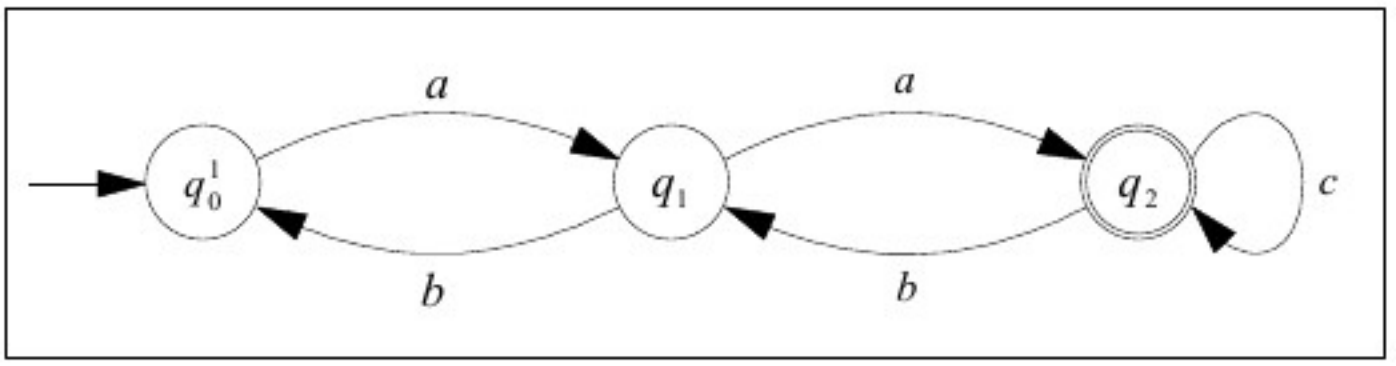


Figura 6.9

E' immediato verificare che M (Figura 6.10) è tale che $T(M) = (T(M_1))^*$.

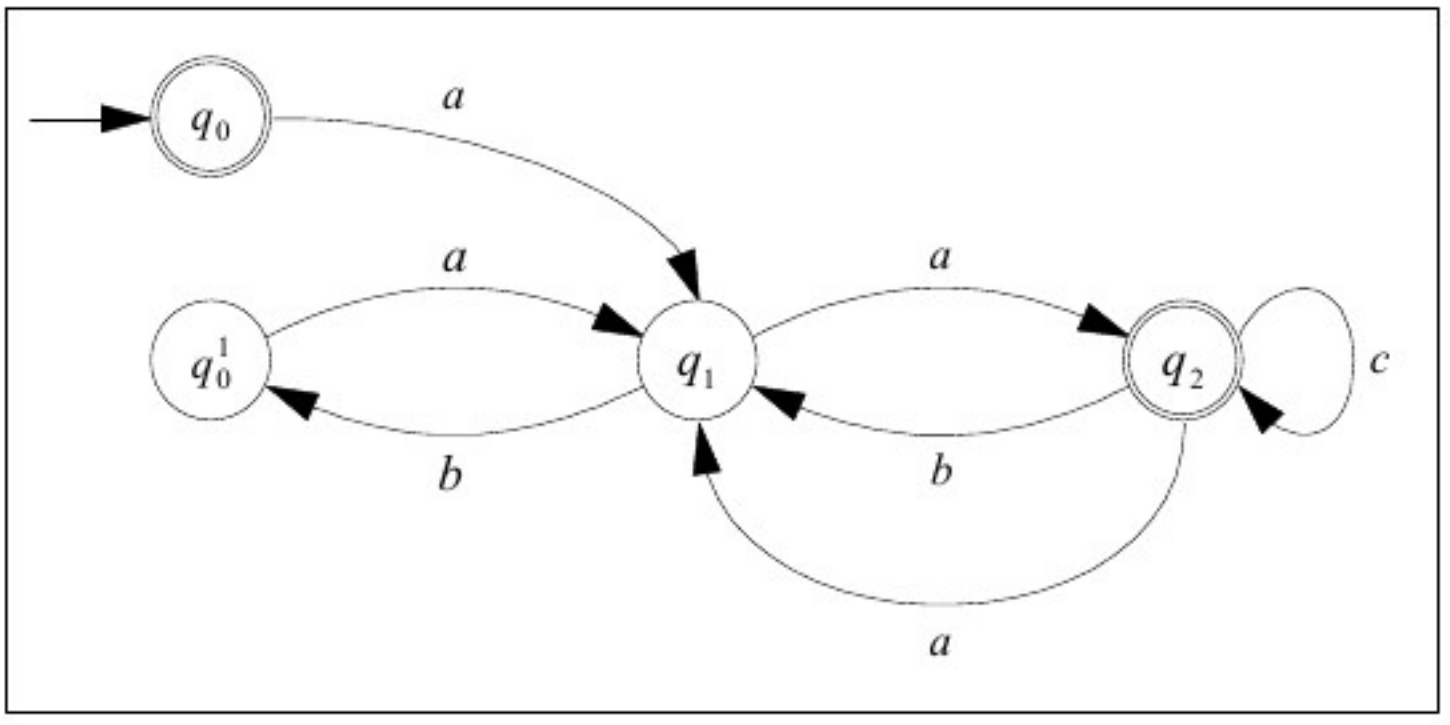


Figura 6.10

6.7 Esercizi

Esercizio 6.1

Sia dato il seguente linguaggio:

$$L = \left\{ w \,\middle|\, w \in \{a,b\}^*,\ w \text{ ha un numero pari di } a \text{ ed un numero dispari di } b \right\}$$

Costruire l'automa accettore a stati finiti che riconosce L.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Dobbiamo sintetizzare l'automa accettore M (Figura 6.11) tale che:

$$L = T(M)$$

$$M = (Q, \delta, q_0, F) \text{ con alfabeto } X = \{a,b\}$$

e con:

i) $Q = \{q_0, q_1, q_2, q_3\}$ dove:

- q_0 = numero pari di a e di b;
- q_1 = numero pari di a e numero dispari di b;
- q_2 = numero dispari di a e numero pari di b;
- q_3 = numero dispari di a e di b;

ii) la funzione di transizione δ è definita come segue:

- $\delta(q_0, a) = \delta(q_3, b) = q_2$;
- $\delta(q_0, b) = \delta(q_3, a) = q_1$;
- $\delta(q_1, a) = \delta(q_2, b) = q_3$;
- $\delta(q_1, b) = \delta(q_2, a) = q_0$;

iii) q_0 è lo stato iniziale;

iv) l'insieme degli stati finali o di accettazione è $F = \{q_1\}$.

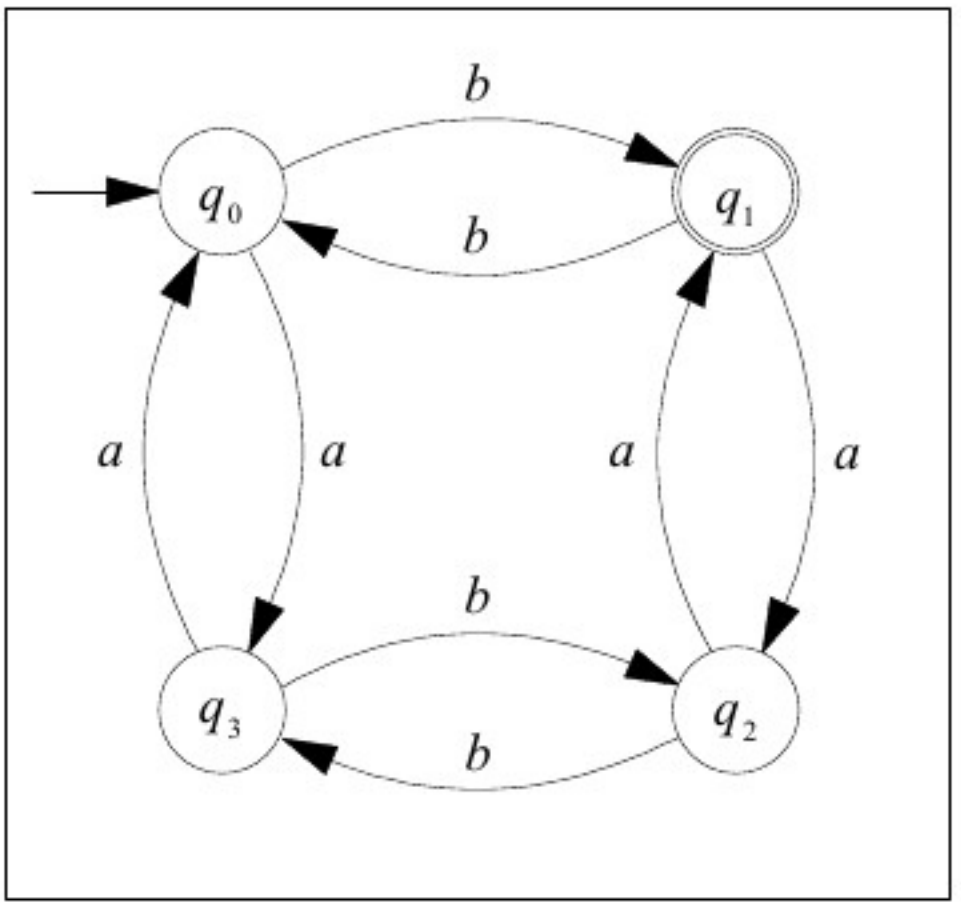


Figura 6.11

Esercizio 6.2

Sia dato il seguente linguaggio:

$$L = \left\{ w \;\middle|\; \begin{array}{c} w \in \{a,b\}^*,\ \ w \neq \alpha\, a\, a\, \beta,\ \ \alpha, \beta \in \{a,b\}^* \\ (w \text{ non contiene due } a \text{ consecutive}) \end{array} \right\}$$

Costruire l'automa accettore a stati finiti deterministico che riconosce L.

Dobbiamo sintetizzare l'automa accettore M tale che:

$$L = T(M)$$

$$M = (Q,\ \delta,\ q_0,\ F) \text{ con alfabeto } X = \{a,b\}$$

con:

i) $Q = \{q_0, q_1, q_2\}$ dove:

- q_0 = parole su $X = \{a,b\}$ che non contengono due o più a consecutive e terminano per b;

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

- q_1 = parole su $X = \{a,b\}$ che non contengono due o più a consecutive e terminano per a;
- q_2 = parole su $X = \{a,b\}$ che contengono due o più a consecutive (*stato pozza*);

ii) la funzione di transizione δ è definita come segue:

$$\delta : Q \times X \to Q$$

- $\delta(q_0, a) = q_1$;
- $\delta(q_0, b) = \delta(q_1, b) = q_0$;
- $\delta(q_1, a) = \delta(q_2, a) = \delta(q_2, b) = q_2$;

iii) q_0 è lo stato iniziale;

iv) l'insieme degli stati finali o di accettazione è $F = \{q_0, q_1\}$.

La tavola di transizione è dunque:

δ	q_0	q_1	q_2
a	q_1	q_2	q_2
b	q_0	q_0	q_2

Il grafo degli stati è indicato in Figura 6.12:

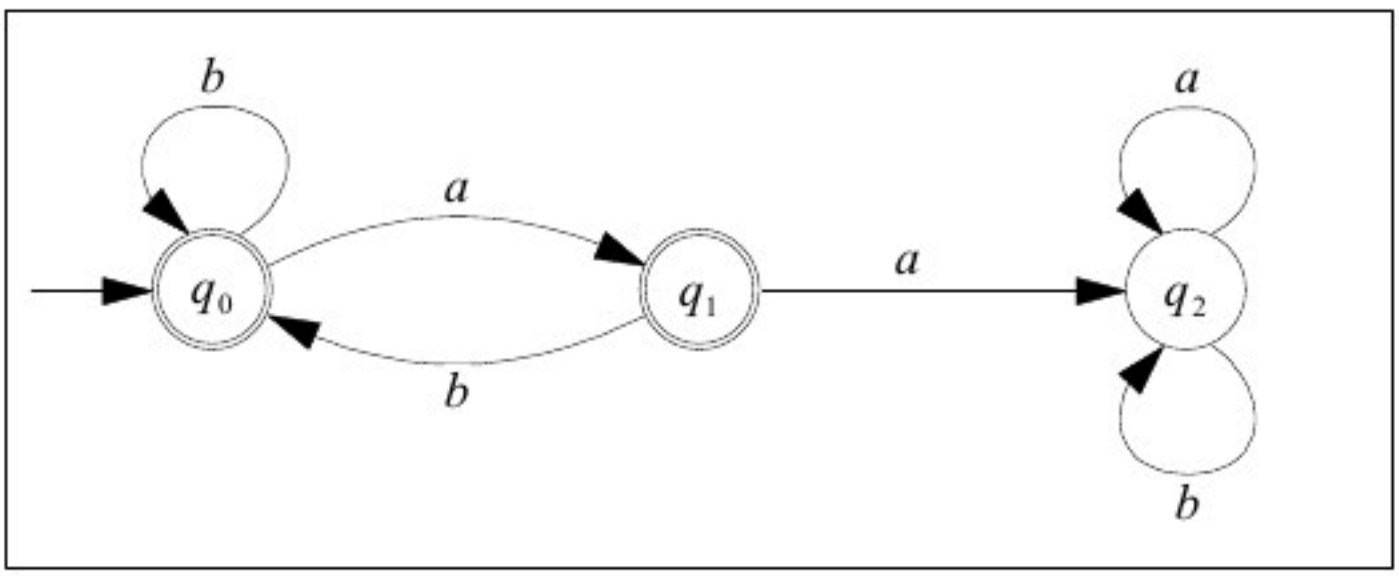


Figura 6.12

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

ed M è deterministico.

Esercizio 6.3

Trasformare il seguente automa non deterministico M (Figura 6.13):

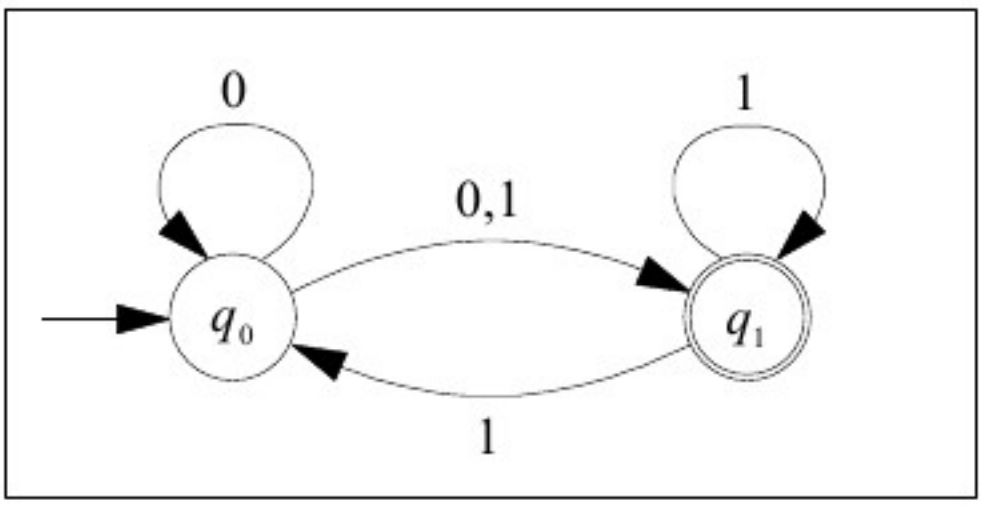


Figura 6.13

in un automa deterministico M' ad esso equivalente.

L'automa accettore a stati finiti deterministico M' tale che $T(M') = T(M)$ si costruisce come segue.

Dato l'automa di Figura 6.13:

$M = (Q, \delta, q_0, F)$ con $X = \{0,1\}$ come alfabeto di ingresso

ove:

1) $Q = \{q_0, q_1\}$;

2) $F = \{q_1\}$;

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

3) la funzione di transizione $\delta : Q \times X \to 2^Q$ è definita dalla seguente tavola di transizione:

δ	q_0	q_1
0	$\{q_0, q_1\}$	–
1	$\{q_1\}$	$\{q_0, q_1\}$

Definiamo M' come segue:

$M' = (Q', \delta', q_0', F')$ con $X = \{0,1\}$ come alfabeto di ingresso

ove:

i) $Q' = 2^Q = 2^{\{q_0, q_1\}}$;

ii) $q_0' = \{q_0\}$;

iii) $F' = \{\{q_1\}, \{q_0, q_1\}\}$;

iv) la funzione di transizione δ' è definita come segue:

$$\delta' : Q' \times X \to Q'$$

- $\delta'(\{q_0\},0) = \delta(q_0,0) = \{q_0, q_1\}$;
- $\delta'(\{q_0\},1) = \delta(q_0,1) = \{q_1\}$;
- $\delta'(\{q_1\},0) = \delta(q_1,0) =$ non è definita;
- $\delta'(\{q_1\},1) = \delta(q_1,1) = \{q_0, q_1\}$;
- $\delta'(\{q_0, q_1\},0) = \delta(q_0,0) \cup \delta(q_1,0) = \{q_0, q_1\}$;
- $\delta'(\{q_0, q_1\},1) = \delta(q_0,1) \cup \delta(q_1,1) = \{q_0, q_1\}$;

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

La tavola di transizione che riassume la definizione della funzione δ' è:

δ'	$\{q_0\}$	$\{q_1\}$	$\{q_0, q_1\}$
0	$\{q_0, q_1\}$	–	$\{q_0, q_1\}$
1	$\{q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_1\}$

Il grafo degli stati di M' è indicato in Figura 6.14:

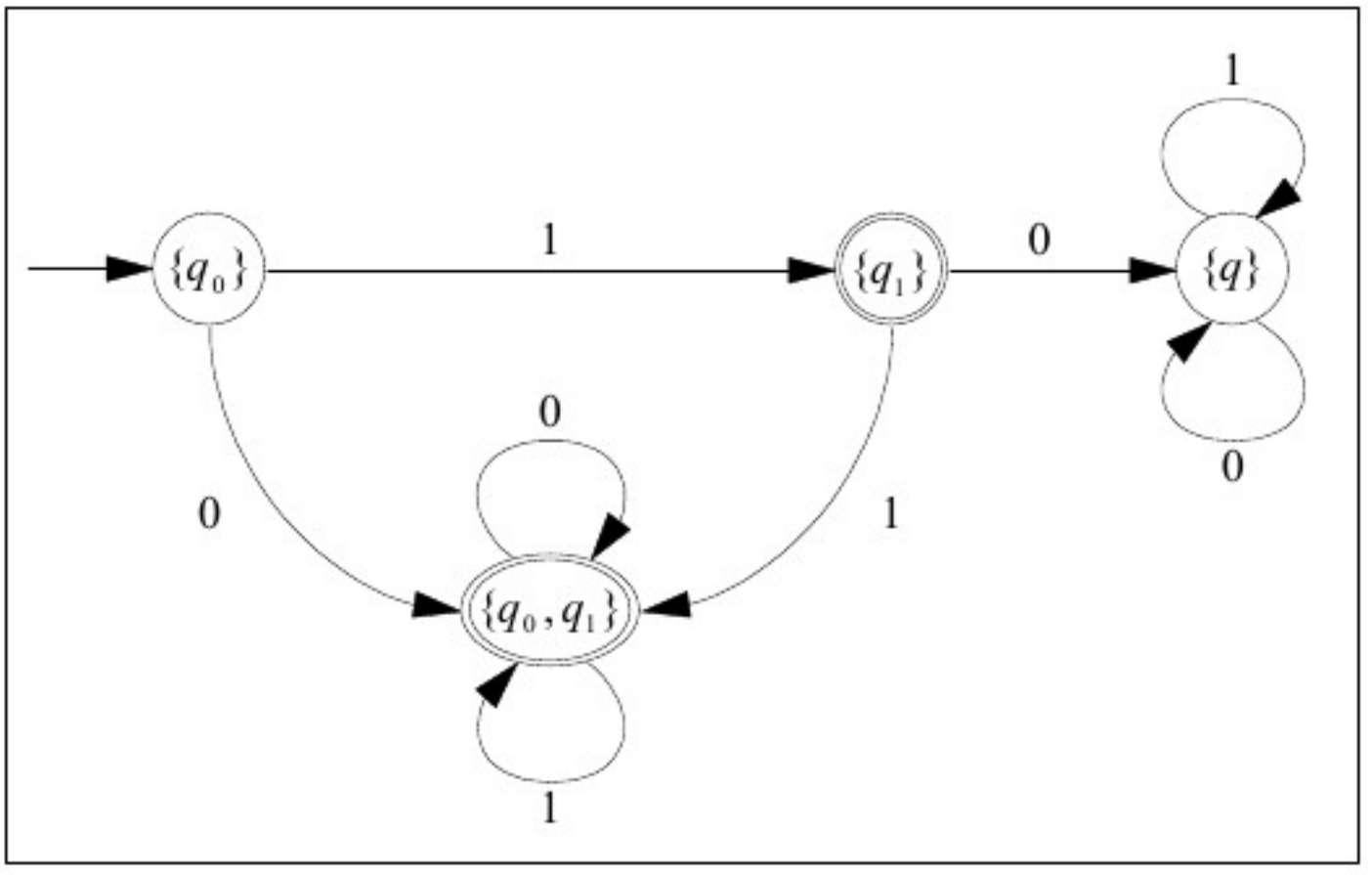


Figura 6.14

6.8 Esercizi proposti

Esercizio 1

Sia dato il seguente linguaggio:

$L = \{ w \in \{a, b, \ldots, z\}^*$: w non contiene come sottostringa *italcasse* $\}$

Costruire un automa a stati finiti deterministico che accetta L.

7. Linguaggi regolari, espressioni regolari e teorema di Kleene

DESCRIZIONE ALGEBRICA DEI LINGUAGGI LINEARI DESTRI

Grammatiche generative	**Automi a stati finiti accettori**	**Espressioni regolari**
GENERATORI	***RICONOSCITORI***	
procedure effettive per		
Costruire stringhe che appartengono al linguaggio	***Stabilire se una stringa appartiene o no ad un linguaggio***	
Definizioni operazionali		***Definizione denotazionale***

7.1 Linguaggi regolari ed espressioni regolari

Definiamo una nuova classe di linguaggi:

<u>Definizione 7.1</u> (Linguaggio regolare)

Sia X un alfabeto finito. Un *linguaggio* $L \subseteq X^*$ è *regolare* se:

- L è finito;

oppure

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

- L può essere ottenuto per induzione utilizzando una delle seguenti operazioni:

  (1) $L = L_1 \cup L_2$, con L_1, L_2 regolari;

  (2) $L = L_1 \cdot L_2$, con L_1, L_2 regolari;

  (3) $L = L_1^*$, con L_1 regolare. ■

Si noti che $\varnothing$ e $\{\lambda\}$ sono linguaggi regolari.

Denotiamo con $\mathcal{L}_{REG}$ la classe dei linguaggi regolari.

<u>Definizione 7.2</u> (Espressione regolare)

Sia X un alfabeto finito. Una stringa R di alfabeto $X \cup \{\lambda, +, *, \cdot, \varnothing, (,)\}$ (con $X \cap \{\lambda, +, *, \cdot, \varnothing, (,)\} = \varnothing$) è una *espressione regolare* di alfabeto X se e solo se vale una delle seguenti condizioni:

(i) $R = \varnothing$;

(ii) $R = \lambda$;

(iii) $R = a$, per ogni $a \in X$;

(iv) $R = (R_1 + R_2)$, *con* R_1, R_2 espressioni regolari di alfabeto X;

(v) $R = (R_1 \cdot R_2)$, *con* R_1, R_2 espressioni regolari di alfabeto X;[12]

(vi) $R = (R_1)^*$, *con* R_1 espressione regolare di alfabeto X. ■

---

[12] Invece di scrivere $(R_1 \cdot R_2)$ si può anche scrivere $(R_1 R_2)$.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Ad ogni espressione regolare R corrisponde un linguaggio regolare $S(R)$ definito nel modo seguente:

ESPRESSIONE REGOLARE	***LINGUAGGIO REGOLARE CORRISPONDENTE***
$\varnothing$	$\varnothing$
λ	$\{\lambda\}$
a	$\{a\}$
$(R_1 + R_2)$	$S(R_1) \cup S(R_2)$
$(R_1 \cdot R_2)$	$S(R_1) \cdot S(R_2)$
$(R_1)^*$	$(S(R_1))^*$

Da una espressione regolare si possono eliminare le coppie di parentesi superflue, tenuto conto che le operazioni ai punti (iv), (v) e (vi) sono elencate in ordine crescente di priorità.

Proposizione 7.1

Un linguaggio su X è regolare se e solo se corrisponde ad una espressione regolare su X.

Quindi, denotato con $\mathcal{R}$ l'insieme delle espressioni regolari di alfabeto X, definiamo la funzione:

$$S : \mathfrak{R} \to 2^{X^*}$$

che ad ogni espressione regolare R associa il corrispondente linguaggio regolare $S(R)$.

Si ha dunque:

$$\mathcal{L}_{REG} = \left\{ L \in 2^{X^*} \mid \exists R \in \Re : L = S(R) \right\}$$

Esempi 7.1

ba^*

$$S(ba^*) = S(b) \cdot S(a^*) = S(b) \cdot \left(S(a)\right)^* = \{b\} \cdot \{a\}^* =$$

= insieme di tutte le parole su $X = \{a,b\}$ che cominciano con una b seguita eventualmente soltanto da delle a.

$a^* \cdot b \cdot a^* \cdot b \cdot a^*$

$$S(a^* \cdot b \cdot a^* \cdot b \cdot a^*) = S(a^*) \cdot S(b) \cdot S(a^*) \cdot S(b) \cdot S(a^*) =$$
$$= \left(S(a)\right)^* \cdot S(b) \cdot \left(S(a)\right)^* \cdot S(b) \cdot \left(S(a)\right)^* =$$
$$= \{a\}^* \cdot \{b\} \cdot \{a\}^* \cdot \{b\} \cdot \{a\}^* =$$

= insieme di tutte le parole su $X = \{a,b\}$ che contengono esattamente due b.

$(a+b)^*$

$$S\left((a+b)^*\right) = \left(S(a+b)\right)^* = \left(S(a) \cup S(b)\right)^* = \left(\{a\} \cup \{b\}\right)^*$$

$= \{a,b\}^*$ = insieme di tutte le parole su $X = \{a,b\}$.

$(b+abb)^*$

$$S\left((b+abb)^*\right) = \left(S(b+abb)\right)^* = \left(S(b) \cup S(abb)\right)^* =$$
$$= \left(S(b) \cup \left(S(a) \cdot S(b) \cdot S(b)\right)\right)^* =$$
$$= \left(\{b\} \cup \left(\{a\} \cdot \{b\} \cdot \{b\}\right)\right)^* =$$
$$= \left(\{b\} \cup \{abb\}\right)^* = \{b, abb\}^* =$$

= insieme di tutte le parole su $X = \{a,b\}$ in cui ogni a è immediatamente seguita da almeno due b.

Osservazione 7.1

Un linguaggio regolare può essere descritto da più di una espressione regolare, ossia $S : \Re \to 2^{X^*}$ non è una funzione iniettiva.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Esempi 7.2

Il linguaggio costituito da tutte le parole su $X = \{a, b\}$ con a e b che si alternano (e che cominciano e terminano con b) può essere descritto sia dall'espressione regolare

$$b \cdot (ab)^*$$

sia da

$$(ba)^* b$$

Definizione 7.3 (Espressioni regolari equivalenti)

Due espressioni regolari R_1 e R_2 su X sono *equivalenti* (per abuso di notazione, scriviamo $R_1 = R_2$) se e solo se $S(R_1) = S(R_2)$. ■

Si ha, pertanto:

$$b \cdot (ab)^* = (ab)^* b$$

7.2 Proprietà delle espressioni regolari

Siano R_1, R_2 ed R_3 espressioni regolari di alfabeto X, risulta:

1) $(R_1 + R_2) + R_3 = R_1 + (R_2 + R_3) =$ Proprietà associativa
$= R_1 + R_2 + R_3$

2) $R_1 + R_2 = R_2 + R_1$ Proprietà commutativa

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

3)	$R_1 + \varnothing = \varnothing + R_1 = R_1$	$\varnothing$ è l'elemento neutro rispetto all'operazione "+" definita sulle espressioni regolari
4)	$R_1 + R_1 = R_1$	Idempotenza
5)	$(R_1 \cdot R_2) \cdot R_3 = R_1 \cdot (R_2 \cdot R_3) = = R_1 \cdot R_2 \cdot R_3$	Proprietà associativa
6)	$R_1 \cdot R_2 \neq R_2 \cdot R_1$	In generale
7)	$R_1 \cdot \lambda = \lambda \cdot R_1 = R_1$	λ è l'elemento neutro rispetto all'operazione "·" definita sulle espressioni regolari
8)	$R_1 \cdot \varnothing = \varnothing \cdot R_1 = \varnothing$	$\varnothing$ è l'elemento assorbente rispetto all'operazione "·"
9)	$R_1 \cdot (R_2 + R_3) = = (R_1 \cdot R_2) + (R_1 \cdot R_3)$	Proprietà distributiva di "·" rispetto all'operazione "+" (distributività sinistra).
10)	$(R_1 + R_2) \cdot R_3 = = (R_1 \cdot R_3) + (R_2 \cdot R_3)$	Proprietà distributiva di "·" rispetto all'operazione "+" (distributività destra).

11) $(R_1)^* = (R_1)^* \cdot (R_1)^* = \left((R_1)^*\right)^* = (\lambda + R_1)^*$

12) $(\varnothing)^* = (\lambda)^* = \lambda$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

13) $(R_1)^* = \lambda + R_1 + R_1^{\ 2} + ... + R_1^{\ n} + (R_1^{\ n+1} \cdot R_1^{\ *})$

Caso particolare:

$(R_1)^* = \lambda + R_1 \cdot R_1^{\ *} = \lambda + R_1^{\ *} \cdot R_1$

14) $(R_1 + R_2)^* = \left(R_1^{\ *} + R_2^{\ *}\right)^* = \left(R_1^{\ *} \cdot R_2^{\ *}\right)^* =$

$= \left(R_1^{\ *} \cdot R_2\right)^* \cdot R_1^{\ *} = R_1^{\ *} \cdot \left(R_2 \cdot R_1^{\ *}\right)^*$

15) $(R_1 + R_2)^* \neq R_1^{\ *} + R_2^{\ *}$ In generale

16) $R_1^{\ *} \cdot R_1 = R_1 \cdot R_1^{\ *}$

17) $R_1 \cdot (R_2 \cdot R_1)^* = (R_1 \cdot R_2)^* \cdot R_1$

18) $\left(R_1^{\ *} \cdot R_2\right)^* = \lambda + (R_1 + R_2)^* \cdot R_2$

19) $\left(R_1 \cdot R_2^{\ *}\right)^* = \lambda + R_1 \cdot (R_1 + R_2)^*$

20) Supponiamo che $\lambda \in S(R_2)$:

$R_1 = R_2 \cdot R_1 + R_3$ se e solo se $R_1 = R_2^{\ *} \cdot R_3$

$R_1 = R_1 \cdot R_2 + R_3$ se e solo se $R_1 = R_3 \cdot R_2^{\ *}$

Per esercizio: dimostrare le 1) - 20).

Aiuto: le 1) - 5) e le 7) - 14) si dimostrano ricorrendo alla funzione *S*; comunque la maggior parte delle 1) - 20) può essere provata con una tecnica generale, detta *dimostrazione mediante riparsificazione*.

Illustriamo questa tecnica dimostrando la 17):

$$R_1 \cdot (R_2 \cdot R_1)^* = (R_1 \cdot R_2)^* \cdot R_1$$

Si consideri una qualunque parola:

$$w \in S\left(R_1 \cdot (R_2 \cdot R_1)^*\right),\ \ w = r_1^{\,0}\left(r_2^{\,1} \cdot r_1^{\,1}\right) \cdot \left(r_2^{\,2} \cdot r_1^{\,2}\right) \cdot \ldots \cdot \left(r_2^{\,n} \cdot r_1^{\,n}\right),\ \ n \geq 0,$$

ove:

$$r_1^{\,i} \in S(R_1), \quad i = 0,1,2,\ldots,n$$
$$r_2^{\,j} \in S(R_2), \quad j = 0,1,2,\ldots,n$$

Riparsificando w ed utilizzando la proprietà associativa della concatenazione, si ha:

$$w = \left(r_1^{\,0} \cdot r_2^{\,1}\right) \cdot \left(r_1^{\,1} \cdot r_2^{\,2}\right) \cdot \ldots \cdot \left(r_1^{\,n-1} \cdot r_2^{\,n}\right) \cdot r_1^{\,n}$$

dunque:

$$w \in S\left((R_1 \cdot R_2)^* \cdot R_1\right)$$

Da cui:

$$S\left(R_1 \cdot (R_2 \cdot R_1)^*\right) \subseteq S\left((R_1 \cdot R_2)^* \cdot R_1\right)$$

In modo analogo si dimostra:

$$S\left(R_1 \cdot (R_2 \cdot R_1)^*\right) \supseteq S\left((R_1 \cdot R_2)^* \cdot R_1\right)$$

Un'altra tecnica comune per dimostrare tali proprietà è semplicemente quella di utilizzare proprietà già note.

Mostreremo ora come si usano le proprietà delle espressioni regolari per provare l'equivalenza di espressioni regolari.

<u>Esempio 7.3</u>

Dimostrare la seguente equivalenza:

$$(b + aa^*b) + (b + aa^*b) \cdot (a + ba^*b)^* \cdot (a + ba^*b) = a^*b \cdot (a + ba^*b)^*$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

$$
\begin{aligned}
(b+aa^*b)+(b+aa^*b)\cdot(a+ba^*b)^*\cdot(a+ba^*b) &\overset{9)}{=} \\
&\overset{9)}{=} (b+aa^*b)\cdot\left[\lambda+(a+ba^*b)^*(a+ba^*b)\right]\overset{13)}{=} \\
&\overset{13)}{=} (b+aa^*b)\cdot(a+ba^*b)^*\overset{10)}{=} \\
&\overset{10)}{=} (\lambda+aa^*)\cdot b\cdot(a+ba^*b)^*\overset{13)}{=} \\
&\overset{13)}{=} a^*b\cdot(a+ba^*b)^*
\end{aligned}
$$

7.3 Teorema di Kleene

<u>Teorema 7.1</u> (di Kleene)

$$\mathcal{L}_3 \equiv \mathcal{L}_{FSL} \equiv \mathcal{L}_{REG}$$

Dimostrazione 7.1

Lo schema della dimostrazione è il seguente:

1) $\mathcal{L}_3 \subset \mathcal{L}_{FSL}$ $\qquad$ $(\mathcal{L}_{FSL} \subset \mathcal{L}_3)$

2) $\mathcal{L}_{FSL} \subset \mathcal{L}_{REG}$

3) $\mathcal{L}_{REG} \subset \mathcal{L}_3$

1) $\mathcal{L}_3 \subset \mathcal{L}_{FSL}$

Sia $L \in \mathcal{L}_3 \overset{def}{\Leftrightarrow} \exists\, G = (X, V, S, P)$, G di tipo '3': $L(G) = L$.

Vogliamo costruire un automa a stati finiti $M = (Q, \delta, q_0, F)$ tale che $T(M) = L(G)$.

Allo scopo si fornisce il seguente algoritmo per la costruzione di un automa a stati finiti non deterministico che riconosce il linguaggio generato da una fissata grammatica lineare destra.

Algoritmo 7.1 (Costruzione di un automa a stati finiti non deterministico equivalente ad una grammatica lineare destra).

Data una grammatica lineare destra:

$$G = (X, V, S, P)$$

l'automa accettore a stati finiti equivalente ($T(M) = L(G)$) viene costruito come segue:

$$M = (Q, \delta, q_0, F)$$

I. X come alfabeto di ingresso;

II. $Q = V \cup \{q\},\ q \notin V$;

III. $q_0 = S$;

IV. $F = \{q\} \cup \{B \mid B \to \lambda \in P\}$;

V. $\delta : Q \times X \to 2^Q \quad \ni'$

V.a $\forall B \to aC \in P,\ C \in \delta(B,a)$

V.b $\forall B \to a \in P,\ q \in \delta(B,a)$

L'algoritmo può generare un automa *non deterministico* per effetto dei passi V.a. e V.b. Si può facilmente constatare che, se:

$$w = x_1 x_2 ... x_n \in L(G)$$

w può essere generata da una derivazione del tipo:

$$S \Rightarrow x_1 X_2 \Rightarrow x_1 x_2 X_3 \Rightarrow ... \Rightarrow x_1 x_2 ... x_{i-1} X_i \Rightarrow x_1 x_2 ... x_i$$

Dalla definizione data, l'automa M, esaminando la stringa $w = x_1 x_2 ... x_n$, compie una serie di mosse (o transizioni) che lo portano dallo stato S ad $X_1, X_2, ..., X_i$ e q; pertanto $L(G) \subseteq T(M)$.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

In modo del tutto analogo, ogni w in $T(M)$ comporta una sequenza di mosse dell'automa a cui corrisponde una derivazione in G, e pertanto $T(M) \subseteq L(G)$.

Se ne deduce che:

$$L(G) = T(M) \qquad c.v.d.$$

Sebbene non sia strettamente necessario per la dimostrazione del Teorema di Kleene, per il suo interesse pratico si riporta di seguito l'algoritmo per la costruzione di una grammatica lineare destra che genera il linguaggio accettato da un automa a stati finiti.

Tale algoritmo costituisce una dimostrazione costruttiva del seguente risultato:

$$\mathcal{L}_{FSL} \subset \mathcal{L}_3$$

<u>Algoritmo 7.2</u> (Costruzione di una grammatica lineare destra equivalente ad un automa accettore a stati finiti).

Sia dato un automa accettore a stati finiti:

$$M = (Q, \delta, q_0, F)$$

con alfabeto di ingresso X.

La grammatica lineare destra G equivalente a M, ossia tale che $L(G) = T(M)$, si costruisce come segue:

$$G = (X, V, S, P)$$

I. X = alfabeto di ingresso di M;

II. $V = Q$;

III. $S = q_0$;

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

IV. $$P = \{q \to xq' \mid q' \in \delta(q,x)\} \cup \{q \to x \mid \delta(q,x) \in F\} \cup \\ \cup \{q_0 \to \lambda \mid q_0 \in F\}$$

Si lascia al lettore il compito di dimostrare che $L(G) = T(M)$.

c.v.d.

2) $\mathscr{L}_{FSL} \subset \mathscr{L}_{REG}$

Sia $L \in \mathscr{L}_{FSL} \overset{def}{\Leftrightarrow} \exists\ M = (Q,\ \delta,\ q_0,\ F)$ automa a stati finiti con alfabeto di ingresso X tale che $T(M) = L$.

Supponiamo $Q = \{q_0, q_1, \ldots, q_n\}$.

Definiamo il seguente linguaggio:

$$R_{ij} = \left\{ w \in X^* \;\middle|\; \delta^*(q_i, w) = q_j \right\}$$

costituito da tutte e sole quelle parole che fanno transitare M dallo stato q_i allo stato q_j.

Dalla definizione di linguaggio accettato da un automa a stati finiti (Definizione 6.4), segue immediatamente:

$$L = \bigcup_{q_j \in F} R_{0j}$$

ossia L può essere riguardato come l'insieme delle parole che fanno transitare M dallo stato iniziale q_0 ad uno qualunque degli stati finali.

Se dimostriamo che ciascuno degli R_{ij}, $0 \le i, j \le n$, è un linguaggio regolare, allora risulterà dimostrato che L è regolare, in quanto unione finita di linguaggi regolari.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Sia:

$$R_{ij}^{k} = \left\{ w \in X^{*} \;\middle|\; \begin{array}{c} w \text{ fa transitare } M \text{ da } q_i \text{ a } q_j \text{ senza passare} \\ \text{per nessuno degli stati } q_k, q_{k+1}, ..., q_n \end{array} \right\}$$

Si noti che:

$$R_{ij}^{n+1} = R_{ij} \,. \qquad (*)$$

Dimostriamo:

$$R_{ij}^{k} \in \mathcal{L}_{REG} \;\; \forall\, i, j\colon 0 \le i, j \le n$$

per induzione su k.

<u>Passo base</u>

$k = 0$

$$R_{ij}^{0} = \left\{ x \in X \;\middle|\; \delta(q_i, x) = q_j \right\}$$

è finito (poiché è un sottoinsieme di X) e quindi è per definizione regolare.

<u>Passo induttivo</u>

R_{ij}^{k} regolare $\forall\, i, j\colon 0 \le i, j \le n$

Si intende dimostrare che R_{ij}^{k+1} è regolare.

Sia $w \in R_{ij}^{k+1}$ (Figura 7.1); per definizione la lettura di w non fa transitare M in nessuno degli stati $q_{k+1}, q_{k+2}, \ldots, q_n$.

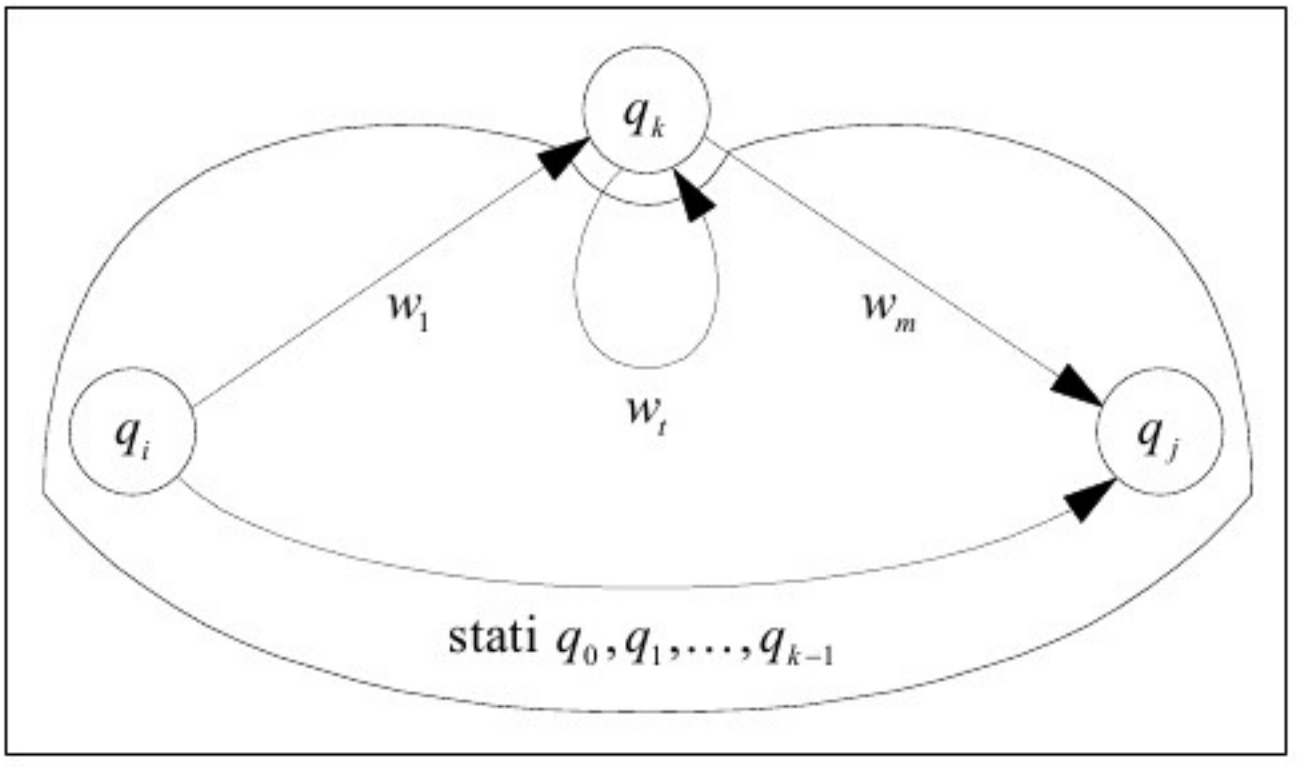


Figura 7.1

Si possono verificare due casi:

a) w non fa transitare M in q_k, ossia $w \in R_{ij}^k$ che, per ipotesi di induzione, è regolare.

   Ciò implica che R_{ij}^{k+1} è regolare.

b) w fa transitare M in q_k.

Ci resta da dimostrare che R_{ij}^{k+1} è regolare anche nel caso b).

In questo caso, w può essere riguardata come la concatenazione di m parole, $m \geq 2$:

$$w = w_1\, w_2 \ldots w_{m-1}\, w_m \qquad m \geq 2$$

ove (con riferimento a Figura 7.1):

$$w_1 \in R_{ik}^k \qquad w_t \in R_{kk}^k \;\; \forall\, t\colon 2 \leq t \leq m\text{-}1 \qquad w_m \in R_{kj}^k$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Si ha dunque:

$$w \in R_{ik}^{k} \cdot \left(R_{kk}^{k}\right)^{m-2} \cdot R_{kj}^{k} \qquad \forall\ m \geq 2$$

da cui:

$$w \in R_{ik}^{k} \cdot \left(R_{kk}^{k}\right)^{*} \cdot R_{kj}^{k}$$

Dunque:

$$R_{ij}^{k+1} = R_{ij}^{k} \cup R_{ik}^{k} \cdot \left(R_{kk}^{k}\right)^{*} \cdot R_{kj}^{k}$$

e R_{ij}^{k+1} risulta espresso come unione, concatenazione ed iterazione di linguaggi regolari.

Risulta così dimostrato che R_{ij}^{k} è regolare, per ogni i, j, $0 \leq i, j \leq n$ e per ogni k, $0 \leq k \leq n$.

Si ha dunque:

$$L = \bigcup_{q_j \in F} R_{0j} \overset{\text{per la (*)}}{=} \bigcup_{q_j \in F} R_{0j}^{n+1}$$

è un linguaggio regolare. *c.v.d.*

3) $\mathcal{L}_{REG} \subset \mathcal{L}_3$

Sia $L \in \mathcal{L}_{REG}$. Per definizione di linguaggio regolare (Definizione 7.1), o L è finito oppure L può essere ottenuto induttivamente come segue:

a) $L = L_1 \cup L_2$, $\qquad L_1, L_2$ regolari

b) $L = L_1 \cdot L_2$, $\qquad L_1, L_2$ regolari

c) $L = L_1^{*}$ $\qquad L_1$ regolare

Procediamo per *induzione sulla costruzione* di L.

Passo Base (induzione sulla costruzione di L)

L è finito

$$L = \{w_1, w_2, \ldots, w_n\} \qquad n \geq 0$$

$$L = L_1 \cup L_2 \cup \ldots \cup L_n = \bigcup_{i=1}^{n} L_i$$

ove $L_i = \{w_i\}$, $1 \leq i \leq n$.

Abbiamo ricondotto la dimostrazione a provare che L_i è lineare destro, per ogni i, $1 \leq i \leq n$.

Procediamo per *induzione* su n.

Passo Base (induzione su n)

$n = 0$

Esiste sicuramente una grammatica lineare destra che genera il linguaggio $L = \varnothing$ (si consideri, per esempio, la grammatica G: $S \rightarrow aA$).

Passo induttivo (induzione su n)

Sia $L = \{w_1, w_2, \ldots, w_n, w_{n+1}\}$ un linguaggio regolare ($n \geq 0$).

$$L = L_1 \cup L_2 \cup \ldots \cup L_{n+1} = \bigcup_{i=1}^{n+1} L_i$$

ove $L_i = \{w_i\}$, $1 \leq i \leq n+1$.

Se dimostriamo che ogni L_i, $1 \leq i \leq n+1$, è un linguaggio lineare destro, allora anche L risulterà lineare destro (per la proprietà di chiusura di $\mathcal{L}_3$ rispetto all'unione - Teorema 5.3).

Consideriamo $L_i = \{w_i\}$, $1 \leq i \leq n+1$.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Poiché (X^*, ·) è il monoide libero generato da X (si veda Osservazione 2.1), si ha:

$$w_i = x_{i1} \cdot x_{i2} \cdot \ldots \cdot x_{im_i} \qquad x_{ij} \in V \qquad 1 \le i \le n+1,\ 1 \le j \le m_i,\ m_i \ge 0$$

$$L_i = \{w_i\} = \{x_{i1}\} \cdot \{x_{i2}\} \cdot \ldots \cdot \{x_{im_i}\} = L_{i1} \cdot L_{i2} \cdot \ldots \cdot L_{im_i}$$

e $G_{ij} : S \to x_{ij}$ è la grammatica lineare destra che genera L_{ij}, $1 \le i \le n+1$, $1 \le j \le m_i$, $m_i \ge 0$ ($L_{ij} = L(G_{ij})$).

Dunque L è ottenuto per unione e concatenazione di linguaggi lineari destri. Per le proprietà di chiusura di $\mathcal{LB}$ rispetto alle suddette operazioni (Teorema 5.3), L risulta essere un linguaggio lineare destro.

<u>Passo induttivo</u> (induzione sulla costruzione di L)

L è ottenuto induttivamente come segue:

a) $L = L_1 \cup L_2$, $\qquad L_1, L_2$ regolari

b) $L = L_1 \cdot L_2$, $\qquad L_1, L_2$ regolari

c) $L = L_1^*$ $\qquad L_1$ regolare

In tutti e tre i casi, la tesi *L è lineare destro* segue immediatamente dalla ipotesi di induzione (L_1, L_2 *lineari destri*) e dal Teorema 5.3.

c.v.d.

7.4 Pumping Lemma per i linguaggi regolari

<u>Teorema 7.2</u> (Pumping Lemma per i linguaggi regolari)

Sia $M = (Q, \delta, q_0, F)$ un automa accettore a stati finiti con n stati ($|Q| = n$) e sia $z \in T(M)$, $|z| \ge n$.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Allora z può essere scritta come uvw, e $uv^*w \subset T(M)$ (ossia $\forall i,\ i \geq 0 : uv^i w \in T(M)$).

Una formulazione alternativa è la seguente:

Sia $L = T(M)$ un linguaggio regolare con $M = (Q, \delta, q_0, F)$ un automa accettore a stati finiti. Allora $\exists n = |Q|$ t.c. $\forall z \in L,\ |z| \geq n : z = uvw$ e:

(1) $|uv| \leq n$

(2) $v \neq \lambda$

(3) $uv^i w \in L,\ \forall i,\ i \geq 0$

<u>Dimostrazione 7.2</u>

Sia $z = x_1\, x_2 \ldots x_k$, $z \in T(M)$.

Possiamo rappresentare il comportamento dell'automa M, con ingresso z, come segue (Figura 7.2):

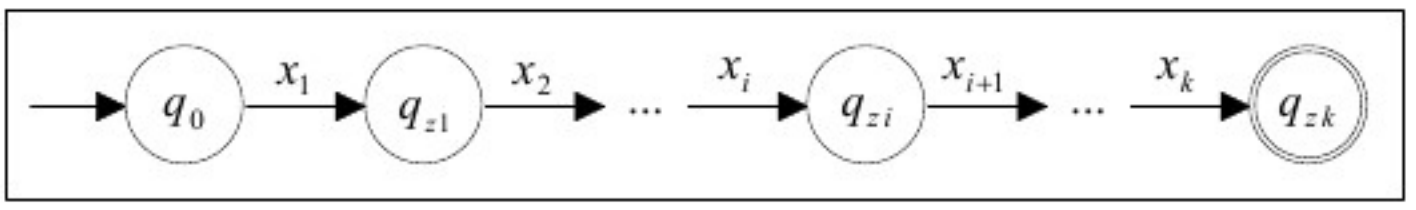


Figura 7.2

Se si ha: $|z| \geq n$, nella Figura 7.2 devono comparire almeno $n+1$ stati ma, poiché M ha solo n stati distinti ($|Q| = n$) almeno una stato tra $q_0, q_{z1}, q_{z2}, \ldots, q_{zk}$ deve comparire due volte.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Supponiamo che si abbia $q_{zi} = q_{zj}$, $i < j$.

Si ha dunque la situazione rappresentata in Figura 7.3.

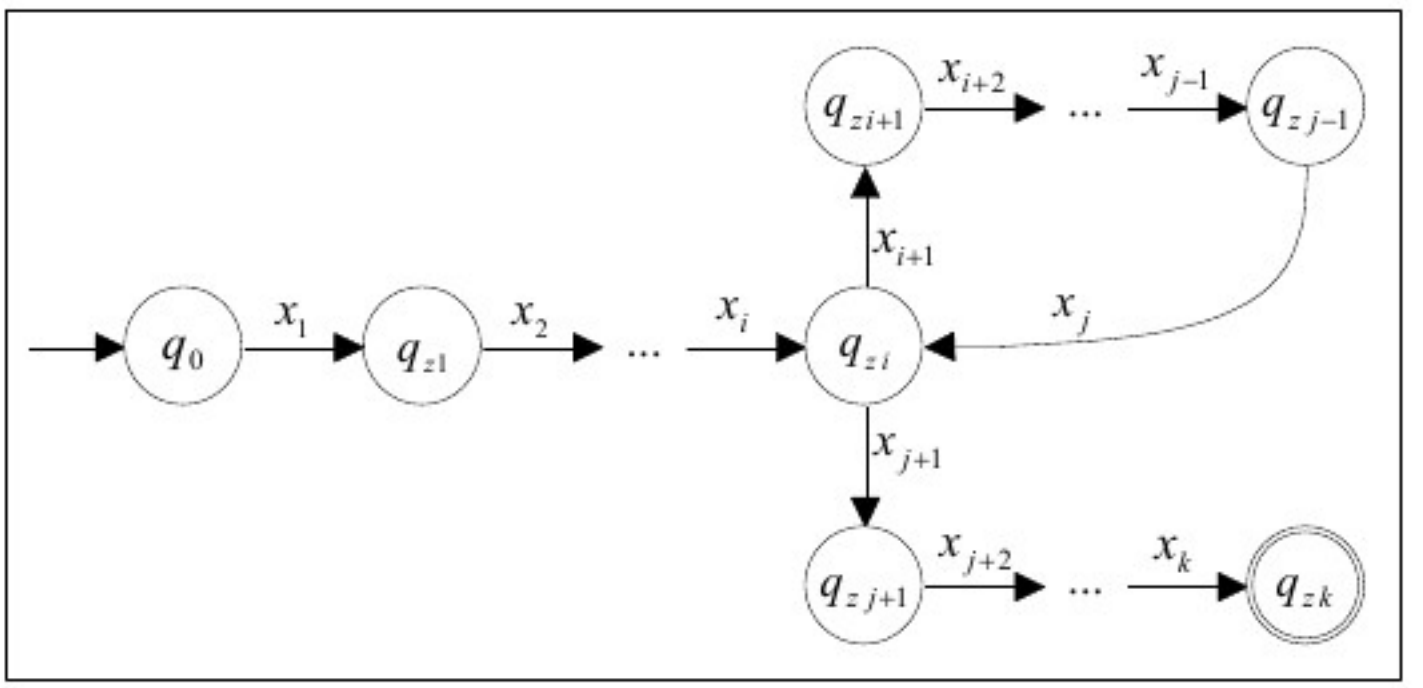


Figura 7.3

Possiamo scrivere z nella forma:

$$z = uvw$$

ove:

$u = x_1 x_2 \ldots x_i$,

$v = x_{i+1} x_{i+2} \ldots x_j$,

$w = x_{j+1} x_{j+2} \ldots x_k$.

Poiché $z \in T(M)$, l'automa M, per effetto dell'ingresso di $z = uvw$, si porta per definizione in uno stato finale ($\delta^*(q_0, z) \in F$).

Ma è immediato osservare che tale stato è lo stesso in cui M si porta per effetto dell'ingresso delle parole uw, uv^2w,… uv^iw, $i \geq 0$.

Dunque si ha: $uv^iw \in T(M)$, $i \geq 0$. *c.v.d.*

Proprietà decidibili dei linguaggi regolari

Come corollario del Pumping Lemma per i linguaggi regolari si ha :

Teorema 7.3

Sia $M = (Q, \delta, q_0, F)$ un automa accettore a stati finiti con $|Q| = n$.

(i) '$T(M) \neq \varnothing$' iff $\exists z \in T(M) : |z| < n$ ('L vuoto ?')

(ii) '$|T(M)| = \infty$' iff $\exists z \in T(M) : n \leq |z| < 2n$ ('L infinito ?')

Teorema 7.4

Siano M_1 e M_2 automi accettori a stati finiti.

'$T(M_1) = T(M_2)$' è decidibile.

Dimostrazione 7.4

Siano M_1 e M_2 automi accettori a stati finiti.

Dimostrare che '$T(M_1) = T(M_2)$' è decidibile equivale a provare che $\big(T(M_1) \setminus T(M_2)\big) \cup \big(T(M_2) \setminus T(M_1)\big) = \varnothing$ è decidibile. A tal proposito basta osservare che

$$T(M_1) \setminus T(M_2) = T(M_1) \cap \overline{T(M_2)}$$
$$T(M_2) \setminus T(M_1) = T(M_2) \cap \overline{T(M_1)}$$

e che $\mathcal{L}_{REG}$ è chiuso rispetto a intersezione e complemento.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

7.5 Esercizi

Esercizio 7.1

Determinare una grammatica lineare destra che genera il linguaggio descritto dalla seguente espressione regolare:

$$b^* + (ab)^*$$

Il linguaggio relativo all'espressione regolare è:

$$\begin{aligned} S\left(b^* + (ab)^*\right) &= S(b^*) \cup S\left((ab)^*\right) = \left(S(b)\right)^* \cup \left(S(ab)\right)^* = \\ &= \{b\}^* \cup \left(S(a) \cdot S(b)\right)^* = \{b\}^* \cup \left(\{a\} \cdot \{b\}\right)^* = \\ &= \{b\}^* \cup \{ab\}^* \end{aligned}$$

Costruiamo dapprima la grammatica G_1 tale che

$$L(G_1) = S(b^*) = \{b\}^*$$

$$G_1 = (X_1, V_1, S_1, P_1)$$

ove:

$$X_1 = \{b\} \qquad V_1 = \{S_1\} \qquad P_1 = \{S_1 \to bS_1 \mid \lambda\}.$$

Costruiamo ora la grammatica G_2 tale che

$$L(G_2) = S(ab) = \{ab\}$$

$$G_2 = (X_2, V_2, S_2, P_2)$$

ove:

$$X_2 = \{a,b\} \qquad V_2 = \{S_2, B_2\} \qquad P_2 = \{S_2 \to aB_2,\ B_2 \to b\}.$$

La grammatica G_3 tale che

$$L(G_3) = S\left((ab)^*\right) = \{ab\}^*$$

è dunque (si veda Capitolo 5, Tavola 1):

$$G_3 = (X_3, V_3, S_3, P_3)$$

ove:

$$X_3 = X_2 = \{a,b\} \qquad V_3 = V_2 \cup \{S_3\}$$

$$\begin{aligned} P_3 &= \{S_3 \to \lambda\} \cup (P_2 - \{S_2 \to \lambda\}) \cup \{S_3 \to w \mid S_2 \to w \in P_2\} \cup \\ &\quad \cup \{A \to bS_3 \mid A \to b \in P_2\} = \\ &= \{S_3 \to \lambda\} \cup P_2 \cup \{S_3 \to aB_2\} \cup \{B_2 \to bS_3\} = \\ &= \{S_3 \to \lambda,\ S_2 \to aB_2,\ B_2 \to b,\ S_3 \to aB_2,\ B_2 \to bS_3\} = \\ &= \{S_3 \to \lambda \mid aB_2,\ S_2 \to aB_2,\ B_2 \to b \mid bS_3\}. \end{aligned}$$

Si osservi che il nonterminale S_2 non compare nella parte destra di nessuna produzione in P_3 (S_2 è un nonterminale inutile - si veda Definizione 8.12), dunque l'intera produzione ($S_2 \to aB_2$) può essere rimossa da P_3 senza alterare il linguaggio generato da G_3. Quindi P_3 diventa:

$$P_3 = \{S_3 \to \lambda \mid aB_2,\ B_2 \to b \mid bS_3\}.$$

La grammatica G tale che:

$$L(G) = S(b^*) \cup S\left((ab)^*\right) = S\left(b^* + (ab)^*\right)$$

è dunque:

$$G = (X, V, S, P)$$

ove:

$$X = X_1 \cup X_3 = \{a,b\}$$

$$V = V_1 \cup V_3 = V_1 \cup V_2 \cup \{S_3\} \cup \{S\} = \{S, S_1, B_2, S_3\}$$

$$\begin{aligned} P &= \{S \to w \mid S_1 \to w \in P\} \cup \{S \to w \mid S_3 \to w \in P_3\} \cup P_1 \cup P_3 = \\ &= \{S \to bS_1 \mid \lambda\} \cup \{S \to aB_2 \mid \lambda\} \cup \{S_1 \to bS_1 \mid \lambda\} \cup \\ &\quad \cup \{S_3 \to \lambda \mid aB_2,\ B_2 \to b \mid bS_3\} = \\ &= \{S \to bS_1 \mid aB_2 \mid \lambda,\ S_1 \to bS_1 \mid \lambda,\ S_3 \to aB_2 \mid \lambda,\ B_2 \to bS_3 \mid b\} \end{aligned}$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Esercizio 7.2

Data la seguente grammatica lineare destra:

$$G = (X, V, S, P)$$

ove:

$$X = \{a, b, c\} \qquad V = \{S, A, B\}$$

$$P = \left\{ S \overset{(1)}{\rightarrow} bA \mid aS \mid b,\ \ A \overset{(2)}{\rightarrow} aB \mid cS \mid a,\ \ B \overset{(3)}{\rightarrow} bA \mid cB \mid c \right\}$$

Determinare un'espressione regolare che denota il linguaggio $L(G)$.

Senza rischio di confusione, denotiamo con A, B ed S gli insiemi delle stringhe derivabili in G dai nonterminali A, B ed S, rispettivamente.

Dalla produzione (2) risulta:

$$A = \{a\} \cdot B \cup \{c\} \cdot S \cup \{a\}$$

che, per brevità, scriviamo nella forma:

$$A = aB \cup cS \cup a \qquad (2')$$

Sostituendo in (1) la (2′), si ha:

$$S = b \cdot (aB \cup cS \cup a) \cup aS \cup b$$

che, per la proprietà distributiva della concatenazione rispetto all'unione, è uguale a:

$$S = baB \cup bcS \cup ba \cup aS \cup b$$

Applicando la proprietà commutativa per l'unione, la proprietà associativa per la concatenazione e la distributività della concatenazione rispetto all'unione si ha:

$$S = baB \cup (bc \cup a)S \cup (ba \cup b) \qquad (1')$$

Sostituendo in (3) la (2′), si ottiene:

$$B = b \cdot (aB \cup cS \cup a) \cup cB \cup c = (ba \cup c)B \cup (bcS \cup ba \cup c) \quad (3')$$

Osserviamo la (1′) e la (3′).

Se denotiamo con *A, B* ed *S* tre espressioni regolari (oltre che gli insiemi di stringhe derivabili dai nonterminali *A, B* ed *S*), si ha che l'identità tra espressioni regolari corrispondente alla (1′) è:

$$S = baB + (bc + a)S + (ba + b) \qquad (1'')$$

mentre l'identità tra espressioni regolari corrispondente alla (3′) è:

$$B = (ba + c)B + (bcS + ba + c) \qquad (3'')$$

In entrambi i casi ci troviamo di fronte ad una identità del tipo:

$$R_1 = R_2 \cdot R_1 + R_3 \ \text{ con } \ R_2 \neq \lambda \ .$$

Per la proprietà 20) sulle espressioni regolari, si ha:

$$R_1 = R_2^{\ *} \cdot R_3 \, .$$

Dunque la (3″) diventa:

$$B = (ba + c)^* \cdot (bcS + ba + c) \qquad (3''')$$

Sostituendo la (3‴) nella (1″), si ottiene:

$$\begin{aligned} S &= \left[ba \cdot (ba + c)^* \cdot (bcS + ba + c)\right] + (bc + a)S + (ba + b) = \\ &= ba \cdot (ba + c)^* bcS + (a + bc)S + ba \cdot (ba + c)^* \cdot (ba + c) + ba + b = \\ &= \left(ba \cdot (ba + c)^* bc + a + bc\right)S + ba \cdot (ba + c)^* \cdot (ba + c) + ba + b = \end{aligned}$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

per la proprietà 20) sulle espressioni regolari

$$= \left(ba \cdot (ba+c)^* bc + a + bc\right)^* \cdot \left(ba \cdot (ba+c)^* (ba+c) + ba + b\right).$$

Esercizio 7.3

Sia L il linguaggio denotato dalla seguente espressione regolare:

$$(aa + aaa)^*$$

1) Trovare un automa a stati finiti che riconosce L.
2) Trasformare l'automa non deterministico trovato al punto 1) in un automa deterministico equivalente.

1) Determiniamo innanzitutto due grammatiche lineari destre G_1 e G_2 tali che:

$$L(G_1) = S(aa) = \{aa\} \qquad L(G_2) = S(aaa) = \{aaa\}$$

$$G_1 = (X, V_1, S_1, P_1) \qquad G_2 = (X, V_2, S_2, P_2)$$

$$X = \{a\}$$

$$V_1 = \{S_1, A\} \qquad V_2 = \{S_2, B, C\}$$

$$P_1 = \{S_1 \to aA,\ A \to a\} \qquad P_2 = \{S_2 \to aB,\ B \to aC,\ C \to a\}$$

Determiniamo poi la grammatica lineare destra G_3 tale che:

$$L(G_3) = L(G_1) \cup L(G_2) = S(aa) \cup S(aaa) = S(aa + aaa)$$

$$G_3 = (X, V_3, S_3, P_3)$$

dove:

$$V_3 = V_1 \cup V_2 \cup \{S_3\} = \{S_1, S_2, S_3, A, B, C\}.$$

Le produzioni di G_3 si ottengono come segue:

$$P_3 = \{S_3 \to w \mid S_1 \to w \in P_1\} \cup \{S_3 \to w \mid S_2 \to w \in P_2\} \cup P_1 \cup P_2.$$

Dunque si ha:

$$P_3 = \{S_3 \to aA \,|\, aB,\ S_1 \to aA,\ A \to a,\ S_2 \to aB,\ B \to aC,\ C \to a\}$$

da cui possiamo eliminare le produzioni che presentano S_1 o S_2 nella parte sinistra (S_1 ed S_2 sono nonterminali inutili perché non sono presenti nella parte destra di nessuna produzione e né S_1 né S_2 sono il simbolo distintivo di G_3). Per cui P_3 diventa:

$$P_3 = \{S_3 \to aA \,|\, aB,\ A \to a,\ B \to aC,\ C \to a\}$$

Determiniamo ora la grammatica lineare destra G tale che:

$$L(G) = \left(L(G_3)\right)^* = S\left((aa + aaa)^*\right)$$
$$G = (X, V, S, P)$$

dove:

$$V = V_3 \cup \{S\} = \{S, S_3, A, B, C\}$$

Le produzioni di G si ottengono come segue (si veda Tavola 1):

$$\begin{aligned} P = {} & \{S \to \lambda\} \cup (P_3 - \{S_3 \to \lambda\}) \cup \{S \to w \,|\, S_3 \to w \in P_3\} \cup \\ & \cup \{A \to bS \,|\, A \to b \in P_3\} \cup \\ & \cup \{A \to bS \,|\, A \to bB \in P_3,\ b \neq \lambda,\ B \to \lambda \in P_1\}. \end{aligned}$$

Dunque si ha:

$$\begin{aligned} P = {} & \{S \to \lambda\} \cup \{S_3 \to aA \,|\, aB,\ A \to a,\ B \to aC,\ C \to a\} \cup \\ & \cup \{S \to aA \,|\, aB\} \cup \{A \to aS,\ C \to aS\} \end{aligned}$$

da cui possiamo eliminare le produzioni che presentano S_3 nella parte sinistra, ottenendo:

$$P = \{S \to aA \,|\, aB \,|\, \lambda,\ A \to aS \,|\, a,\ B \to aC,\ C \to aS \,|\, a\}.$$

L'automa non deterministico che riconosce $L(G)$ si costruisce come segue:

$M = (Q, \delta, q_0, F)$ con alfabeto di ingresso X

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

ove, per il relativo algoritmo di trasformazione (Algoritmo 7.1), si ha:

1) $Q = V \cup \{q\} = \{S, A, B, C, q\}$;

2) $q_0 = S$;

3) $F = \{q, S\}$;

4) δ è definita dalla seguente tavola di transizione:

δ	S	A	B	C	q
a	$\{A, B\}$	$\{S, q\}$	$\{C\}$	$\{S, q\}$	$-$

Quindi il grafo degli stati è come in Figura 7.4.

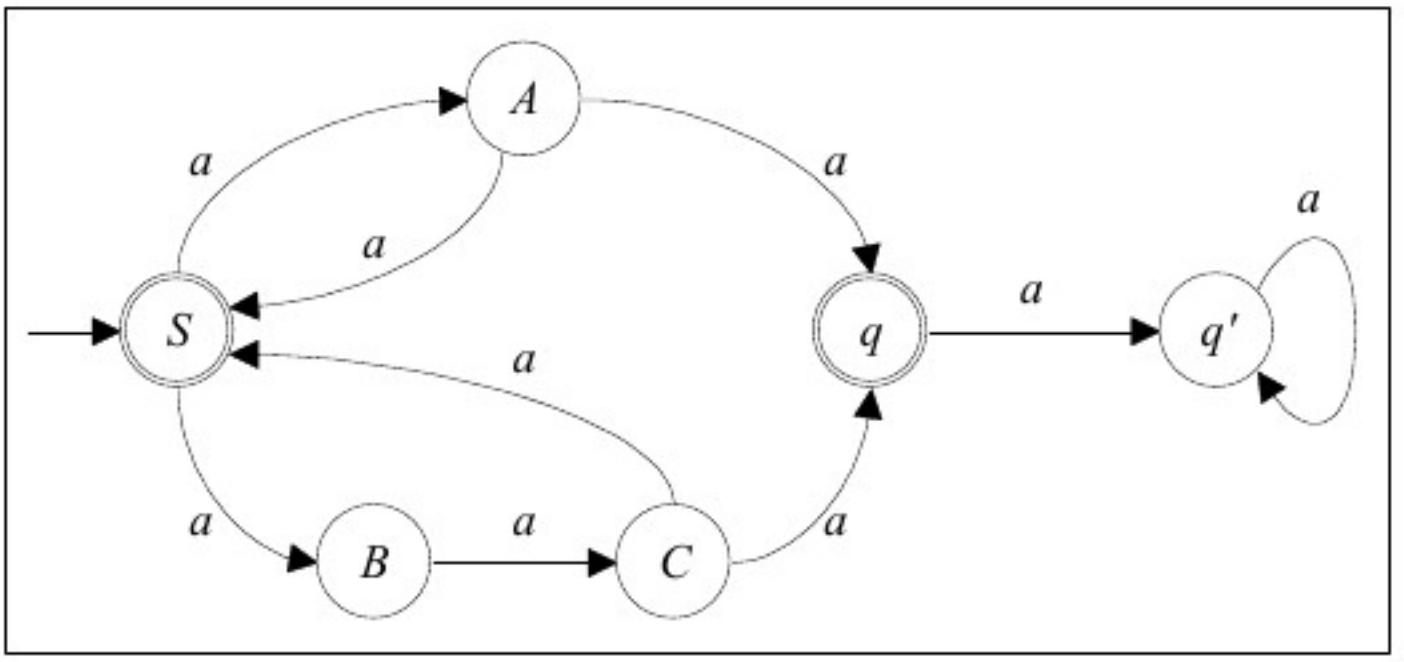


Figura 7.4

2) L'automa accettore deterministico M' equivalente ad M (ossia tale che $T(M') = T(M)$) si costruisce, secondo l'Algoritmo 6.1, come segue:

$$M' = (Q', \delta', q_0', F')$$

ove:

a) $Q' = 2^Q$;

b) $q_0' = \{q_0\}$;

c) $F' = \{p \subseteq Q \mid p \cap F \neq \varnothing\}$;

d) $\delta' : Q' \times X \to Q' \quad \ni'$

$\forall q' = \{q_1, q_2, ... q_i\} \in Q', \ \forall x \in X$:

$\delta'(q', x) = \delta'(\{q_1, q_2, ..., q_i\}, x) =$

$$= \delta(q_1, x) \cup \delta(q_2, x) \cup ... \cup \delta(q_i, x) = \bigcup_{j=1}^{i} \delta(q_j, x)$$

Dunque si ha:

$$M' = (Q', \delta', q_0', F')$$

con:

a) $Q' = 2^Q = 2^{\{S,A,B,C,q\}}, \ |Q'| = 32$;

b) $q_0' = \{S\}$;

c) $F' = \left\{ \begin{matrix} \{q\}, \{S\}, \{q, A\}, \{q, B\}, \{q, C\}, ..., \\ \{S, A\}, \{S, B\}, \{S, C\}, ..., \{q, S\} \end{matrix} \right\}$.

Nella pratica, non è necessario considerare tutti gli stati di Q', in quanto molti sono irraggiungibili. Per questo motivo, iniziamo a definire δ' dal nuovo stato iniziale e proseguiamo definendo δ' per un nuovo stato di Q' non appena esso viene generato.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Per cui si ha:

$$\delta'(\{S\},a) = \delta(S,a) = \{A,B\}$$

$$\delta'(\{A,B\},a) = \delta(A,a) \cup \delta(B,a) = \{S,q\} \cup \{C\} = \{S,C,q\}$$

$$\begin{aligned}\delta'(\{S,C,q\},a) &= \delta(S,a) \cup \delta(C,a) \cup \delta(q,a) = \\ &= \{A,B\} \cup \{S,q\} \cup \varnothing = \{S,A,B,q\}\end{aligned}$$

$$\begin{aligned}\delta'(\{S,A,B,q\},a) &= \delta(S,a) \cup \delta(A,a) \cup \delta(B,a) \cup \delta(q,a) = \\ &= \{A,B\} \cup \{S,q\} \cup \{C\} \cup \varnothing = \\ &= \{S,A,B,C,q\}\end{aligned}$$

$$\delta'(\{S,A,B,C,q\},a) = \delta(\{S,A,B,q\},a) \cup \delta(C,a) = \{S,A,B,C,q\}$$

Il grafo degli stati di M' è dato in Figura 7.5.

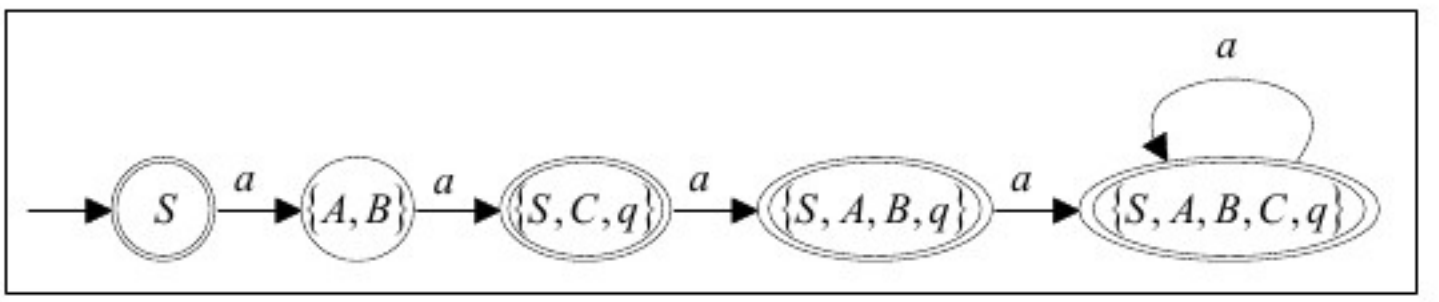


Figura 7.5

Quanto illustrato in precedenza costituisce il modo "standard" di svolgere l'esercizio.

Un modo alternativo è il seguente.

$$\left((aa + aaa)^*\right)$$

Osserviamo che:

$$S\left((aa + aaa)^*\right) = \{\lambda, aa, aaa, aaaa, aaaaa, \ldots\} = \{a\}^* - \{a\}$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

L'automa (minimo e deterministico) che riconosce $\{a\}^* - \{a\}$ è (Figura 7.6):

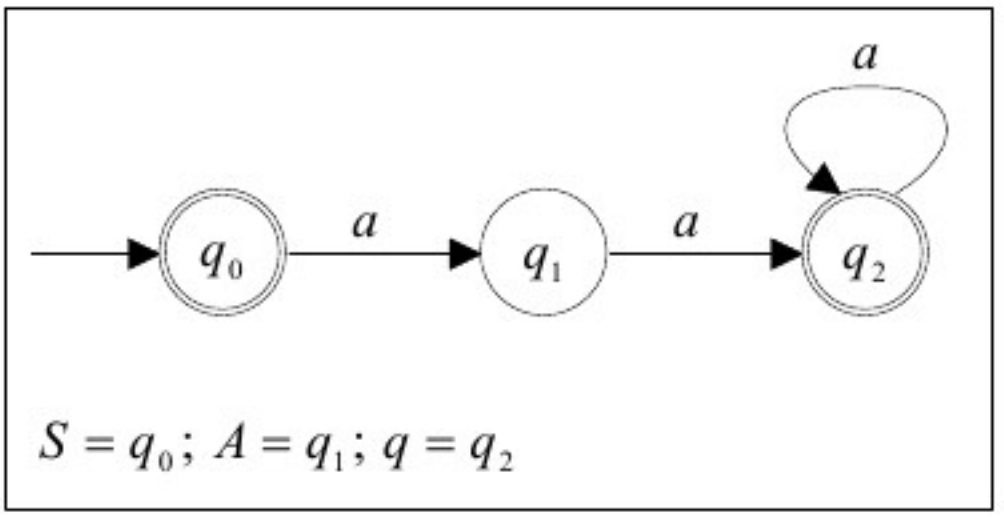


Figura 7.6

corrispondente alla grammatica:

$$S \to aA \mid \lambda$$
$$A \to aA \mid a$$

Si osservi che, data la grammatica *G,* le cui produzioni sono:

$$S \to aA \mid \lambda$$
$$A \to aA \mid a$$

l'automa (non deterministico) che riconosce $L(G)$ è (Figura 7.7):

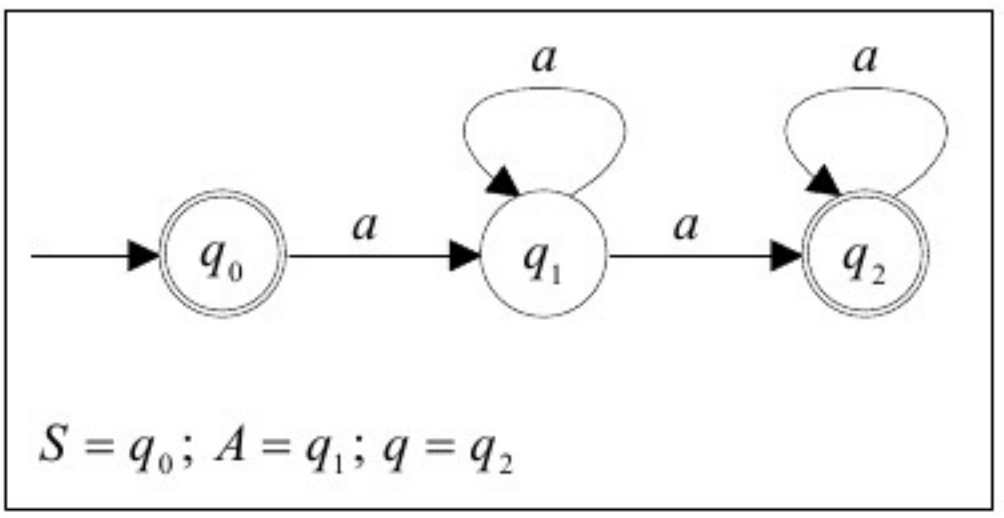


Figura 7.7

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

che può essere comunque trasformato nell'automa deterministico equivalente in modo molto più semplice:

$$\delta'(\{q_0\},a) = \delta(q_0,a) = \{q_1\}$$
$$\delta'(\{q_1\},a) = \delta(q_1,a) = \{q_1,q_2\}$$
$$\delta'(\{q_1,q_2\},a) = \delta(q_1,a) \cup \delta(q_2,a) = \{q_1,q_2\}$$

per cui l'automa equivalente è (Figura 7.8):

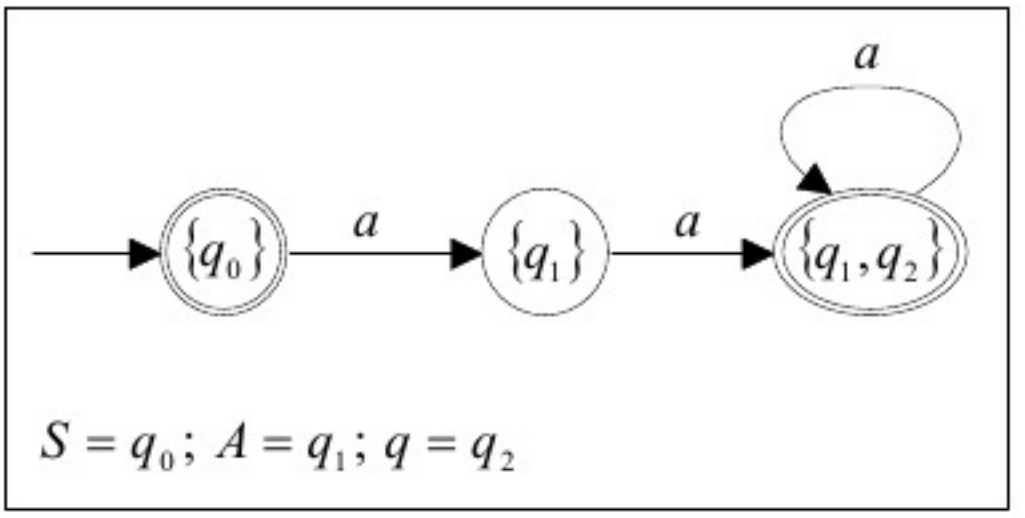


Figura 7.8

Un terzo modo di risolvere l'esercizio:

$$\{a\}^* - \{a\} = \overline{\{a\}}$$

L'automa che riconosce $L = \{a\}$ è dato in Figura 7.9.

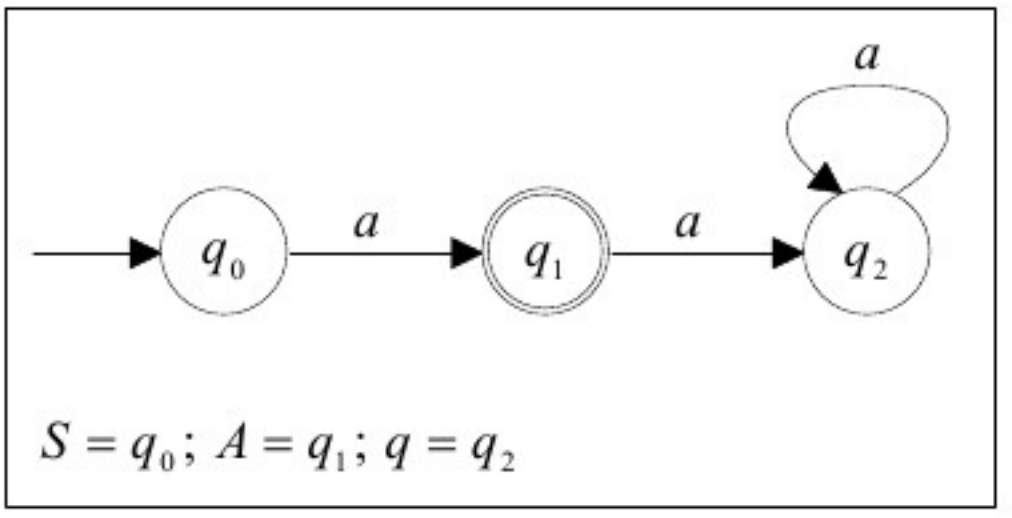


Figura 7.9

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

L'automa che riconosce $\overline{L} = \overline{\{a\}}$ si costruisce dall'automa che riconosce L considerando come stati di accettazione il complementare di F (Figura 7.10).

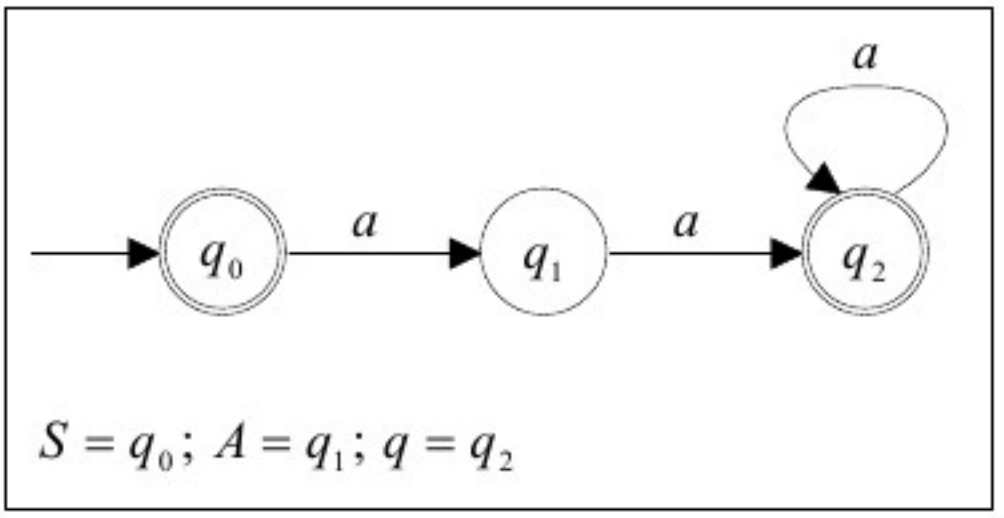


Figura 7.10

Cosa accade se consideriamo l'automa che riconosce $L = \{a\}$ con funzione di transizione δ definita parzialmente?

<u>Esercizio 7.4</u>

Con riferimento all'Esercizio 6.1, determinare una grammatica lineare destra che genera il linguaggio:

$$L = \left\{ w \,\middle|\, \begin{array}{r} w \in \{a,b\}^*,\ w \text{ ha un numero pari di } a \text{ ed} \\ \text{un numero dispari di } b \end{array} \right\}$$

Una grammatica lineare destra che genera il linguaggio L può essere costruita a partire dall'automa accettore a stati finiti determinato nell'Esercizio 6.1 utilizzando l'Algoritmo 7.2, come segue:

$$G = (X, V, S, P)$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

dove:

1) $X = \{a,b\}$;
2) $V = Q = \{q_0, q_1, q_2, q_3\}$;
3) $S = q_0$;
4) $P = \{q_0 \to aq_2 \mid bq_1 \mid b,\ q_1 \to aq_3 \mid bq_0,\ q_2 \to aq_0 \mid bq_3,\ q_3 \to aq_1 \mid bq_2 \mid a\}$.

Esercizio 7.5

Sia L il linguaggio denotato dalla seguente espressione regolare:

$$ab(bb)^* c$$

1) Trovare un automa a stati finiti che riconosce L.

2) Trasformare l'automa non deterministico al punto 1) in un automa deterministico equivalente.

1) Determiniamo una grammatica lineare destra che genera L.

Poiché:

$$\begin{aligned} S\left(ab(bb)^* c\right) &= S(a) \cdot S(b) \cdot S\left((bb)^*\right) \cdot S(c) = \\ &= \{a\} \cdot \{b\} \cdot \{bb\}^* \cdot \{c\} = \\ &= \{ab\} \cdot \{bb\}^* \cdot \{c\} \end{aligned}$$

possiamo determinare una grammatica che genera L sfruttando le proprietà di chiusura dei linguaggi di tipo '3'.

Una grammatica che genera $\{ab\}$ è:

$$G_1 = (X, V_1, S_1, P_1)$$

dove:

- $X = \{a,b\}$;

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

- $V_1 = \{S_1, A\}$;
- $P_1 = \{S_1 \to aA,\ A \to b\}$.

Una grammatica G_3 che genera $\{bb\}^*$ si ottiene per iterazione dalla grammatica G_2 che genera $\{bb\}$:

$$G_2 = (X, V_2, S_2, P_2)$$

dove:

- $X = \{b\}$;
- $V_2 = \{S_2, B\}$;
- $P_2 = \{S_2 \to bB,\ B \to b\}$.

$$G_3 = (X, V_3, S_3, P_3)$$

dove:

- $X = \{b\}$;
- $V_3 = V_2 \cup \{S_3\} = \{S_2, B, S_3\}$;
- $P_3 = \{S_3 \to bB \mid \lambda,\ B \to bS_3 \mid b,\ \cancel{S_2 \to bB}\}$.[13]

Una grammatica che genera $\{c\}$ è:

$$G_4 = (X, V_4, S_4, P_4)$$

dove:

- $X = \{c\}$;
- $V_4 = S_4$;
- $P_4 = \{S_4 \to c\}$.

---

[13] La produzione $S_2 \to bB$ viene cancellata in quanto S_2 è un nonterminale inutile.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Una grammatica che genera $\{ab\} \cdot \{bb\}^*$ è:

$$G_5 = (X, V_5, S_5, P_5)$$

dove:

- $X = \{a,b\}$;
- $V_5 = V_1 \cup V_3 = \{S_1, A, S_3, B\}$;
- $S_5 = S_1$;
- $P_5 = \{S_1 \to aA,\ A \to bS_3,\ S_3 \to bB \mid \lambda,\ B \to bS_3 \mid \mathrm{b}\}$.

Infine, una grammatica che genera il linguaggio $L = \{ab\} \cdot \{bb\}^* \cdot \{c\}$ è:

$$G = (X, V, S, P)$$

dove:

- $X = \{a,b,c\}$;
- $V = V_4 \cup V_5 = \{S_4, S_1, A, S_3, B\}$;
- $S = S_1$;
- $P = \{S_1 \to aA,\ B \to bS_3 \mid bS_4,\ A \to bS_3 \mid bS_4,\ S_3 \to bB,$
  $S_4 \to c\}$

.

L'automa che riconosce $L(G)$ si costruisce come segue, in base all'Algoritmo 7.1:

$$M = (Q, \delta, q_0, F) \text{ con alfabeto di ingresso } X$$

- $Q = V \cup \{q\},\ q \notin V$;
- $q_0 = S_1$;

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

- $F = \{q\}$;

e il diagramma di transizione è dato in Figura 7.11.

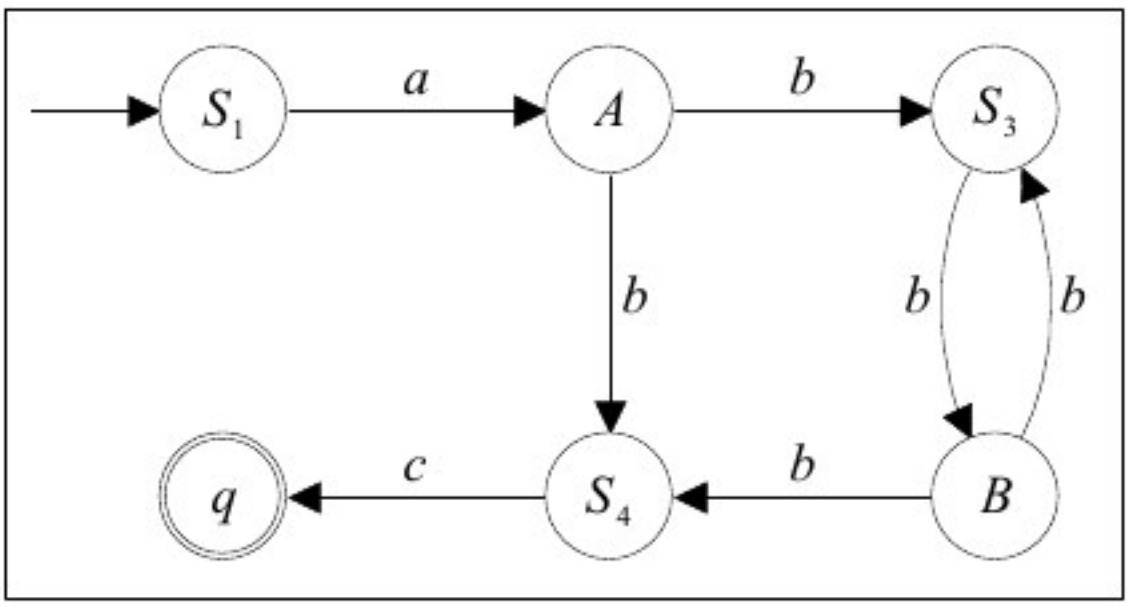


Figura 7.11

2) L'automa deterministico M' equivalente ad M si costruisce come segue, in base all'Algoritmo 6.1:

$$M' = (Q', \delta', q_0', F')$$

dove:

- $Q' = 2^Q$;
- $q_0' = \{S_1\}$;
- $F' = \{p \subseteq Q \mid p \cap F \neq \varnothing\}$ con $|F'| = 32$;
- δ' è definita come segue:

$$\delta' : Q' \times X \to Q'$$

  - $\delta'(\{S_1\}, a) = \delta(S_1, a) = \{A\}$;
  - $\delta'(\{S_1\}, b) = \delta(S_1, b) =$ non è definita;
  - $\delta'(\{S_1\}, c) = \delta(S_1, c) =$ non è definita;

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

- $\delta'(\{A\},a) = \delta(A,a) =$ non è definita;
- $\delta'(\{A\},b) = \delta(A,b) = \{S_3, S_4\}$;
- $\delta'(\{A\},c) = \delta(A,c) =$ non è definita;
- $\delta'(\{S_3,S_4\},a) = \delta(S_3,a) \cup \delta(S_4,a) =$ non è definita;
- $\delta'(\{S_3,S_4\},b) = \delta(S_3,b) \cup \delta(S_4,b) = \{B\}$;
- $\delta'(\{S_3,S_4\},c) = \delta(S_3,c) \cup \delta(S_4,c) = \{q\}$;
- $\delta'(\{B\},a) = \delta(B,a) =$ non è definita;
- $\delta'(\{B\},b) = \delta(B,b) = \{S_3, S_4\}$;
- $\delta'(\{B\},c) = \delta(B,c) =$ non è definita;

e il diagramma degli stati è dato in Figura 7.12.

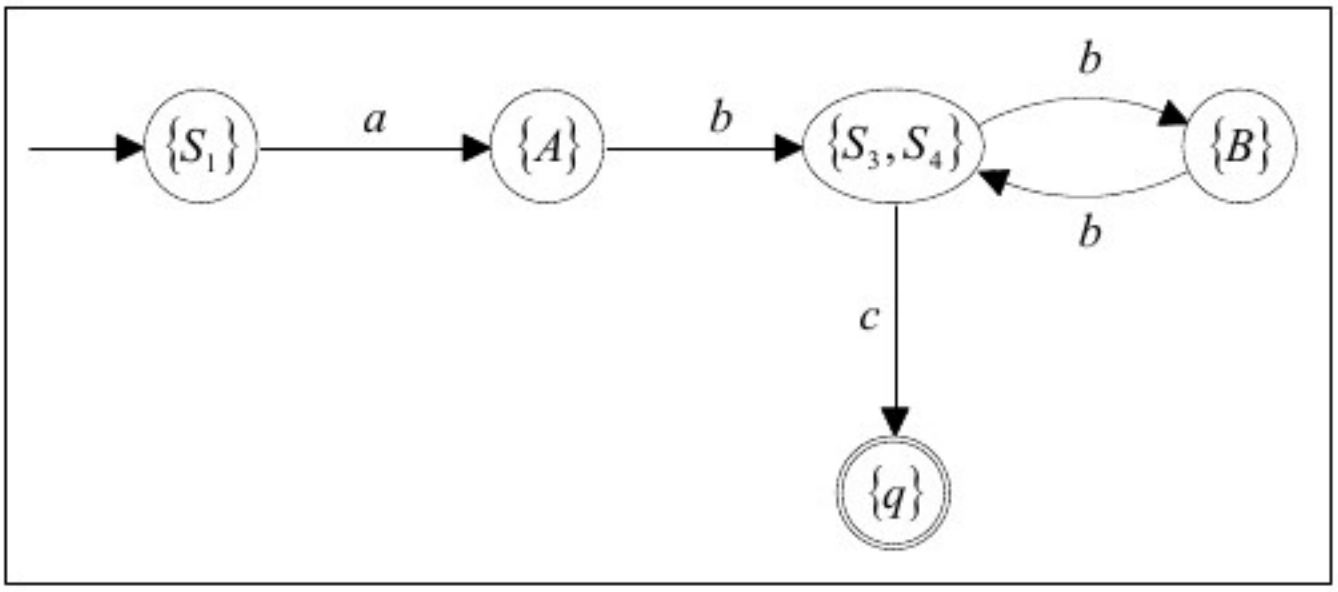


Figura 7.12

Esercizio 7.6

Si consideri la seguente grammatica lineare destra:

$$G = (X, V, S, P)$$

con

$$X = \{a,b\} \qquad V = \{S,B\}$$
$$P = \{S \to aB,\ B \to aB \mid bS \mid a\}$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Determinare un automa deterministico M tale che:

$$L(G) = T(M)$$

Facciamo riferimento all'algoritmo per la costruzione dell'automa accettore a stati finiti non deterministico equivalente ad una grammatica lineare destra (Algoritmo 7.1).

$$M = (Q,\ \delta,\ q_0,\ F) \text{ con}$$

1) $X = \{a, b\}$ alfabeto di ingresso;

2) $Q = \{S, B, q\}$;

3) $q_0 = S$;

4) $F = \{q\}$;

δ è definita come segue:

5) $\delta : Q \times X \to 2^Q$

5.a) $S \to aB$ dà origine a : $B \in \delta(S, a)$;

$B \to aB$ dà origine a : $B \in \delta(B, a)$;

$B \to bS$ dà origine a : $S \in \delta(B, b)$;

5.b) $B \to a$ dà origine a : $q \in \delta(B, a)$.

Per cui, la tavola di transizione che riassume la definizione della funzione δ è la seguente:

δ	S	B	q
a	$\{B\}$	$\{B, q\}$	–
b	–	$\{S\}$	–

Il grafo degli stati di M è dato in Figura 7.13.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

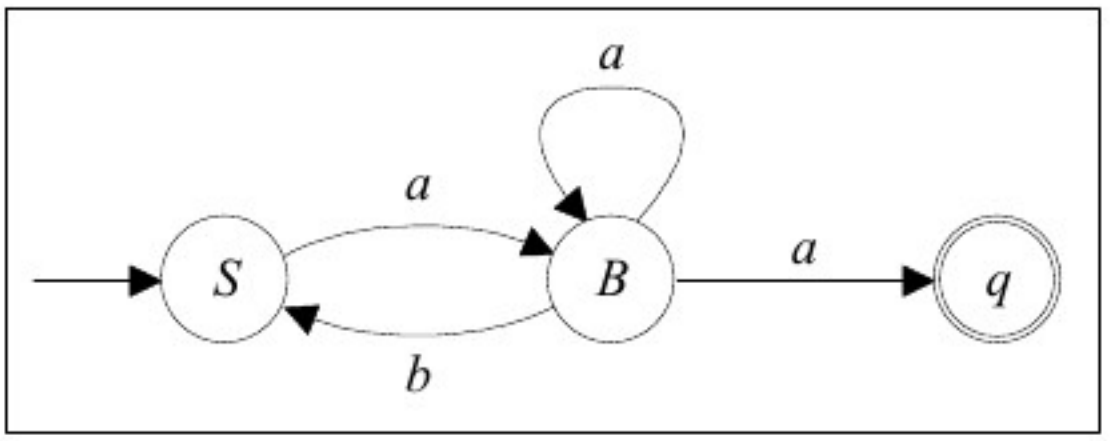


Figura 7.13

L'automa accettore a stati finiti deterministico M' equivalente ad M si costruisce come segue (Algoritmo 6.1):

$$M' = (Q', \delta', q_0', F') \qquad X = \{a, b\} \text{ alfabeto di ingresso}$$

con:

i) $Q' = 2^Q = 2^{\{S,B,q\}}$;

ii) $q_0' = \{S\}$;

iii) $F' = \{\{q\}, \{q,S\}, \{q,B\}, \{q,S,B\}\}$;

iv) δ' è definita come segue:

$$\delta' : Q' \times X \to Q'$$

- $\delta'(\{S\}, a) = \delta(S, a) = \{B\}$;
- $\delta'(\{S\}, b) = \delta(S, b) =$ non definita;
- $\delta'(\{B\}, a) = \delta(B, a) = \{B, q\}$;
- $\delta'(\{B\}, b) = \delta(B, b) = \{S\}$;
- $\delta'(\{B, q\}, a) = \delta(B, a) \cup \delta(q, a) = \{B, q\}$;
- $\delta'(\{B, q\}, b) = \delta(B, b) \cup \delta(q, b) = \{S\}$.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

La tavola di transizione che riassume la definizione della funzione δ' è la seguente:

δ'	$\{S\}$	$\{B\}$	$\{B,q\}$
a	$\{B\}$	$\{B,q\}$	$\{B,q\}$
b	$-$	$\{S\}$	$\{S\}$

Il grafo degli stati è indicato in Figura 7.14.

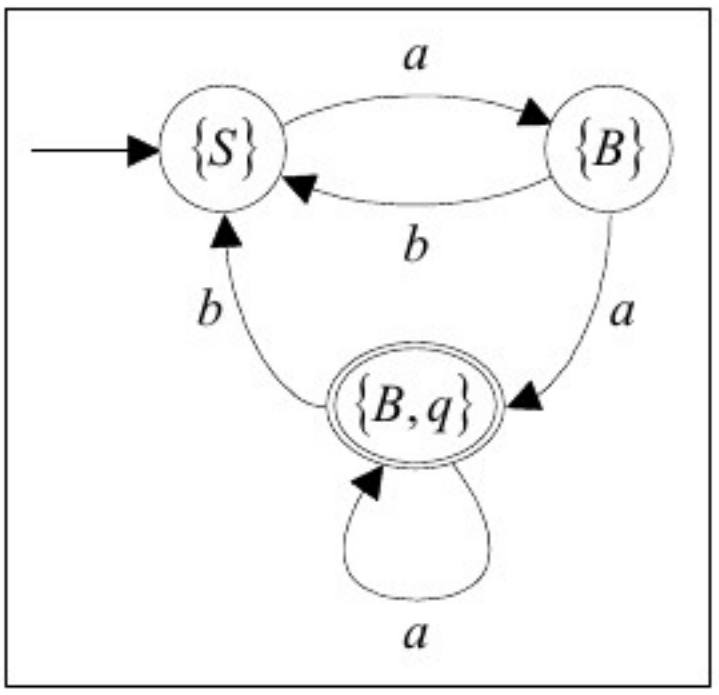


Figura 7.14

<u>Esercizio 7.7</u>

Con riferimento all'Esercizio 6.2, determinare una grammatica lineare destra che genera il linguaggio:

$$L = \left\{ w \;\middle|\; \begin{array}{l} w \in \{a,b\}^*,\ \ w \neq \alpha aa\beta,\ \ \alpha,\beta \in \{a,b\}^* \\ (w \text{ non contiene due } a \text{ consecutive}) \end{array} \right\}$$

Una grammatica lineare destra che genera il linguaggio L può essere costruita a partire dall'automa accettore a stati finiti

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

determinato nell'Esercizio 6.2, utilizzando l'Algoritmo 7.2, come segue:

$$G = (X, V, S, P)$$

con:

- $X = \{a, b\}$;
- $V = Q = \{q_0, q_1, q_2\}$;
- $S = q_0$;
- $P = \{q \to xq' \mid q' \in \delta(q, x)\} \cup \{q \to x \mid \delta(q, x) \in F\}$
  $= \{q_0 \to bq_0 \mid aq_1 \mid a \mid b \mid \lambda,\ q_1 \to bq_0 \mid \cancel{aq_2} \mid b,\ q_2 \to \cancel{bq_2} \mid \cancel{aq_2}\} =$
  $= \{q_0 \to bq_0 \mid aq_1 \mid a \mid b \mid \lambda,\ q_1 \to bq_0 \mid b\}$.

Esercizio 7.8

Si utilizzi il Pumping Lemma per i linguaggi regolari (Teorema 7.2) per dimostrare che i seguenti linguaggi non sono regolari:

1) $L = \{a^k b^k \mid k > 0\}$

2) $L = \{a^n b^m c^k \mid n > k,\ m, n, k \geq 1\}$

1) Supponiamo, per assurdo, che L sia regolare. Questo implica che $\exists\ M = (Q, \delta, q_0, F) :\ T(M) = L$.

   Supponiamo che $|Q| = n$ con $n > 0$. Consideriamo la seguente parola di L:

$$a^n b^n$$

$$\overbrace{aaaa\ldots a}^{n}\, bbbb\ldots b$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

L'automa M parte dallo stato q_0 e legge una a per volta. Dopo aver letto la prima a si porta in q_1, dopo la seconda a in q_2, ..., dopo la n-esima a si porta in q_n. Abbiamo dunque $n+1$ stati q_0, $q_1,\ldots q_n$, in cui M transita.

Si ha:

$$q_0 \xrightarrow{a} q_1$$
$$q_1 \xrightarrow{a} q_2$$
$$\ldots$$
$$q_{n-1} \xrightarrow{a} q_n$$

Poiché M ha solo n stati, due tra gli stati q_0, $q_1,\ldots q_n$ devono coincidere.

Siano, ad esempio, q_i e q_j gli stati coincidenti:

$$q_i = q_j, \qquad i < j$$

Si ha dunque un ciclo nel grafo degli stati di M (Figura 7.15).

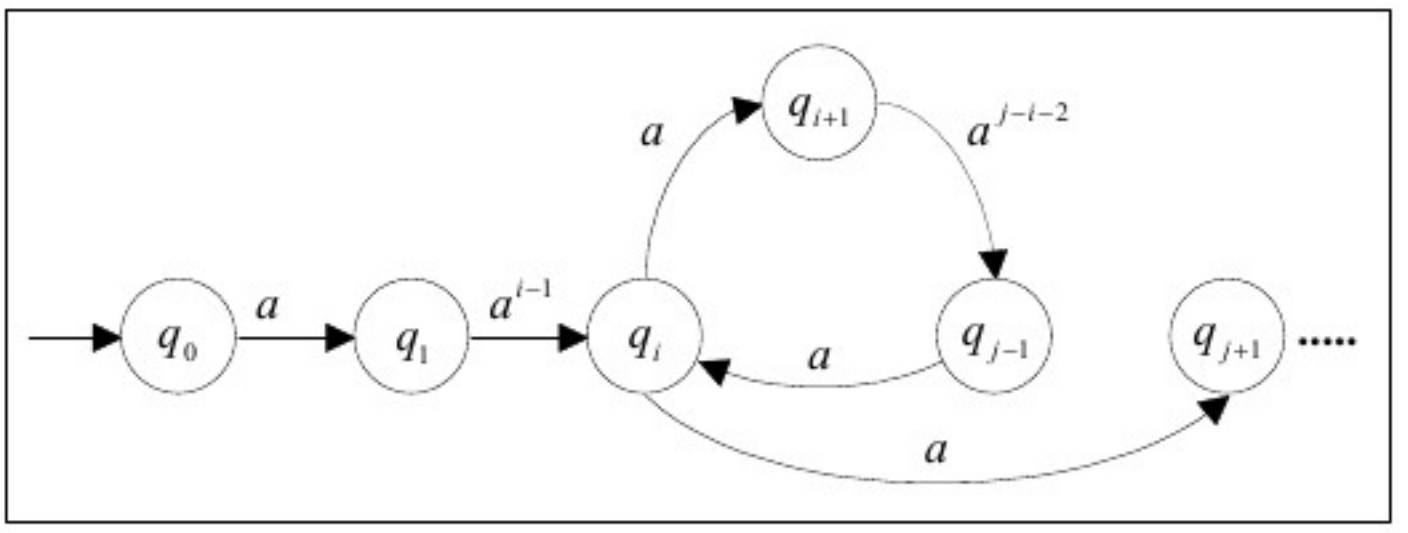


Figura 7.15

Tale ciclo ha lunghezza $j-i$.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Poiché esiste tale ciclo, possiamo aggiungere indefinitamente un numero arbitrario di a nella parola in ingresso, senza modificare l'esito del riconoscimento (vale a dire che la parola viene comunque accettata) se tale numero è un multiplo di $j-i$.

Dunque anche $a^{n+k(j-i)}b^n \in T(M),\ k = 0,1,2,\ldots$.

Ma ciò è in contraddizione con l'ipotesi $T(M) = L$. Dunque L non è regolare.

2) 1° Modo.

Supponiamo per assurdo che:

$$L = \{a^n b^m c^k \mid n > k,\ m,n,k \geq 1\}$$

sia regolare.

Allora esiste un automa accettore a stati finiti $M = (Q, \delta, q_0, F)$ tale che $L = T(M)$. Supponiamo $|Q| = n$ con $n > 0$. Consideriamo le sottostringhe:

$$a^2, a^3, \ldots, a^n, a^{n+1}, a^{n+2}$$

della stringa $a^{n+2}bc$, e gli stati in cui M si porta quando ha in ingresso una delle suddette sottostringhe.

Si ha:

$$a^2 \to q_{a^2}$$
$$a^3 \to q_{a^3}$$
$$\dots$$
$$a^n \to q_{a^n}$$
$$a^{n+1} \to q_{a^{n+1}}$$
$$a^{n+2} \to q_{a^{n+2}}$$

Se $q_{a^2}, \dots q_{a^{n+1}}, q_{a^{n+2}}$ fossero tutti distinti tra loro, avremmo $n+1$ stati nell'automa (contro l'ipotesi fatta).

Dunque almeno due stati devono coincidere. Siano a^i e a^j le stringhe per cui M si porta in uno stesso stato e supponiamo $i < j$. Si ha dunque:

$$q_{a^i} = q_{a^j} = q$$

Consideriamo ora la parola di L: $a^j bc^{j-1}$. Poiché $L = T(M)$, tale parola deve essere accettata da M. Ma allora la parola $a^i bc^{j-1}$, $i < j$, viene accettata da M.

Ma questo è un assurdo in quanto $a^i bc^{j-1} \notin L$, $i < j$.

2° Modo.

Supponiamo, per assurdo, che L sia regolare. Questo implica che $\exists\ M = (Q, \delta, q_0, F) :\ T(M) = L$.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Supponiamo che $|Q| = n$ con $n > 0$. Consideriamo la seguente parola di L:

$$a^{n+1}bc^n$$

$$aaaa...ab\overbrace{cccc...c}^{n}$$

L'automa M parte dallo stato q_0, legge le $n+1$ a e si porta in q, poi legge l'unica b e si porta in q^1. Di seguito, legge la prima c e si porta in q^2, legge la seconda c e si porta in q^3,…, legge la n-esima c e si porta in q^{n+1}. Abbiamo dunque $n+1$ stati q^1, q^2, …, q^{n+1} in cui M transita. Poiché M ha solo n stati, due tra gli stati q^1, q^2, …, q^{n+1} devono coincidere.

Siano q^i e q^j tali stati:

$$q^i = q^j = q, \quad i < j$$

Nel grafo degli stati di M esiste dunque un ciclo di lunghezza $j-i$. L'esistenza di tale ciclo nel grafo di M ci permette di aggiungere altre c nella parola in ingresso, ottenendo ancora parole accettate da M se il numero di c aggiunte è un multiplo di $j-i$. Dunque anche: $a^{n+1}bc^{n+k(j-i)} \in T(M)$. Ma $a^{n+1}bc^{n+k(j-i)} \notin L, \ k = 1,2,...$.

Da cui l'assurdo. Dunque L non è regolare.

<u>Esercizio 7.9</u>

Dimostrare che

$$L = \{a^n b^m c^k \mid m > k, \ n,m,k \geq 1\};$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

non è un linguaggio regolare.

Supponiamo per assurdo che L sia un linguaggio regolare.
Allora esiste un automa accettore a stati finiti $M = (Q, \delta, q_0, F)$ di alfabeto di ingresso $X = \{a, b, c\}$ tale che $L = T(M)$. Supponiamo che $|Q| = n$ con $n > 0$. Consideriamo le seguenti sottostringhe:

$$ab^2, ab^3, ..., ab^{n+1}, ab^{n+2}$$

della parola $ab^{n+2}c^{n+1}$ e gli stati in cui M si porta quando ha in ingresso una delle suddette sottostringhe:

$$ab^2 \to q_{ab^2}$$
$$ab^3 \to q_{ab^3}$$
$$...$$
$$ab^n \to q_{ab^n}$$
$$ab^{n+1} \to q_{ab^{n+1}}$$
$$ab^{n+2} \to q_{ab^{n+2}}$$

Se $q_{ab^2}, q_{ab^3}, ..., q_{ab^{n+2}}$ fossero tutti stati distinti di M, avremmo $n+1$ stati nell'automa (contro l'ipotesi che $|Q| = n$). Dunque almeno due stati devono coincidere.
Siano ab^i e ab^j le stringhe per cui M si porta in uno stesso stato e supponiamo $2 \le i < j \le n+2$. Dunque si ha:

$$q_{ab^i} = q_{ab^j} = q$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Consideriamo ora la stringa:

$$ab^j c^{j-1}$$

Evidentemente:

$$ab^j c^{j-1} \in L$$

Poiché $L = T(M)$, tale stringa deve essere accettata da M.

Ma allora anche la stringa $ab^i c^{j-1}$, $i < j$, viene accettata da M (lo stato in cui M si porta quando ha in ingresso $ab^j c^{j-1}$ e $ab^i c^{j-1}$ è lo stesso).

Siamo dunque giunti ad una contraddizione:

$$ab^i c^{j-1} \in T(M) \qquad \text{e} \qquad ab^i c^{j-1} \notin L$$

Ne consegue che L non è regolare.

7.6 Esercizi proposti

<u>Esercizio 1</u>

Sia $M = (Q, \delta, q_0, F)$ un automa a stati finiti non deterministico di alfabeto $X = \{a, b\}$,

ove:

$Q = \{ q_0, q_1, q_2 \}$,

$\delta(q_0, a) = \{q_0, q_1, q_2 \}$ $\qquad$ $\delta(q_0, b) = -$

$\delta(q_1, a) = -$ $\qquad$ $\delta(q_1, b) = \{q_0\}$

$\delta(q_2, a) = -$ $\qquad$ $\delta(q_2, b) = -$

ed $F = \{q_2\}$.

Descrivere il linguaggio $T(M)$ accettato da M.

Costruire un automa a stati finiti deterministico equivalente ad M.

Determinare una grammatica lineare destra che genera $T(M)$.

Determinare una espressione regolare che denota $T(M)$.

Esercizio 2

Sia $M = (Q, \delta, q_0, F)$ un automa a stati finiti nondeterministico di alfabeto $X = \{0, 1\}$, ove:

$Q = \{ q_0, q_1, q_2, q_3 \}$,

$\delta(q_0, 0) = \{q_0\}$ $\quad$ $\delta(q_0, 1) = \{q_1, q_2 \}$

$\delta(q_1, 0) = \{q_3\}$ $\quad$ $\delta(q_1, 1) = \{q_0, q_1 \}$

$\delta(q_2, 0) = \{q_0, q_1 \}$ $\quad$ $\delta(q_2, 1) = \{q_0, q_2 \}$

$\delta(q_3, 0) = \{q_3 \}$ $\quad$ $\delta(q_3, 1) = \{q_3 \}$

ed $F = \{q_2\}$

Descrivere il linguaggio accettato $T(M)$.

Costruire un automa a stati finiti deterministico che accetta $T(M)$.

Determinare una grammatica di Tipo 3 che genera $T(M)$.

Esercizio 3

Sia dato il seguente linguaggio:

$L = \{ \alpha 00 \beta : \alpha, \beta \in \{0, 1\}^* \} \cup \{ \alpha 11 \beta : \alpha, \beta \in \{0, 1\}^* \}$

Costruire un automa a stati finiti deterministico che accetta L.

Esercizio 4

Siano dati i seguenti linguaggi:

d) $\{ w \in \{b, s, u, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}^* : w$ non contiene come sottostringa $ubs633369 \}$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

e) $\{ w \in \{p, -, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}^* : w$ non contiene come sottostringa la parola p2-1816 $\}$

f) $\{ w \in \{a, b, c\}^* : w$ non contiene come sottostringa la parola $babc$ $\}$

g) $\{ \alpha 00\beta : \alpha, \beta \in \{0, 1\}^* \} \cap \{ \alpha 11\beta : \alpha, \beta \in \{0, 1\}^* \}$

Costruire un automa a stati finiti deterministico che accetta L.

Determinare una grammatica lineare destra che genera L.

Esercizio 5

Sia dato il seguente linguaggio:

$L = \{ \alpha 00\beta : \alpha, \beta \in \{0, 1\}^* \} \cup \{ w \in \{0, 1\}^* : w$ non contiene come sottostringa la parola 11 $\}$

Costruire un automa a stati finiti che accetta L.

Costruire un automa a stati finiti deterministico che accetta L.

Determinare una grammatica lineare destra che genera L.

Esercizio 6

Sia L il linguaggio denotato dalla seguente espressione regolare su $X = \{0, 1\}$:

$0\,(0+1)^* + 0^*1$

Costruire un automa a stati finiti deterministico che accetta L.

Determinare una grammatica lineare destra che genera L.

Esercizio 7

Sia data la seguente grammatica lineare destra:

$G = (X, V, S, P)$

ove $X = \{0, 1\}$

$V = \{S, A, B\}$

$P = \{ \quad S \to 0S \mid 1B \mid 1A,$

$A \to 0B \mid 0,$

$B \to 1A \qquad \}$

Determinare un'espressione regolare che denota $L(G)$.

Costruire un automa a stati finiti che riconosce $L(G)$.

Esercizio 8

Sia data la seguente grammatica lineare destra:

$G = (X, V, S, P)$

ove $X = \{a, b, c\}$

$V = \{S, A, B, C\}$

$P = \{ \quad S \to aS \mid bA \mid cS \mid b \mid \lambda,$

$A \to aB \mid bA \mid cS \mid a \mid b \mid c,$

$B \to aS \mid bC \mid cS \mid a \mid b \mid c,$

$C \to aB \mid bA \mid a \mid b \qquad \}$

Determinare un'espressione regolare che denota il linguaggio $L(G)$.

Costruire un automa a stati finiti deterministico che accetta il linguaggio $L(G)$.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Esercizio 9

Stabilire quali dei seguenti linguaggi sono regolari e giustificare formalmente la risposta:

h) l'insieme delle stringhe $w \in \{ 0, 1\}^*$ tali che il numero di occorrenze in di 0 in w è il triplo delle occorrenze di 1 in w;

i) l'insieme delle stringhe $w \in \{ a, b\}^*$ la cui lunghezza è divisibile per 5.

j) l'insieme delle parole $w \in \{ 0, 1\}^*$ tali che il numero di occorrenze di 1 in w è diverso da quello delle occorrenze di 0 in w;

k) l'insieme delle parole $w \in \{ a, b\}^*$ del tipo $a^i b^j$, con $0 \leq i \leq j \leq 2i$.

l) il linguaggio generato dalla grammatica $G = (X, V, S, P)$, con: $X = \{$ x, and, or, not $\}$, $V = \{$E, T, F$\}$, $S =$ E, $P = \{$E $\rightarrow$ E or T | T, T $\rightarrow$ T and F | F, F $\rightarrow$ not F | x $\}$

m) $\{ w: w \in \{ a, b \}^*, |w| = 5n, n \geq 0 \} \cap \{ w: w \in \{ a, b \}^*, |w| = 3n, n \geq 0 \}$

n) $\{ w: w \in \{ a, b, c \}^*$, il numero di a in w è uguale al numero di b in $w \} \cap \{ w: w \in \{ a, b, c \}^*$, il numero di a in w è diverso dal numero di b in $w \}$

o) $\{ ww': w, w' \in \{ a, b \}^*, |w| = |w'| \} \cup \{ ww': w, w' \in \{ a, b \}^*, |w| \neq |w'| \}$

p) $\{ ww': w, w' \in \{ a, b \}^*, |w| = |w'| \} \cap \{ ww': w, w' \in \{ a, b \}^*, |w| \neq |w'|$

q) $\{ w: w \in \{ a, b \}^*, |w| = 4n, n \geq 0 \} \cup \{ w: w \in \{ a, b \}^*, |w| = 2n, n \geq 0 \}$

r) $\{ a^m b^n : m, n$ interi non negativi, $n < m$ OR $m < n \}$

Esercizio 10

Sia dato il linguaggio:

$L = \{ w \in \{a, b, c \}^* : w$ non contiene come sottostringa la parola $bbc \}$

Costruire un automa a stati finiti deterministico che accetta L.

Determinare una grammatica lineare destra che genera L.

Determinare un'espressione regolare che denota L.

Esercizio 11

Sia dato il seguente linguaggio:

$L = \{ w \in \{a, b\}^* : |w| = 3k, k > 0$ AND w non contiene come sottostringa la parola $aa \}$

Costruire un automa a stati finiti che accetta L.

Determinare una grammatica lineare destra che genera L.

Esercizio 12

Siano L_1 ed L_2 i linguaggi denotati dalle espressioni regolari $(ab)^*$ e $a(ba + a)^*$, rispettivamente.

Costruire un automa a stati finiti che accetta il linguaggio $L = L_1 \cup L_2$.

titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Esercizio 13

Siano dati il linguaggio L_1 denotato dalla espressione regolare $(a+b)^* \cdot b$ ed il linguaggio L_2 denotato dalla espressione regolare c^*. Si determini una grammatica G tale che $L(G) = L_1 \cup L_2$.

Esercizio 14

Siano dati i linguaggi L_1 ed L_2 denotati dalle espressioni regolari $(0\ \ (0+1)^*) + 0^*1$ e 2^*, rispettivamente.

Determinare una grammatica G tale che $\mathrm{L}(G) = L_1 \cup L_2$.

234 - LINGUAGGI REGOLARI, ESPRESSIONI REGOLARI E TEOREMA DI KLEENE

8. Automi a pila e grammatiche libere

8.1 Automi a pila

Problema Può un automa a stati finiti (FSA) riconoscere un linguaggio non contestuale?
NO!

Esempio 8.1
Si consideri il linguaggio:

$$L_1 = \{a^n b^n \mid n \in \aleph\}$$

e si supponga che l'automa a stati finiti M_1 che riconosce L_1 abbia n stati. Cioè:

$$M_1 = (Q, \delta, q_0, F), \ |Q| = n$$

per cui si ha:

$$T(M_1) = L_1$$

Si consideri la parola: $a^n b^n \in L_1$. Per riconoscere $a^n b^n$, M_1 deve transitare per $n+1$ stati. Poiché gli stati sono n, i vertici del cammino non sono tutti distinti, ma si hanno dei cicli (Figura 8.1). Se il ciclo ha lunghezza $k < n$, allora la macchina riconosce anche $a^{n+k} b^n$, che non è una parola di L_1.
M_1 non ha "stati" a sufficienza per riconoscere L_1. Gli stati sono l'unico modo di *memorizzare*, di *ricordare il passato* per un automa a stati finiti.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

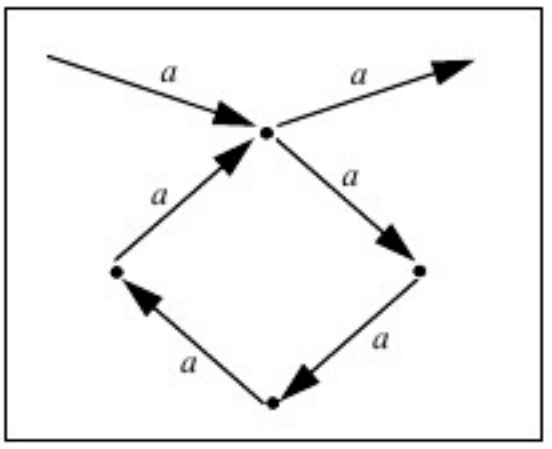


Figura 8.1

Gli automi a stati finiti hanno memoria finita.

Molti linguaggi, invece, hanno bisogno di memorie potenzialmente infinite.

Gli automi a stati finiti falliscono laddove la procedura di riconoscimento di un linguaggio non stabilisce un limite superiore al numero di stati per il calcolo (è il caso di L_1, perché non si conosce a priori il numero massimo di a in ingresso).

Per estendere la memoria di un automa, si possono pensare due soluzioni:

1) aumentare il numero dei suoi stati (interni);
2) dotare l'automa di una memoria esterna (*memoria ausiliaria*).

La soluzione 1) non risolve il problema di riconoscere linguaggi come L_1.

La soluzione 2) definisce un nuovo tipo di automa riconoscitore.

Lo schema di un automa riconoscitore, nella sua forma più generale, è illustrato in Figura 8.2.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

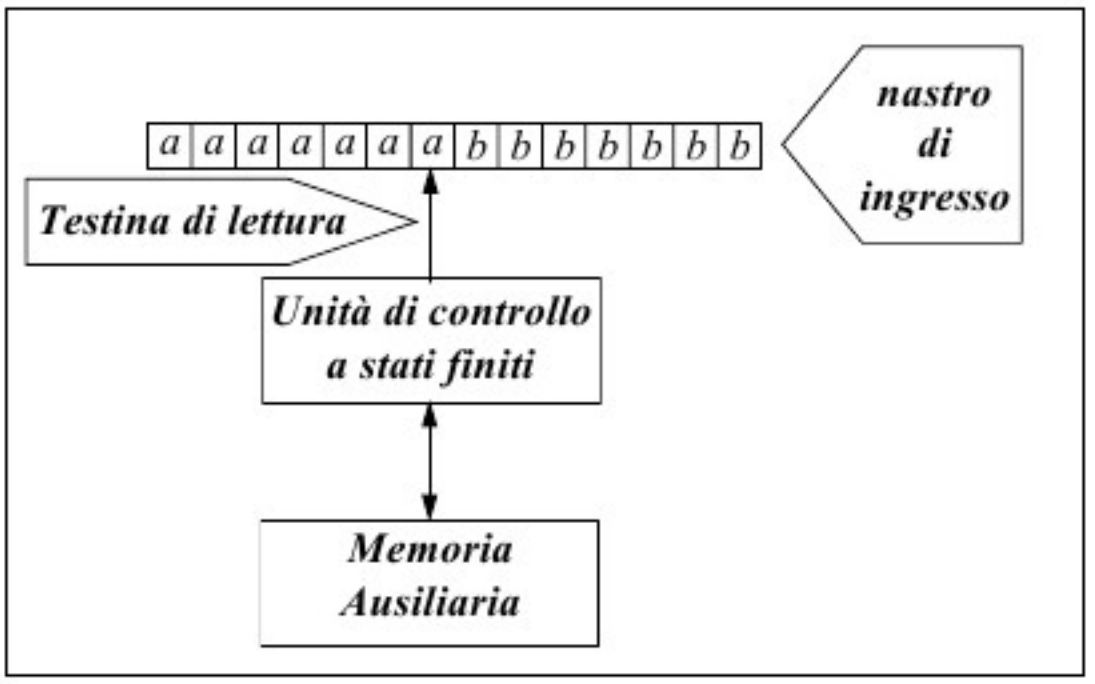


Figura 8.2

Le tre unità che compongono il dispositivo sono (Figura 8.3):

1) il *nastro di ingresso*, suddiviso in caselle adiacenti, ciascuna contenente un simbolo dell'alfabeto di ingresso.
   L'automa legge il carattere sottostante la *testina di lettura*, che può essere fatta scorrere lungo il nastro in entrambe le direzioni;
2) la *memoria ausiliaria* è un'unità, opportunamente organizzata, in generale di capacità illimitata, in grado di accumulare informazioni composte da simboli di un *alfabeto di memoria* o *di lavoro*.
   Tali informazioni, che costituiscono il risultato di una elaborazione parziale della stringa, vengono successivamente fornite quando richieste.
   La memoria è quindi utilizzabile sia in lettura sia in scrittura.
   Se la memoria ausiliaria è organizzata *a pila* (*stack*), l'automa prende il nome di *automa a pila* o *automa push-down* (*PDA*);

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

3) l'*unità di controllo a stati finiti* è il cuore dell'automa riconoscitore. In essa è fisicamente *inciso* il programma invariante che costituisce l'algoritmo di riconoscimento.

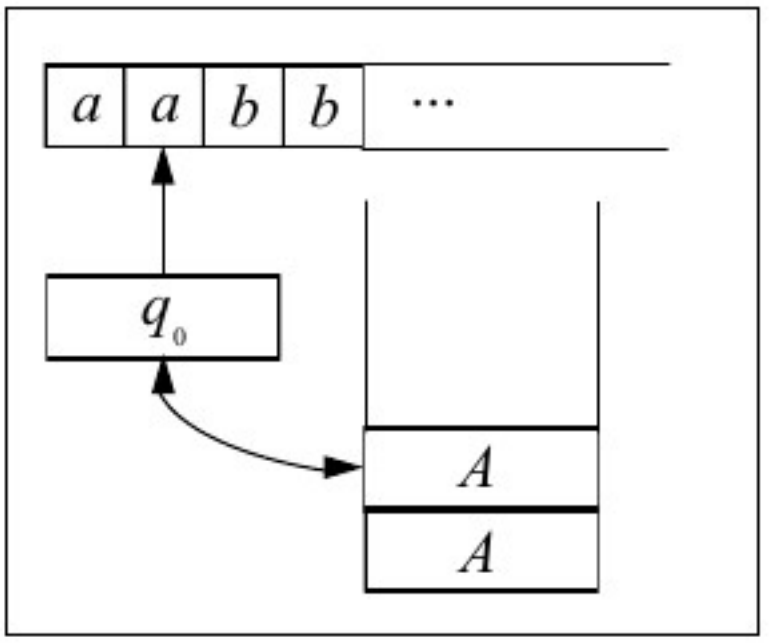


Figura 8.3

Formalmente, possiamo quindi definire un automa a pila non deterministico come segue:

<u>Definizione 8.1</u> (Automa a pila non deterministico o automa push-down o PDA)

Un *automa a pila* (PDA) non deterministico è una settupla:

$$M = (Q, X, \Gamma, \delta, q_0, Z_0, F)$$

dove:

- Q è un insieme finito non vuoto di *stati*;
- X è un insieme finito non vuoto di simboli, detto *alfabeto di ingresso*;

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

- Γ è un insieme finito non vuoto di simboli, detto *alfabeto di pila*;
- δ è una funzione

$$\delta : Q \times (X \cup \{\lambda\}) \times \Gamma \rightarrow 2^{Q \times \Gamma^*}$$

detta *funzione di transizione*.

δ può anche essere vista come l'insieme delle transizioni $< q, x, Z, q', \sigma >$, che possiamo anche scrivere nella notazione $(q', \sigma) \in \delta(q, x, Z)$;

- $q_0 \in Q$ è lo *stato iniziale*;
- $Z_0 \in \Gamma$ è il *simbolo inizialmente posto nella pila*;
- $F \subseteq Q$ è *l'insieme degli stati finali*. ■

Il funzionamento di un automa push-down può essere descritto anche attraverso una sequenza di *descrizioni istantanee* (ID).

Definizione 8.2 (Descrizione istantanea o ID)

Una *descrizione istantanea* (ID) per un automa push-down è una terna:

$$(q, w, \sigma) \in Q \times X^* \times \Gamma^*$$

dove:

- $q \in Q$ è lo stato corrente dell'unita di controllo;
- $w \in X^*$, $w = x_1 x_2 ... x_n$, è la porzione di stringa di simboli dell'alfabeto di ingresso da esaminare sul cui primo carattere, x_1, è posizionata la testina di lettura;

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

- $\sigma = Z_1 Z_2 ... Z_m$ è il contenuto della pila con Z_1 al top e Z_m al bottom. ■

Definizione 8.3 (Descrizione iniziale)

Una *descrizione* (q_0, z, Z_0), con $z \in X^*$, è detta *iniziale.* ■

Definizione 8.4 (Descrizione finale)

Una *descrizione* (q, λ, σ) è detta *finale.* ■

Le *transizioni* fanno corrispondere (in maniera non deterministica) descrizioni istantanee a descrizioni istantanee in due modi:

se M è nello stato q, il simbolo di input corrente è x ed il simbolo al top dello stack è A, allora:

1) se $< q, x, A, q', A_1 ... A_k >$, per $x \in X$, è una quintupla dell'automa a pila, allora M può (non deterministicamente) causare la seguente transizione:

$$(q, xw, A\sigma) \Rightarrow (q', w, A_1 ... A_k \sigma)$$

che può anche essere scritta nella forma:

$$\delta(q, x, A) = (q', A_1 ... A_k)$$

2) se $< q, \lambda, A, q', A_1 ... A_k >$ è una quintupla dell'automa a pila, allora la sua esecuzione può provocare la seguente transizione:

$$(q, xw, A\sigma) \Rightarrow (q', xw, A_1 ... A_k \sigma)$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

che può anche essere scritta nella forma:

$$\delta(q, \lambda, A) = (q', A_1 ... A_k)$$

Il caso 2) denota una transizione non deterministica che avviene senza leggere simboli in ingresso. Dunque, al termine, la testina di lettura non viene fatta avanzare.

La notazione $\overset{*}{\Rightarrow}$ denota la chiusura riflessiva e transitiva *dell'operatore di transizione*, ossia tale simbolo indica una successione di transizioni che porta M da una data descrizione istantanea ad un'altra.

La notazione $< terna1 > \overset{*}{\Rightarrow} < terna2 >$ significa che la descrizione istantanea $< terna2 >$ può essere derivata dopo una sequenza finita di zero o più transizioni da $< terna1 >$.

8.2 Linguaggi accettati da automi a pila

<u>Definizione 8.5</u> (Stringa accettata in condizione di pila vuota)

Dato un PDA M, diremo che $w \in X^*$ è una *stringa accettata da M in condizione di pila vuota* se:

$$(q_0, w, Z_0) \overset{*}{\Rightarrow} (q, \lambda, \lambda) \qquad q \in Q \qquad \blacksquare$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Definizione 8.6 (Linguaggio accettato in condizione di pila vuota)
Dato un PDA M, diremo che:

$$T(M) = \left\{ w \,\middle|\, (q_0, w, Z_0) \overset{*}{\Rightarrow} (q, \lambda, \lambda), \quad q \in Q \right\}$$

è il *linguaggio accettato da M nella condizione di pila vuota*. ■

Osservazione 8.1
I valori di X o di Γ per cui non è definita la funzione δ in uno stato q corrispondono a stringhe respinte dall'automa.

In alternativa alle definizioni precedenti di stringa accettata e di linguaggio accettato da un automa a pila nella condizione di pila vuota, si possono dare le seguenti definizioni:

Definizione 8.7 (Stringa accettata in condizione di stato finale)
Dato un PDA M, diremo che $w \in X^*$ è una *stringa accettata da M in condizione di stato finale* se:

$$(q_0, w, Z_0) \overset{*}{\Rightarrow} (q_a, \lambda, \sigma), \;\; q_a \in F, \;\; \sigma \in \Gamma^* \qquad ■$$

Definizione 8.8 (Linguaggio accettato in condizione di stato finale)
Dato un PDA M, diremo che:

$$T(M) = \left\{ w \,\middle|\, (q_0, w, Z_0) \overset{*}{\Rightarrow} (q_a, \lambda, \sigma), \;\; q_a \in F, \;\; \sigma \in \Gamma^* \right\}$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

è il *linguaggio accettato da M nella condizione di stato finale.* ■

Teorema 8.1

La classe dei linguaggi riconosciuti da un automa a pila nella condizione di stato finale coincide con quella riconosciuta nella condizione di pila vuota.

Dimostrazione 8.1

Per dimostrare $\mathcal{L}_{PDA_F} \equiv \mathcal{L}_{PDA_\lambda}$ si deve provare $\mathcal{L}_{PDA_F} \subseteq \mathcal{L}_{PDA_\lambda}$ e $\mathcal{L}_{PDA_F} \supseteq \mathcal{L}_{PDA_\lambda}$.

i) $\mathcal{L}_{PDA_F} \subseteq \mathcal{L}_{PDA_\lambda}$

Sia $L \in \mathcal{L}_{PDA_F} \overset{def}{\Leftrightarrow} \exists M = (Q, X, \Gamma, \delta, q_0, Z_0, F)$:

$$T(M) = \left\{ w \,\middle|\, (q_0, w, Z_0) \overset{*}{\Rightarrow} (q, \lambda, \sigma),\ q \in F \right\} = L.$$

Si deve costruire $M' = (Q', X, \Gamma', \delta', q_0', Z_0', F' = \varnothing)$:

$$T(M') = \left\{ w' \,\middle|\, (q_0', w, Z_0) \overset{*}{\Rightarrow} (q', \lambda, \lambda) \right\} = L$$

e quindi si deve avere:

- $\delta' = \delta$, tranne che:
  - i.1) $\delta'(q_0', \lambda, Z_0') = (q_0, Z_0 Z_0')$ (che serve ad M' per imitare la configurazione iniziale di M)
  - i.2) $\delta'(q, \lambda, A) = (q_p', \lambda) \quad \forall q \in F,\ \forall A \in \Gamma \cup \{Z_0'\} = \Gamma'$
  - i.3) $\delta'(q_p', \lambda, A) = (q_p', \lambda) \quad \forall A \in \Gamma \cup \{Z_0'\} = \Gamma'$
- $Q' = Q \cup \{q_0', q_p'\}$

- $\Gamma' = \Gamma \cup \{Z_0'\}$
- $F' = \varnothing$

ii) $\mathcal{L}_{PDA_F} \supseteq \mathcal{L}_{PDA_\lambda}$

Sia $L \in \mathcal{L}_{PDA_\lambda} \overset{def}{\Leftrightarrow} \exists M = (Q, X, \Gamma, \delta, q_0, Z_0, F)$:

$$T(M) = \left\{ w \,\middle|\, (q_0, w, Z_0) \overset{*}{\Rightarrow} (q, \lambda, \lambda) \right\} = L.$$

Si deve costruire $M' = (Q', X, \Gamma', \delta', q_0', Z_0', F')$:

$$T(M') = \left\{ w' \,\middle|\, (q_0', w, Z_0) \overset{*}{\Rightarrow} (q', \lambda, \sigma), \quad q' \in F \right\} = L$$

e quindi si deve avere:

- $\delta' = \delta$, tranne che:

  ii.1) $\delta'(q_0', \lambda, Z_0') = (q_0, Z_0 Z_0')$ (che serve ad M' per imitare la configurazione iniziale di M e per avere Z_0' al bottom, così quando il riconoscimento di una parola porterà M nella condizione di pila vuota, M' avrà in pila il solo carattere Z_0')

  ii.2) $\delta'(q, \lambda, Z_0') = (q_F', \lambda) \quad \forall q \in Q$ (M' si porta nell'unico stato finale)
- $Q' = Q \cup \{q_0', q_F'\}$
- $\Gamma' = \Gamma \cup \{Z_0'\}$
- $F' = \{q_F'\}$.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

8.3 Esempi di linguaggi riconosciuti da automi a pila

Esempio 8.2

Sia dato il seguente linguaggio L_2:

$$L_2 = \{w \mid w \text{ ha lo stesso numero di } a \text{ e di } b\}.$$

Una grammatica che genera L_2 è:

$$G_2 = (X, V, S, P_2)$$

dove:

$$X = \{a, b\} \qquad V = \{S\}$$
$$P_2 = \{S \to ab \mid ba \mid SS \mid aSb \mid bSa\}$$

che rispecchia la seguente definizione ricorsiva di L_2:

$$\begin{cases} ab, ba \in L_2 \\ \text{se } \ v, w \in L_2 \ \text{ allora } \ vw, awb, bwa \in L_2 \end{cases}$$

Costruiamo l'automa a pila M_2 che riconosce L_2 nella condizione di pila vuota:

$$M_2 = (Q, X, \Gamma, \delta, q_0, Z_0, F)$$

dove:

- $Q = \{q_0\}$
- $X = \{a, b\}$
- $\Gamma = \{Z_0, A, B\}$ $\qquad$ Z_0 indicatore del fondo dello stack
- $F = \varnothing$ (si intende accettare il linguaggio nella condizione di pila vuota)

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

La tavola di transizione di M_2 è la seguente:

CARATTERE DI PILA	STATO	CARATTERE DI INGRESSO	
		a	b
A	q_0	AA (3)	λ (5)
B	q_0	λ (6)	BB (4)
Z_0	q_0	AZ_0 (1)	BZ_0 (2)

corrispondente al seguente algoritmo.

Inizialmente lo stack è vuoto; la stringa w viene esaminata da sinistra a destra e si effettuano le seguenti operazioni:

(1) se lo stack è vuoto ed il simbolo corrente di w è a, poni A sullo stack;

(2) se lo stack è vuoto ed il simbolo corrente di w è b, poni B sullo stack;

(3) se il simbolo al top dello stack è A e il simbolo corrente di w è a, poni un'altra A sullo stack;

(4) se il simbolo al top dello stack è B e il simbolo corrente di w è b, poni un'altra B sullo stack;

(5) se il simbolo al top dello stack è A e il simbolo corrente di w è b, estrai un elemento dallo stack;

(6) se il simbolo al top dello stack è B e il simbolo corrente di w è a, estrai un elemento dallo stack.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Si osservi che l'unità di controllo di M_2 ha un unico stato, dunque nella casella della tavola di transizione non è necessario indicare lo stato.

Quindi:

$$T(M_2) = \left\{ w \,\middle|\, (q_0, w, Z_0) \overset{*}{\Rightarrow} (q, \lambda, \lambda), \quad q \in Q \right\} = L_2 .$$

L'algoritmo visto sopra corrisponde al seguente *programma per PDA*:

(1) $< a, Z_0, AZ_0 >$;
(2) $< b, Z_0, BZ_0 >$;
(3) $< a, A, AA >$;
(4) $< b, B, BB >$;
(5) $< a, B, \lambda >$;
(6) $< b, A, \lambda >$;
(7) $< \lambda, Z_0, \lambda >$;

in cui abbiamo introdotto la λ-regola (7) che ci consente di cancellare il marcatore del fondo dello stack Z_0 senza leggere alcun carattere di ingresso.

<u>Esempio 8.3</u>

Sia dato il seguente linguaggio L_3:

$$L_3 = \left\{ wcw^R \mid w \in \{a,b\}^* \right\}$$

Una grammatica che genera L_3 è:

$$G_3 = (X, V, S, P_3)$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

dove:

$$X = \{a, b, c\} \qquad V = \{S\}$$
$$P_3 = \{S \to c \mid aSa \mid bSb\}$$

che rispecchia la seguente definizione ricorsiva di L_3:

$$\begin{cases} c \in L_3 \\ \text{se } \ w \in L_3 \ \text{ allora } \ awa, bwb \in L_3 \end{cases}$$

Costruiamo l'automa a pila M_3 che riconosce L_3 nella condizione di pila vuota:

$$M_3 = (Q, X, \Gamma, \delta, q_0, Z_0, F)$$

dove:

- $Q = \{q_0, q_1\} \qquad q_0 = read, \ q_1 = match$
- $X = \{a, b, c\}$
- $\Gamma = \{Z_0, A, B\}$
- $F = \varnothing$ (si intende accettare il linguaggio nella condizione di pila vuota)

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

La tavola di transizione di M_3 è la seguente:

CARATTERE DI PILA	STATO	CARATTERE DI INGRESSO		
		a	b	c
A	q_0	(q_0, AA) (2)	(q_0, BA) (5)	(q_1, A) (8)
	q_1	(q_1, λ) (10)		
B	q_0	(q_0, AB) (3)	(q_0, BB) (6)	(q_1, B) (9)
	q_1		(q_1, λ) (11)	
Z_0	q_0	(q_0, AZ_0) (1)	(q_0, BZ_0) (4)	(q_1, Z_0) (7)
	q_1			

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

corrispondente al seguente programma per PDA:

(1) $< q_0, a, Z_0, q_0, AZ_0 >$;

(2) $< q_0, a, A, q_0, AA >$;

(3) $< q_0, a, B, q_0, AB >$;

(4) $< q_0, b, Z_0, q_0, BZ_0 >$;

(5) $< q_0, b, A, q_0, BA >$;

(6) $< q_0, b, B, q_0, BB >$;

(7) $< q_0, c, Z_0, q_1, Z_0 >$;

(8) $< q_0, c, A, q_1, A >$;

(9) $< q_0, c, B, q_1, B >$;

(10) $< q_1, a, A, q_1, \lambda >$;

(11) $< q_1, b, B, q_1, \lambda >$;

(12) $< q_1, \lambda, Z_0, q_1, \lambda >$; ($\lambda$ – regola)

dove:

- (1), (2) e (3) si possono riassumere con $< q_0, a, Z, q_0, AZ >$ con $Z = Z_0, A, B$;
- (4), (5) e (6) si possono riassumere con $< q_0, b, Z, q_0, BZ >$ con $Z = Z_0, A, B$;
- (7), (8) e (9) si possono riassumere con $< q_0, c, Z, q_1, Z >$ con $Z = Z_0, A, B$.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Esempio di *trace*:

TRACE	
STRINGA DA ESAMINARE	***STACK***
aabcbaa	Z_0
abcbaa	AZ_0
bcbaa	AAZ_0
cbaa	$BAAZ_0$
baa	$BAAZ_0$
aa	AAZ_0
a	AZ_0
	Z_0

Si osservi che:

- M_3 nello stato $q_1 = match$ si ferma se:

  legge una a con B al top dello stack;

  oppure:

  legge una b con A al top dello stack;

  …
- L_3 è semplice da riconoscere per un PDA perché il centro c dei palindromi segnala al PDA quando cambiare stato ed iniziare a svuotare lo stack.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Esempio 8.4

Sia dato il seguente linguaggio L_4:

$$L_4 = \left\{ ww^R \;\middle|\; w \in \{a,b\}^* \right\}$$

Una grammatica che genera L_4 è:

$$G_4 = (X, V, S, P_4)$$

dove:

$$X = \{a,b\}, \qquad V\{(S\},$$
$$P_4 = \{S \to aa \mid bb \mid aSa \mid bSb \mid \lambda\},$$

corrispondente alla seguente definizione ricorsiva di L_4:

$$\begin{cases} aa, bb \in L_4 \\ \text{se } \; w \in L_4 \; \text{ allora } \; awa, bwb \in L_4 \end{cases}$$

Osservazione

La sostanziale differenza tra L_3 ed L_4 è la mancanza del carattere c che in L_3 fungeva da separatore fra la stringa w e la speculare w^R. L'automa riconoscitore, in questo caso, non è in grado di capire se il carattere letto appartiene a w o a w^R; quindi deve operare come se fosse contemporaneamente nella fase di accumulo di informazioni nella pila oppure di scarico della pila.

Questo implica che solo un automa a pila non deterministico può riconoscere L_4.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

<u>Definizione 8.9</u> (Automa push-down deterministico)

Sia:

$$M = (Q, X, \Gamma, \delta, q_0, Z_0, F)$$

un PDA. M è per definizione deterministico se e solo se:

1) $\delta : Q \times (X \cup \{\lambda\}) \times \Gamma \to Q \times \Gamma^*$ $\qquad$ ($|\delta(q, x, Z)| = 1$)

2) se $|\delta(q, x, Z)| = 1$ e $x \neq \lambda$ allora $\delta(q, \lambda, Z)$ non è definito

■

Costruiamo l'automa a pila M_4 che riconosce L_4 nella condizione di pila vuota:

$$M_4 = (Q, X, \Gamma, \delta, q_0, Z_0, F)$$

dove:

- $Q = \{q_0, q_1\}$
- $X = \{a, b\}$
- $\Gamma = \{Z_0, A, B\}$
- $F = \varnothing$

La tavola di transizione di M_4 è la seguente:

CARATTERE DI PILA	STATO	CARATTERE DI INGRESSO	
		a	b
A	q_0	$\{(q_0,AA), (q_1,\lambda)\}$	(q_0,BA)
	q_1	(q_1,λ)	
B	q_0	(q_0,AB)	$\{(q_0,BB), (q_1,\lambda)\}$
	q_1		(q_1,λ)
Z_0	q_0	(q_0,AZ_0)	(q_0,BZ_0)
	q_1		

corrispondente al seguente programma per PDA:

(1) $< q_0, a, Z, q_0, AZ >$; $\qquad Z = Z_0, A, B$

(2) $< q_0, b, Z, q_0, BZ >$;

(3) $< q_0, \lambda, Z, q_1, Z >$ (sostituisce la λ - regola dell'esempio precedente);

(4) $< q_1, a, A, q_1, \lambda >$;

(5) $< q_1, b, B, q_1, \lambda >$;

(6) $< q_1, \lambda, Z_0, q_1, \lambda >$.

L'automa è non deterministico perché, ogniqualvolta è nello stato $q_0 = read$, ha due possibili transizioni:

- leggere un simbolo di input;
- commutare nello stato $q_1 = match$.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Osservazione

L_3 è accettato da un PDA deterministico.

L_4 è accettato da un PDA non deterministico,

e inoltre $\nexists M'_4$, M'_4 PDA deterministico, tale che:

$$T(M'_4) = L_4 .$$

Dunque *la classe dei linguaggi accettati da automi a pila non deterministici contiene propriamente la classe dei linguaggi accettati da automi a pila deterministici*.

8.4 Forme normali per grammatiche non contestuali

Esempio 8.5

Sia dato il seguente automa a stati finiti non deterministico (Figura 8.4):

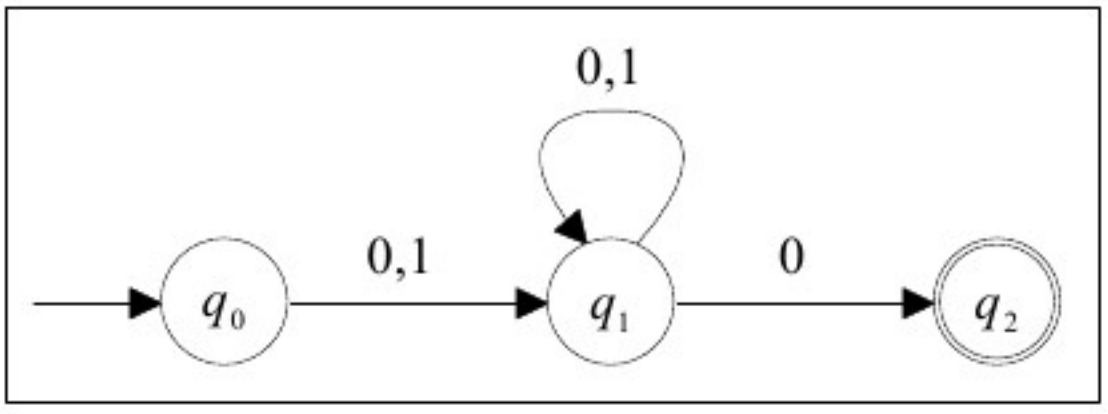


Figura 8.4

che riconosce il linguaggio:

$$L = \left\{ w0 \,\middle|\, w \in \{0,1\}^+ \right\} = S\left((0+1)^+ 0\right)$$

Una grammatica che genera L è:

$$G = (X, V, S, P)$$

$$X = \{0,1\} \qquad V = \{S, A\}$$

$$P = \{S \to 0A \mid 1A,\ A \to 0A \mid 1A \mid 0\}$$

Consideriamo la stringa $w = 0110$.

Esaminiamo come può essere riconosciuta; in particolare cerchiamo di identificare come, partendo dall'assioma S, può essere derivata w per successive applicazioni delle regole sintattiche.

La lettura del primo carattere, 0, permette di ricostruire univocamente la produzione che è stata usata per riscrivere S, cioè $S \to 0A$.

I successivi due caratteri 1 identificano entrambi la regola $A \to 1A$, mentre la lettura dell'ultimo carattere, 0, implica una scelta non deterministica tra la produzione $A \to 0A$ e $A \to 0$.

Ovviamente, qualora venga effettuata la prima scelta, sarà necessario successivamente, a causa del suo fallimento, considerare la seconda, che determina un esito positivo nell'analisi della stringa.

Quanto è stato qui esposto in modo informale può essere espresso in modo più rigoroso, considerando il seguente automa a pila:

$$M = (Q, X, \Gamma, \delta, q_0, Z_0, F)$$

dove:

- $Q = \{q_0\}$
- $X = \{0,1\}$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

- $\Gamma = \{S, A\}$
- $Z_0 = S$
- $F = \varnothing$
- δ è data da:
  - $\delta(q_0, 0, S) = \{(q_0, A)\}$;
  - $\delta(q_0, 1, S) = \{(q_0, A)\}$;
  - $\delta(q_0, 0, A) = \{(q_0, A), (q_0, \lambda)\}$;
  - $\delta(q_0, 1, A) = \{(q_0, A)\}$.

Osservazione 8.2

La pila dell'automa M contiene sempre un solo carattere al massimo. Ciò è dovuto al fatto che G è di tipo '3'.

Le operazioni che ad ogni istante vengono effettuate dall'automa M sono basate su una serie di *predizioni*; ad esempio, leggendo 0 sul nastro di ingresso, con S in cima alla pila, l'automa predice che il resto della stringa dovrà derivare da A.

Esempio 8.6

Esaminiamo la seguente grammatica non contestuale:

$$G = (X, V, S, P)$$

$$X = \{0,1,2\} \qquad V = \{S, A, B\}$$

$$P = \{S \to 0SAB \mid 1,\ \ A \to 1A \mid 1,\ \ B \to 2B \mid 2\}$$

Basandosi sulle considerazioni precedenti, anche in questo caso è possibile costruire un automa che riconosca il linguaggio generato

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

da G in modo predittivo. Per esempio, la lettura di 0 quando in cima alla pila c'è S fa sì che il resto della stringa debba derivare da SAB.

Esaminando quindi il primo carattere di tale sequenza e leggendo ancora uno 0 sul nastro di ingresso, si identifica nuovamente la produzione $S \to 0SAB$, quindi il resto della stringa deve derivare da $SABAB$.

I risultati delle successive predizioni vengono messi in una memoria a pila, che in questo esempio può effettivamente contenere un numero illimitato di simboli.

In modo formale, il riconoscimento di $L(G)$ può essere effettuato dal seguente automa a pila:

$$M = (Q, X, \Gamma, \delta, q_0, Z_0, F)$$

dove:

- $Q = \{q_0\}$
- $X = \{0,1,2\}$
- $\Gamma = \{S, A, B\}$
- $Z_0 = S$
- $F = \varnothing$ (linguaggio accettato nella condizione di pila vuota)
- δ è data da:
  - $\delta(q_0, 0, S) = \{(q_0, SAB)\}$
  - $\delta(q_0, 1, S) = \{(q_0, \lambda)\}$
  - $\delta(q_0, 1, A) = \{(q_0, A), (q_0, \lambda)\}$
  - $\delta(q_0, 2, B) = \{(q_0, B), (q_0, \lambda)\}$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Gli esempi precedenti trovano una giustificazione nel seguente:

Teorema 8.2 (senza dimostrazione)

Sia $G = (X, V, S, P)$ una grammatica non contestuale le cui produzioni sono del tipo $A \to a\alpha,\ \alpha \in V^*,\ a \in X$.

$L(G)$ è riconosciuto dall'automa a pila:

$$M = (Q, X, \Gamma, \delta, q_0, Z_0, F)$$

dove:

- $Q = \{q_0\}$
- $\Gamma = V$
- $Z_0 = S$
- $F = \varnothing$ (linguaggio accettato nella condizione di pila vuota)
- $\forall A \to a\alpha \in P : (q_0, \alpha) \in \delta(q_0, a, A)$
- $\forall A \to \lambda \in P : (q_0, \lambda) \in \delta(q_0, \lambda, A)$ (poi vedremo che in Forma normale di Greibach è ammessa solo $S \to \lambda$)

Definizione 8.9 (Forma normale di Chomsky)

Si dice che una grammatica non contestuale $G = (X, V, S, P)$ è in *forma normale di Chomsky* se ogni produzione è in una delle seguenti forme:

i) $S \to \lambda$;

ii) $A \to BC$, dove $A, B, C \in V$; se $S \to \lambda \in P$ allora $B, C \in V - \{S\}$;

iii) $A \to a$, dove $A \in V,\ a \in X$. ■

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Definizione 8.10 (Forma normale priva di ricorsioni sinistre)

Si dice che una grammatica non contestuale $G = (X, V, S, P)$ è in *forma normale priva di ricorsioni sinistre* (*NLR = No Left Recursion*) se non ha produzioni della forma:

$$A \to Av$$

dove $A \in V$ e $v \in (X \cup V)^*$ (dette *produzioni ricorsive a sinistra*). ■

Definizione 8.11 (Forma normale di Greibach)

Si dice che una grammatica non contestuale $G = (X, V, S, P)$ è in *forma normale di Greibach* (*GNF = Greibach Normal Form*) se ogni produzione è del tipo:

i) $S \to \lambda$;

ii) $A \to a\alpha$, dove $a \in X$ e $\alpha \in V^*$. ■

Teorema 8.3

Sia $G = (X, V, S, P)$ un'arbitraria grammatica non contestuale. Esistono allora tre grammatiche G', G'' e G''', equivalenti a G, ossia tali che $L(G) = L(G^i)$, con $i = ', '', '''$,

con G' in forma normale di Chomsky,

G'' in forma normale di Greibach,

G''' in forma normale priva di ricorsioni sinistre.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Dimostrazione 8.3

La dimostrazione di questo teorema è costruttiva.

L'algoritmo per la trasformazione di una grammatica non contestuale G in una grammatica equivalente G' in forma normale di Chomsky è illustrato nell'Esercizio 8.1, mentre quello per la trasformazione di G in una grammatica equivalente G'' in forma normale di Greibach è dato nell'Esercizio 8.2. Un passo di questo algoritmo effettua la trasformazione di G in una grammatica equivalente G''' in forma normale priva di ricorsioni sinistre.

c.v.d.

Esercizio 8.1

Convertire la seguente grammatica in forma normale di Chomsky:

$$G = (X, V, S, P)$$

$$X = \{0,1,2\} \qquad V = \{S, A, B\}$$

$$P = \{S \to 00A \mid B \mid 1,\ \ A \to 1AA \mid 2,\ \ B \to 0\}$$

G non presenta λ-regole. Se così non fosse, occorrerebbe applicare preventivamente il Passo 1 dell'Algoritmo 8.2.

Algoritmo 8.1 (Trasformazione in forma normale di Chomsky).

Passo 0 **Eliminazione delle λ-produzioni del tipo** $A \to \lambda,\ \ A \neq S$ (vedi Algoritmo 8.2).

Passo 1 ***Conversione dei terminali che compaiono nella parte destra di qualche produzione in nuovi nonterminali e aggiunta delle produzioni appropriate che riscrivono i***

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

nuovi ***nonterminali***

$(\forall A \to \alpha \in P',\ \alpha \in (X \cup V)^*,\ |\alpha| > 1)$.

$S \to BBA \mid B \mid 1$

$A \to CAA \mid 2$

$C \to 1$

$B \to 0$

Tutte le produzioni in P' sono ora del tipo:

i) $S \to \lambda$

ii) $A \to \alpha,\ \alpha \in V^+,\ A \in V$

iii) $A \to a,\ a \in X,\ A \in V$

<u>Passo 2</u> ***Suddivisione delle produzioni le cui parti destre hanno più di due nonterminali*** ossia delle produzioni del tipo $A \to \alpha$, $\alpha \in V^+$, $|\alpha| > 2$, introducendo k-1 nuove produzioni, se $|\alpha| = k$, del tipo:

$$A \to X_1 X_2 \ldots X_k \quad \begin{matrix}\text{viene}\\ \text{rimpiazzata}\\ \text{con}\end{matrix} \quad \left.\begin{matrix} A & \to & X_1 Y_1 \\ Y_1 & \to & X_2 Y_2 \\ \vdots & & \\ Y_{k-2} & \to & X_{k-1} X_k \end{matrix}\right\} k-1$$

$S \to BD \mid B \mid 1$

$D \to BA$

$A \to CE \mid 2$

$E \to AA$

$C \to 1$

$B \to 0$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Tutte le produzioni in P' sono ora del tipo:

i) $S \to \lambda$

ii) $A \to \alpha,\ |\alpha| \le 2,\ \alpha \in V^+$

iii) $A \to a,\ a \in X$

Le uniche produzioni che <u>non</u> sono ancora in forma normale di Chomsky sono le $A \to B$ in ii).

<u>Passo 3</u> ***Sostituzione dei nonterminali che costituiscono, da soli, le parti destre di qualche produzione del tipo*** $A \to B$. Si utilizzano le produzioni che riscrivono B, ossia le $B \to \beta_1 \mid \beta_2 \mid \ldots \mid \beta_k$, ove le $B \to \beta_i$ sono già in forma normale di Chomsky oppure sono del tipo $B \to C$, $\beta_i = C$. E' dunque un passo iterativo, che termina se la G di partenza non aveva nonterminali ciclici (vedi p. 229).

$S \to BD \mid 0 \mid 1$

$D \to BA$

$A \to CE \mid 2$

$E \to AA$

$C \to 1$

$B \to 0$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Esercizio 8.2

Convertire la seguente grammatica in forma normale di Greibach:

$$G = (X, V, S, P)$$

$$X = \{a, b, c, d\} \qquad V = \{S, A, B, C, D\}$$

$$P = \{S \to AaB,\ B \to CD \mid c,\ D \to ab \mid a,\ C \to BC \mid d,\ A \to \lambda\}$$

Algoritmo 8.2 (Trasformazione in forma normale di Greibach).

Per convertire una grammatica in forma normale di Greibach sono previsti i seguenti passi:

1) Eliminazione delle λ-regole;
2) Eliminazione dei nonterminali inutili;
3) Eliminazione dei nonterminali ciclici;
4) Eliminazione delle produzioni $A \to B$;
5) Eliminazione delle produzioni del tipo $A \to B\beta$;
6) Eliminazione delle ricorsioni sinistre;
7) Introduzione di nuovi nonterminali.

Passo 1 ***Eliminazione delle λ-regole*** (del tipo $A \to \lambda$).

Dividiamo V in due sottoinsiemi V_1 e V_2:

$$V_1 = \left\{ A \in V \;\middle|\; A \overset{*}{\Rightarrow} \lambda \right\},\ V_2 = V - V_1 .$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Applichiamo il seguente algoritmo (Algoritmo 8.2.1) che, per ogni nonterminale A di una grammatica G, calcola l'attributo $e(A)$, che è vero se e solo se $A \overset{*}{\Rightarrow} \lambda$, cioè:

$$A \in V : e(A) = true \overset{def}{\Leftrightarrow} A \overset{*}{\Rightarrow} \lambda .$$

<u>Algoritmo 8.2.1</u> (Calcolo dei nonterminali che generano λ).

1) $\forall A \in V : e(A) \leftarrow false$;
2) se $A \to \lambda \in P$, $e(A) \leftarrow true$;

   e si marcano (con un apice) tutte le occorrenze di A che appaiono nelle parti destre delle produzioni di G;
3) $\forall A \to \alpha \in P$, si cancellano da α i nonterminali marcati; se la parte destra diventa vuota $e(A) \leftarrow true$, e si marcano (con un apice) tutte le occorrenze di A che appaiono nelle parti destre delle produzioni di G;
4) se nel passo precedente è mutato qualche $e(A)$ si riapplica 3), altrimenti l'algoritmo termina.

Nel caso della grammatica dell'Esercizio 8.2, si ha:

$e(\ )$	1)	2)	3)	4)
S	0	0	0	0
A	0	1	1	1
B	0	0	0	0
C	0	0	0	0
D	0	0	0	0

0 = false

1 = true

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

L'unico nonterminale per cui $e(\ )$ è vero è proprio A. Per cui:

$$V_1 = \{A\}.$$

Si trasforma la grammatica $G = (X, V, S, P)$ contenente λ-regole in una grammatica $G' = (X, V, S, P')$ (secondo il Lemma della Stringa Vuota - Lemma 5.2), attraverso le seguenti regole, che costituiscono la dimostrazione costruttiva del Lemma 5.2:

i) $S \to \lambda \in P' \Leftrightarrow S \overset{*}{\underset{G}{\Rightarrow}} \lambda \quad (\lambda \in L(G))$;

ii) Se $A \to X_1 X_2 ... X_r \in P,\ r \geq 1,$ allora in P' si trovano tutte le produzioni del tipo:

$$A \to Y_1 Y_2 ... Y_r$$

ove:

se $X_i \in X \cup V_2$, allora $Y_i = X_i$;

altrimenti (se $X_i \in V_1$), si pone $Y_i = X_i$ oppure $Y_i = \lambda$, eliminando però il caso $Y_1 Y_2 ... Y_r = \lambda$.

La grammatica G' è data da:

$$P' = \{S \to aB \mid AaB,\ B \to CD \mid c,\ D \to ab \mid a,\ C \to BC \mid d\}.$$

<u>Passo 2</u> ***Eliminazione dei nonterminali inutili***.

Un'operazione di potatura alla quale si assoggetta normalmente una grammatica consiste nella eliminazione di eventuali *nonterminali inutili* in essa presenti.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Definizione 8.12 (Nonterminale inutile)

Un *nonterminale A* è *inutile* se:

a) A non genera alcuna stringa terminale oppure

b) da S non deriva alcuna forma di frase contenente A. ■

L'algoritmo seguente calcola, per ogni nonterminale A, gli attributi $t(A)$ e $s(A)$, così definiti:

$$A \in V: \quad t(A) = true \overset{def}{\Leftrightarrow} A \overset{*}{\Rightarrow} w, \ w \in X^*$$
$$s(A) = true \overset{def}{\Leftrightarrow} S \overset{*}{\Rightarrow} \alpha A \beta, \ \alpha, \beta \in (X \cup V)^*$$

Algoritmo 8.2.2 (Eliminazione dei nonterminali inutili).

1) $\forall A \in V : t(A)(A) \leftarrow s(A) \leftarrow false$;

2) se $A \to w \in P, \ w \in X^*, \ t(A) \leftarrow true$,

   e si marcano (con un apice 't') tutte le occorrenze di A che appaiono nelle parti destre delle produzioni di G;

3) $s(S) \leftarrow true$, e si marcano con apice 's' tutte le occorrenze di S che appaiono come parte sinistra in P;

4) si effettuano le procedure seguenti 4.1) e 4.2):

   4.1) $\forall A \to \alpha \in P$, se tutti i nonterminali di α sono marcati con 't', $t(A) \leftarrow true$, e si marcano con apice 't' tutte le occorrenze di A che appaiono nelle parti destre delle produzioni di G;

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

4.2) se A è marcato con 's', si pone $s(B) \leftarrow true$ $\forall A \rightarrow \alpha B \beta$, e si marcano con apice 's' tutte le occorrenze di B che appaiono come parte sinistra in P;

5) se nel passo precedente è mutato qualche $t(A)$ o $s(A)$ si riapplica 4), altrimenti

6) per ogni A per cui $t(A) = false$ oppure $s(A) = false$, si cancellano da G tutte le produzioni in cui compare A (si cancellano gli A per cui $\neg s(A) \vee \neg t(A)$).

Nella grammatica data si ha:

	1)		2)		3)		4.1)		4.2)		5)		5)	
	$t(\,)$	$s(\,)$	$t(\,)$	$s(\,)$	$t(\,)$	$s(\,)$	$t(\,)$	$s(\,)$	$t(\,)$	$s(\,)$	$t(\,)$	$s(\,)$	$t(\,)$	$s(\,)$
S	0	0	0	1	1	1	1	1	1	1				
A	0	0	0	0	0	1	0	1	0	1				
B	0	0	1	0	1	1	1	1	1	1				
C	0	0	1	0	1	0	1	1	1	1				
D	0	0	1	0	1	0	1	1	1	1				

$S^s \rightarrow aB^t \mid AaB^t$

$B^s \rightarrow C^t D^t \mid c$

$D^s \rightarrow ab \mid a$

$C^s \rightarrow B^t C^t \mid d$

A è l'unico nonterminale inutile. Dunque (al passo 6) si cancella la sola produzione $S \rightarrow AaB$.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Passo 3 ***Eliminazione dei nonterminali ciclici***.

Un'altra operazione di potatura alla quale si assoggetta una grammatica, prima di ogni ulteriore elaborazione, consiste nella eliminazione di eventuali *nonterminali ciclici*.

Definizione 8.13 (Nonterminale ciclico)

Un *nonterminale A* è *ciclico* se:

$$A \overset{+}{\Rightarrow} A$$

ossia se A genera se stesso in un numero finito di passi. ■

L'algoritmo seguente calcola, per ogni nonterminale A, l'insieme:

$$c(A) = \left\{ B \;\middle|\; A \overset{+}{\Rightarrow} B \right\}.$$

Algoritmo 8.2.3 (Calcolo dell'insieme $c(A)$).

1) $\forall A \in V : c(A) \leftarrow \varnothing$;

2) $\forall A \to \alpha \in P$, dove $\alpha = \beta B \gamma$ e $\beta \overset{*}{\Rightarrow} \lambda$, $\gamma \overset{*}{\Rightarrow} \lambda$, si pone:

$$c(A) \leftarrow c(A) \cup \{B\};$$

3) $\forall A \in V$, se $B \in c(A)$, $c(A) \leftarrow c(A) \cup c(B)$;

4) Se in 3) è mutata la composizione di almeno un insieme c, si riapplica 2), altrimenti l'algoritmo termina.

I nonterminali ciclici sono gli A per cui $A \in c(A)$.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Ricerchiamo gli eventuali nonterminali ciclici in:

$$S \to aB \qquad D \to ab \mid a$$
$$B \to CD \mid c \qquad C \to BC \mid d$$

$c(\ )$	1)	2)	3)
S	$\varnothing$	$\varnothing$	$\varnothing$
B	$\varnothing$	$\varnothing$	$\varnothing$
C	$\varnothing$	$\varnothing$	$\varnothing$
D	$\varnothing$	$\varnothing$	$\varnothing$

Dunque, non vi sono nonterminali ciclici.

Riprendendo l'algoritmo, le eventuali ciclicità possono essere eliminate sopprimendo tutte le produzioni del tipo $A \to B$, senza alterare il linguaggio generato, mediante il seguente algoritmo (Algoritmo 8.2.4) che costruisce, a partire da P, un nuovo insieme P' di produzioni.

<u>Passo 4</u> ***Eliminazione delle produzioni*** $A \to B$.

<u>Algoritmo 8.2.4</u> (Eliminazione delle produzioni $A \to B$).[14]

1) Si partiziona P in P' e P'', ove P'' contiene tutte e sole le produzioni del tipo: $A \to B$

$$P'' = \{A \to B \in P \mid A, B \in V\};$$

2) $P' \leftarrow P' \cup \{A \to \alpha \mid B \in c(A) \text{ e } B \to \alpha \in P'\}$.

[14] Questo algoritmo può generare non terminali ciclici.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Passo 5 ***Eliminazione delle produzioni del tipo*** $A \to B\beta$.[15]

Supponendo che le produzioni che trascrivono B siano:

$$B \to \alpha_1 \mid \alpha_2 \mid ... \mid \alpha_n$$

la $A \to B\beta$ può essere sostituita dalle produzioni:

$$A \to \alpha_1\beta \mid \alpha_2\beta \mid ... \mid \alpha_n\beta$$

senza che il linguaggio generato venga modificato.

Se tutti gli α_i iniziano con un carattere terminale allora il procedimento di modifica innescato dalla $A \to B\beta$ ha termine, altrimenti se α_k, $1 \le k \le n$, non inizia con un carattere terminale occorre ripetere il procedimento per $A \to \alpha_k\beta$.

Tale procedimento induce una relazione d'ordine sull'insieme dei nonterminali della grammatica ed ha termine soltanto se:

a) i nonterminali inutili sono stati già eliminati;

b) la grammatica non contiene *ricorsioni sinistre*, cioè non esiste alcuna derivazione del tipo:

$$A \overset{*}{\Rightarrow} A\alpha$$

Sotto le ipotesi a) e b), G viene trasformata in G' tale che $L(G) = L(G')$ ed avente solo produzioni del tipo $A \to a\alpha,\ \alpha \in (X \cup V)^*,\ a \in X$.

---

[15] Il seguente procedimento può generare nonterminali inutili.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

A questo punto, è immediato introdurre, per ogni terminale di α, un nuovo nonterminale e trasformare G' in forma normale di Greibach.

Tornando al nostro esempio,

$$S \to aB$$
$$B \to CD \mid c$$
$$D \to ab \mid a$$
$$C \to BC \mid d$$

non vi sono produzioni del tipo $A \to B$ (Passo 4) mentre vi sono due produzioni del tipo $A \to B\beta$ (Passo 5), vale a dire:

$$B \to CD \qquad \text{e} \qquad C \to BC\,.$$

Imponiamo una relazione d'ordine sull'insieme dei nonterminali della grammatica:

$$V = \{S, B, C, D\} = \{A_1, A_2, A_3, A_4\}$$
$$ord(S) < ord(B) < ord(C) < ord(D)$$

Desideriamo che le produzioni di G siano di questo tipo:

(a) $A_i \to A_j v$, con $i < j$;

oppure

(b) $A_i \to av$, con $a \in X$, $v \in (X \cup V)^*$.

Esaminiamo le produzioni di G, tenendo presente l'ordine imposto sull'insieme dei nonterminali:

$S \to aB$ è di tipo (b)

$B \to CD$ è di tipo (a)

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

$B \to c$ è di tipo (b)

$D \to ab \mid a$ è di tipo (b)

$C \to d$ è di tipo (b)

$C \to BC$ non è né di tipo (a) (perché $ord(C) > ord(B)$) né di tipo (b).

La trasformiamo come segue:

le produzioni che trascrivono B sono:

$$B \to \underbrace{CD}_{\alpha_1} \mid \underbrace{c}_{\alpha_2}$$

La $C \to B\underbrace{C}_{\beta}$ può dunque essere sostituita da:

$$C \to \underbrace{CD}_{\alpha_1}\underbrace{C}_{\beta} \mid \underbrace{c}_{\alpha_2}\underbrace{C}_{\beta}$$

ottenendo ancora una grammatica equivalente a G:

$$S \to aB$$
$$B \to CD \mid c$$
$$D \to ab \mid a$$
$$C \to CDC \mid cC \mid d$$

<u>Passo 6</u> ***Eliminazione delle ricorsioni sinistre***.

Supponiamo che G abbia una produzione del tipo

$$A \to Av \mid w$$

con $v, w \in (X \cup V)^*$ e $w \neq A\gamma,\ \gamma \in (X \cup V)^*$.[16]

[16] Cioè w non inizia per A.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

È immediato osservare che:

$$A \overset{*}{\Rightarrow} wv^* .$$

Ma il linguaggio wv^* può essere generato per concatenazione di w e di v^*, che sono generati rispettivamente da:

$$A \to w \qquad \text{e} \qquad B \to vB \mid \lambda$$

Concatenando queste due grammatiche si ottiene:

$$A \to w \mid wB$$
$$B \to vB \mid v \mid \lambda$$

In G, l'unica ricorsione sinistra è data da:

$$C \to C\underbrace{DC}_{v} \mid \underbrace{cC}_{w_1} \mid \underbrace{d}_{w_2}$$

Dunque, questa produzione può essere trasformata in:

$$C \to cC \mid d \mid cCE \mid dE$$
$$E \to DCE \mid DC$$

ottenendo così:

$$S \to aB$$
$$B \to CD \mid c$$
$$D \to ab \mid a$$
$$C \to cC \mid d \mid cCE \mid dE$$
$$E \to DCE \mid DC$$

Poiché abbiamo introdotto un nuovo nonterminale, E, dobbiamo ridefinire un ordinamento di V. Se consideriamo il seguente ordinamento:

$$V = \{S, B, C, E, D\} = \{A_1, A_2, A_3, A_4, A_5\}$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

tutte le produzioni di G sono del tipo (a) o (b).

Ritorniamo ad effettuare il passo 5.

$$B \to C\underbrace{D}_{\beta} \text{ diventa } B \to \underbrace{cC}_{\alpha_1}\underbrace{D}_{\beta} \mid \underbrace{d}_{\alpha_2}\underbrace{D}_{\beta} \mid \underbrace{cCE}_{\alpha_3}\underbrace{D}_{\beta} \mid \underbrace{dE}_{\alpha_4}\underbrace{D}_{\beta}$$

considerando le produzioni che trascrivono C:

$$C \to \underbrace{cC}_{\alpha_1} \mid \underbrace{d}_{\alpha_2} \mid \underbrace{cCE}_{\alpha_3} \mid \underbrace{dE}_{\alpha_4}$$

e

$$E \to DC\mathrm{E} \mid DC \text{ diventa } E \to abCE \mid aCE \mid abC \mid aC$$

dando luogo a:

$$S \to aB$$
$$B \to cCD \mid dD \mid cCED \mid dED \mid c$$
$$D \to ab \mid a$$
$$C \to cC \mid d \mid cCE \mid dE$$
$$E \to abCE \mid aCE \mid abC \mid aC$$

Poiché non vi sono ricorsioni sinistre, possiamo finalmente effettuare il:

<u>Passo 7</u> ***Introduzione di nuovi nonterminali***.

Le produzioni sono ora tutte nella forma:

$$A \to a\alpha,\ a \in X,\ \alpha \in (X \cup V)^*.$$

Per ogni terminale in α, introduciamo un nuovo nonterminale ed una opportuna produzione.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Si ottiene quindi la grammatica G:

$$S \to aB$$
$$B \to cCD \mid dD \mid cCED \mid dED \mid c$$
$$D \to aF \mid a$$
$$C \to cC \mid d \mid cCE \mid dE$$
$$E \to aFCE \mid aCE \mid aFC \mid aC$$
$$F \to b$$

G è in forma normale di Greibach.

La possibilità di trasformare in modo algoritmico una qualunque grammatica non contestuale in una grammatica in forma normale di Greibach è un risultato particolarmente importante.

Da esso discendono i seguenti due teoremi:

Teorema 8.4

Ogni linguaggio non contestuale è generabile da una grammatica in forma normale di Greibach; tale grammatica è costruibile in modo algoritmico.

Teorema 8.5

Ogni linguaggio non contestuale è riconoscibile da un automa a pila.

Dimostrazione 8.5

Sia L un linguaggio non contestuale, allora per definizione

$$\exists\, G = (X, V, S, P) \text{ di Tipo } 2 : L(G) = L$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

ed inoltre si può assumere che G sia in forma normale di Greibach per il Teorema 8.4.

Si vuole costruire $M = (Q, X', \Gamma, \delta, q_0, Z_0, F)$ tale che $T\lambda(M) = L$.

Poniamo allora:

$$Q = \{q_0\}$$
$$X' = X$$
$$\Gamma = V$$
$$Z_0 = S$$
$$\delta : Q \times (X \cup \{\lambda\}) \times \Gamma \to 2^{Q \times \Gamma^*} \quad \text{con} \quad \begin{array}{lll} (q_0, \alpha) \in \delta(q_0, a, A) & \text{se} & A \to a\alpha \in P \\ (q_0, \lambda) \in \delta(q_0, \lambda, S) & \text{se} & S \to \lambda \in P \end{array}$$

Per dimostrare che $T\lambda(M) = L$ si prova il risultato più forte:

$$(q_0, w_1 w_2, S) \overset{*}{\underset{M}{\Rightarrow}} (q_0, w_2, \sigma) \;\Leftrightarrow\; S \overset{*}{\underset{G}{\Rightarrow}} w_1 \sigma$$

che come caso particolare per $w_2 = \lambda = \sigma$ si ha :

$$\begin{array}{ccc} w_1 \in T_\lambda(M) & & w_1 \in L(G) \\ (q_0, w_1 w_2, S) \overset{*}{\underset{M}{\Rightarrow}} (q_0, w_2, \sigma) & \Leftrightarrow & S \overset{*}{\underset{G}{\Rightarrow}} w_1 \sigma \end{array}$$

Si procede per induzione su $|w_1|$

<u>Passo base</u>: $|w_1| = 0 \quad w_1 = \lambda$

$(q_0, w_2, S) \overset{*}{\underset{M}{\Rightarrow}} (q_0, w_2, S)$ e dunque $S \overset{*}{\Rightarrow} S$

oppure

$(q_0, w_2, S) \overset{*}{\underset{M}{\Rightarrow}} (q_0, w_2, \lambda)$ (se $S \to \lambda \in P$) e dunque $S \overset{*}{\Rightarrow} \lambda$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Passo induttivo:

Supponiamo che w_1 sia tale che

$(q_0, w_1 w_2, S) \overset{*}{\underset{M}{\Rightarrow}} (q_0, w_2, \sigma)$ *iff* $S \overset{*}{\underset{G}{\Rightarrow}} w_1 \sigma$.

Siano $\sigma = A\tau$ e $w_2 = a\, w_3$, per ipotesi induttiva si ha:

$$(q_0, w_1\, a\, w_3, S) \overset{*}{\underset{M}{\Rightarrow}} (q_0, a\, w_3, A\tau) \ \ iff \ \ S \overset{*}{\underset{G}{\Rightarrow}} w_1 A \tau$$

Ma allora, se $A \to a\sigma' \in P$, si ha:

$$(q_0, w_1\, a\, w_3, S) \overset{*}{\underset{M}{\Rightarrow}} (q_0, w_3, \sigma'\tau) \ \ iff \ \ S \overset{*}{\underset{G}{\Rightarrow}} w_1\, a\, \sigma'\tau \ \text{ c.v.d.}$$

Si può anche dimostrare il seguente risultato:

Teorema 8.6

Ogni linguaggio riconoscibile da un automa a pila è generabile da una grammatica non contestuale.

Dimostrazione 8.6

Viene dimostrato preventivamente che $\mathcal{L}_{PDA} \equiv \mathcal{L}_{PDA_{1-stato}}$.

Sia $M = (Q, X, \Gamma, \delta, q_0, Z_0, F)$ un PDA con $|Q| > 1$, si vuole costruire $M' = (Q', X, \Gamma', \delta', q_0', Z_0', F')$ con $|Q'| = 1$ e $T(M') = T(M)$: l'idea per realizzare ciò è di definire δ' in modo che M' possa simulare sul suo stack i cambiamenti di stato dell'unità di controllo di *M*.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Se $p,q \in Q,\ A \in \Gamma$ e in M si ha che "l'unità di controllo è in p con A al top dello stack, quando A viene cancellato, M si porta nello stack q" cioè $(p,w,A\sigma) \overset{*}{\underset{M}{\Rightarrow}} (q,w',\sigma)$, allora scrivo $[pAq]$ sulla pila di M' con lo stesso significato.

Se $\begin{array}{c} < p,a,A,q,BCD > \\ (q,BCD) \in \delta(p,a,A) \end{array}$ in M, allora si può imitare il comportamento di M in M' con le transizioni

$$< q_0,a,[pA*_3],q_0,[qB*_1][*_1 C*_2][*_2 C*_3] >$$
$$(q_0,[qB*_1][*_1 C*_2][*_2 C*_3]) \in \delta(q_0,a,[pA*_3])$$

Da qui si può dimostrare che $T(M') = T(M)$ e ciò dimostra che $\mathcal{L}_{PDA} \equiv \mathcal{L}_{PDA_{1-stato}}$.

Riprendiamo a dimostrare che $\mathcal{L}_{PDA} \subseteq \mathcal{L}_2$.

Sia $L \in \mathcal{L}_{PDA}$ allora $\exists M = (Q,X,\Gamma,\delta,q_0,Z_0,F) : T(M) = L$ e per quanto dimostrato in precedenza si può supporre, senza ledere la generalità della dimostrazione, che M sia un $PDA_{1\text{-}stato}$ con $Q = \{q_0\}$.

Volendo determinare $G = (X',V,S,P)$ di Tipo 2 tale che $L(G) = L$, basta porre

$$X' = X$$
$$V = \Gamma$$
$$S = Z_0$$

e costruire P come

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

$$A \to a\,\alpha \in P \quad se \quad \begin{array}{c} < q_0, a, A, q_0, \alpha > \\ (q_0, \alpha) \in \delta(q_0, a, A) \end{array}$$

$$A \to \alpha \in P \quad se \quad \begin{array}{c} < q_0, \lambda, A, q_0, \alpha > \\ (q_0, \alpha) \in \delta(q_0, \lambda, A) \end{array}$$

Si può dimostrare (similmente a quanto fatto nella dimostrazione 8.5) che $L(G) = T(M)$.

c.v.d.

Questo teorema con i due precedenti, stabilisce l'*equivalenza tra grammatiche non contestuali ed automi a pila*.

Esercizio 8.3

Convertire la seguente grammatica in forma normale di Greibach:

$$S \to 00A \mid B \mid 1$$
$$A \to 1AA \mid 2$$
$$B \to 0$$

Nell'Esercizio 8.1 abbiamo trasformato la grammatica data in forma normale di Chomsky:

$$S \to BD \mid 0 \mid 1$$
$$D \to BA$$
$$A \to CE \mid 2$$
$$E \to AA$$
$$C \to 1$$
$$B \to 0$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

I passi 1) - 4) della procedura di trasformazione di una CFG in una grammatica in forma normale di Greibach ad essa equivalente non modificano la grammatica.

Eliminiamo le produzioni che non sono in forma normale di Greibach, ossia quelle del tipo:

$$A \to B\beta$$

Esse sono:

$$S \to BD$$
$$D \to BA$$
$$A \to CE$$
$$E \to AA$$

$S \to BD$	diventa	$S \to 0D$	per mezzo di $B \to 0$
$D \to BA$	diventa	$D \to 0A$	per mezzo di $B \to 0$
$A \to CE$	diventa	$A \to 1E$	per mezzo di $C \to 1$
$E \to AA$	diventa	$E \to 1EA \mid 2A$	per mezzo di $A \to 1E \mid 2$

e la grammatica in forma normale di Greibach, equivalente a quella di partenza, è:

$$S \to 0D \mid 0 \mid 1$$
$$D \to 0A$$
$$A \to 1E \mid 2$$
$$E \to 1EA \mid 2A$$
$$\cancel{C \to 1}$$
$$\cancel{B \to 0}$$

ove le due ultime produzioni sono state eliminate in virtù del fatto che C e B sono nonterminali inutili.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

8.5 Proprietà decidibili dei linguaggi liberi da contesto

Teorema 8.8

Sia L un linguaggio libero. Il problema:

(i) ‘ $L = \varnothing$ ’ è decidibile.

(ii) ‘L finito’ è decidibile.

(iii) ‘L infinito’ è decidibile.

(iv) della derivabilità $x \overset{*}{\underset{G}{\Rightarrow}} y$ è decidibile.

Dimostrazione 8.8

(i) Sia G una grammatica t.c. $L(G) = L$. Applico a G gli algoritmi 8.2.2 di eliminazione dei nonterminali inutili e 8.2.4 di eliminazione dei nonterminali ciclici.

‘ $L(G) = \varnothing$ ’ *iff* $S \overset{*}{\underset{G}{\Rightarrow}} w,\ \forall w \in X^*\ \ \left(\Leftrightarrow t(S) = \mathit{false}\right)$.

(ii) Riapplico gli algoritmi 8.2.2 (eliminazione dei nonterminali inutili) e 8.2.4 (eliminazione dei nonterminali ciclici)

‘$L(G)$ finito’ *iff* $\nexists A \in V : A \overset{+}{\Rightarrow} \alpha A \beta$ con $\alpha\beta \neq \lambda$, A nonterminale ricorsivo.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

$L(G) = \varnothing$ — SI se $t(S)$=*false*; NO se $t(S)$=*true*

L infinito — SI $\exists A \in V : A \overset{+}{\Rightarrow} \alpha A \beta$, $\alpha\beta \neq \lambda$; NO altrimenti

Teorema 8.9

Siano G e G' grammatiche libere da contesto.

'$L(G) = L(G')$' è indecidibile (l'equivalenza di grammatiche Tipo 2 è indecidibile) poiché '$[L(G) \setminus L(G')] \cup [L(G') \setminus L(G)] = \varnothing$' non è decidibile ($\mathcal{L}_2$ non è chiusa rispetto a complemento e intersezione).

Osservazione

Nel caso di linguaggi dipendenti da contesto si ha:

$\mathcal{L}_0$	il problema della derivabilità $x \underset{G}{\overset{*}{\Rightarrow}} y$	è indecidibile
$\mathcal{L}_1$	il problema della derivabilità $x \underset{G}{\overset{*}{\Rightarrow}} y$	è decidibile[17]
	il problema $L(G) = \varnothing$	è indecidibile

[17] Cfr. Teorema 9.5

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

8.6 Esercizi proposti

Esercizio 1

Siano dati i seguenti linguaggi:

$L = \{ a^i b^j c^k d^l : i + k = j + l, i, j, k, l > 0 \}$

Costruire un automa a pila che accetta L.

Esercizio 2

Siano dati i seguenti linguaggi:

s) $\{ a^m b^n : m \leq n \leq 2m, m, n \geq 1 \}$

t) $\{ a^m b^n : m, n$ interi non negativi, $n < m$ OR $2m < n \}$

u) $\{ a^i b^j c^k : k = i - j, i, j, k > 0 \}$

Costruire un automa a pila che accetta L.

Determinare una grammatica libera da contesto che genera L.

Esercizio 3

Sia L il linguaggio costituito da tutte le parole sull'alfabeto $\{a, b, \lambda, +, *, \cdot, \varnothing, (,)\}$ che sono espressioni regolari (ben formate) di alfabeto $X = \{a, b\}$.

Determinare una grammatica libera da contesto che genera L.

Dimostrare formalmente che G è corretta per L ($L = L(G)$).

Costruire un automa a pila che accetta L.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Esercizio 4

Stabilire a quale classe di linguaggi della gerarchia di Chomsky appartengono i seguenti linguaggi e giustificare formalmente la risposta:

$\{ a^i b^j c^k : i \leq k \text{ or } j \leq k, i, j, k > 0 \}$

$\{ w^R w : w \in \{a, b, c\}^* \}$

$\{ ww : w \in \{a, b, c\}^* \}$

Esercizio 5

Sia dato il seguente linguaggio:

$L = \{ a^n b^n c^n : n > 0 \} \cdot \{ b^n a^n c^n : n > 0 \}$

Stabilire a quale classe di linguaggi della gerarchia di Chomsky appartiene L.

Giustificare formalmente la precedente risposta.

Determinare una grammatica che genera L.

Esercizio 6

Sia data la seguente grammatica:

$G = (X, V, S, P)$

ove $X = \{a, b, c, d, e, 0, 1\}$

$V = \{S, A, B, C, D, E\}$

$P = \{ \quad S \rightarrow AD,$

$A \rightarrow aA \mid e,$

$D \rightarrow Bc \mid Cd,$

$B \rightarrow bB \mid 1,$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

$C \to 0E,$

$E \to dE \mid e \quad \}$

Descrivere il linguaggio $L(G)$ generato da G.

Stabilire a quale classe di linguaggi della gerarchia di Chomsky appartiene $L(G)$.

Giustificare formalmente la precedente risposta.

Esercizio 7

Convertire la seguente grammatica:

$G = (X, V, S, P)$

ove $X = \{0, 1, 2\}$

$V = \{S, A, B\}$

$P = \{ \quad S \to A2B,$

$A \to 0A,$

$\mathrm{A} \to \lambda,$

$B \to 1B,$

$B \to \lambda \ \}$

in forma normale di Chomsky.

Esercizio 8

Sia data la seguente grammatica libera da contesto:

$G = (X, V, S, P)$

ove $X = \{a, b\}$

$V = \{S, A, B\}$

$P = \{ \quad S \to bA \mid aB,$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

$A \to bAA \mid aS \mid a,$

$B \to aBB \mid bS \mid b \qquad \}$

Descrivere il linguaggio $L(G)$ generato da G.

Convertire G in forma normale di Chomsky.

Esercizio 9

Sia data la grammatica:

$G = (X, V, S, P)$

ove $\quad X = \{a, b, c\}$

$V = \{S, A, B\}$

$P = \{ \quad S \to cABc,$

$A \to aAa \mid c,$

$B \to bBb \mid c \quad \}$

Costruire l'albero di derivazione per la parola $w = cacabcbc$.

Convertire G in forma normale di Chomsky.

Esercizio 10

Sia data la seguente grammatica libera da contesto:

$G = (X, V, S, P)$

ove $\quad X = \{a, b, c, d\}$

$V = \{S, A\}$

$P = \{ \quad S \to aSb \mid cAd,$

$A \to S \mid \lambda \qquad \}$

Costruire l'albero di derivazione per la parola $w = aacacdbdbb$.

Convertire G in forma normale di Chomsky.

Esercizio 11

Sia data la grammatica:

$G = (X, V, S, P)$

ove $X = \{0, 1, 2\}$

$V = \{S, A, B, C, D, E, F, G, H, I\}$

$$
\begin{aligned}
P = \{ \quad & S \to GC, \\
& A \to HD \mid 2, \\
& B \to IE \mid 2, \\
& C \to 2, \\
& D \to 0, \\
& E \to 1, \\
& F \to CA, \\
& H \to DA, \\
& I \to EB, \\
& G \to FB \qquad \}
\end{aligned}
$$

Costruire l'albero di derivazione per la parola $w = 20201212$.

Convertire G in forma normale di Greibach.

Esercizio 12

Sia data la seguente grammatica libera da contesto:

$G = (X, V, S, P)$

ove $X = \{a, b, c\}$

$V = \{S, A, B, C\}$

$$
\begin{aligned}
P = \{ \quad & S \to AB, \\
& A \to CC \mid a,
\end{aligned}
$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

$B \to b,$

$C \to AA \mid c \quad \}$

Convertire G in forma normale di Greibach.

Costruire un automa a pila che riconosce $L(G)$.

Esercizio 13

Sia data la seguente grammatica libera da contesto:

$G = (X, V, S, P)$

ove $X = \{a, b\}$

$V = \{ S, A, B \}$

$P = \{ \quad S \to AB,$

$A \to BS \mid b,$

$B \to SA \mid a \quad \}$

Convertire G in forma normale di Greibach.

Costruire un automa a pila che riconosce $L(G)$.

Esercizio 14

Sia data la seguente grammatica libera da contesto:

$G = (X, V, S, P)$

ove $X = \{0, 1, 2, 3, 4, a, b\}$

$V = \{S, A, B, C, D, E\}$

$P = \{ \quad S \to AD,$

$A \to 0A \mid 4,$

$D \to B2 \mid C3,$

$B \to B1 \mid b,$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

$C \to aE,$

$E \to E3 \mid 4 \quad \}$

Convertire G in forma normale di Greibach.

Costruire un automa a pila che riconosce $L(G)$.

Esercizio 15

Sia data la seguente grammatica:

$G = (X, V, S, P)$

ove $X = \{0, 1\}$

$V = \{S, A, B, C\}$

$P = \{ \; S \to AB, \quad S \to CA,$

$B \to BC, \quad B \to AB,$

$A \to 0,$

$C \to 0B, \quad C \to 1 \; \}$

Determinare una grammatica G' equivalente a G e priva di nonterminali inutili.

9. Cenni su automi limitati lineari e macchine di Turing

9.1 Automi limitati lineari

Definizione 9.1 (Automa limitato lineare o LBA)

Un *automa limitati lineare* (Linear Bounded Automaton) è una macchina di Turing non deterministica in cui la testina si mantiene all'interno dello spazio (porzione finita di nastro) contenente inizialmente l'input (la parola da analizzare)

Lemma 9.1 (Lemma dell'alfabeto)

Sia L un linguaggio tale che $\exists$ m.d.T. $T=(Q,X,\delta,q_0,q_F)$, $L=L(T)$ e T usa $k\cdot|w|$ celle per riconoscere $w\in X^*$.

Allora $\exists$ m.d.T. $T=(Q,X,\delta',q_0',q_F')$ tale che $L=L(T')$ e T' usa $|w|$ celle per riconoscere $w\in X^*$ (la potenza delle macchine di Turing non cambia se la lunghezza del nastro è proporzionale a $|w|$).

Teorema 9.2 ($\mathcal{L}_{LBA}\subseteq\mathcal{L}_1$)

Sia $T=(Q,X,\delta,q_0,q_F)$ un LBA, allora

$$L=L(T)\ \Rightarrow\ \exists G(X,V,S,P)\ \text{di Tipo 1}: L=L(G)$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Dimostrazione 9.2

Supponiamo $\lambda \notin L$ (in caso contrario si dimostrerebbe il Teorema per $L - \{\lambda\}$ e poi si aggiungerebbe $S \to \lambda$ in G).

Supponiamo che l'input sul nastro dell'LBA sia limitato a sinistra da \$ e a destra da ¢.

	~~¢~~	\$	a	b	b	b	a	a	b	¢	

Sfruttiamo le singole transizioni dell'LBA per scrivere le produzioni (di una grammatica) che riscrivono le configurazioni istantanee di *T* nelle configurazioni istantanee successive.

Scriviamo dapprima una $G = (X, V, S, P)$ di Tipo 1 per il linguaggio:

$$\left\{ w \$ q_0 \, w ¢ \mid w \in X^* \right\}$$

ove

$w \in X^*$ è una stringa qualsiasi sull'alfabeto di *T*.

\$ è un simbolo terminale per *G*.

¢ è un simbolo non terminale per *G*.

A destra si \$ c'è $q_0 w$, la configurazione iniziale della computazione su *w*.

(i) Se $\begin{array}{c} (q_j, c, D) \in \delta(q_0, a) \\ < q_0, a, q_j, c, D > \end{array}$ poniamo $q_0 a \to c q_j \in P$

(ii) $< q_i, a, q_j, c, S >$ poniamo $b q_i a \to q_j b c \in P \;\; \forall b \in P$

(iii) $< q_i, a, q_j, c, F >$ poniamo $q_i a \to q_j c \in P$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Alla fine, quando appare q_F (stato di accettazione), ci comportiamo come segue:

(1) Convertiamo i simboli a destra di \$ in un nuovo simbolo "/", così da avere $w\$/^n q_F ¢$, con produzioni del tipo $q_F\, a \to /\, q_F$ e $a\, q_F \to q_F\, /$.

(2) Se $w = x_1 x_2 \ldots x_n$, usiamo una produzione di Tipo 1 per costruire la stringa: $r\, /\, x_1\, /\, x_2\, / \ldots /\, x_n\, /\# q$ (r, q nuovi simboli terminali).

(3) Cancelliamo tutte le occorrenze di r, /, #, q.

Quindi w a sinistra di \$ serve per ricordare quale parola deve essere generata da G.

c.v.d.

Vale anche il contrario

Teorema 9.3

$\mathcal{L}_1 \subseteq \mathcal{L}_{LBA}$

Dimostrazione 9.3

Sia $L = L(G)$ ove $G = (X, V, S, P)$ di Tipo 1. Vogliamo costruire $T = (Q, X, \delta, q_0, q_F)\ LBA : L(T) = L$.

L'equivalenza tra LBA deterministici ed LBA non deterministici aiuta molto.

La δ è fatta in modo che T analizza (sintatticamente) una parola w sul nastro in maniera non deterministica.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

La configurazione iniziale è $q_0 w$, cioè q_0 è lo stato iniziale e la testina è sul primo carattere di *w*. La testina si muove da sinistra verso destra e non appena scopre che una sequenza di simboli corrisponde alla parte destra di una produzione, fa una "riduzione", ossia inserisce la parte sinistra della produzione nel punto corrispondente sul nastro. Poiché l'analisi è non deterministica, si devono prevedere anche transizioni che continuano ad elaborare la stringa come se la produzione non fosse mai stata applicata. Se alla fine sul nastro rimane *S*, l'LBA passa in q_F.

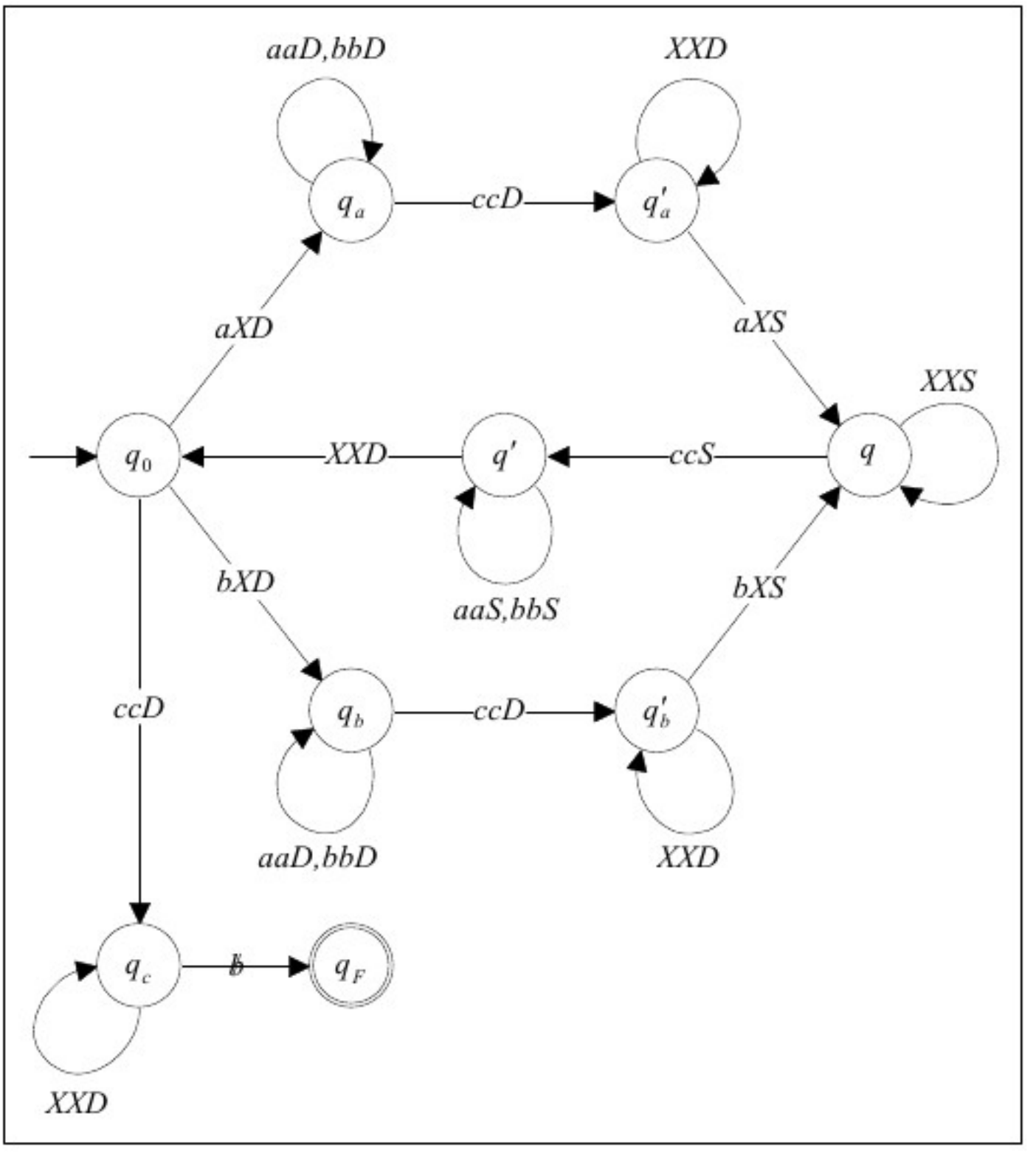
c.v.d.

Esempio 9.1

$L = \left\{ zcz \mid z \in \{a,b\}^* \right\}$ è un linguaggio di Tipo 1.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Un LBA T tale che $L(T) = L$ è rappresentato dal diagramma:



Dunque $\mathcal{L}_1 \equiv \mathcal{L}_{LBA}$.

9.2 Insiemi ricorsivi

Che rapporto esiste tra $\mathcal{L}_1$ e gli insiemi ricorsivi?

Vale il seguente

Teorema 9.4

$\mathcal{L}_{LBA} \underset{\neq}{\subset} Insiemi\ Ricorsivi$

Dimostrazione 9.4

La dimostrazione prevede due fasi: prima $\mathcal{L}_{LBA} \subseteq Insiemi\ Ricorsivi$, poi $\mathcal{L}_{LBA} \neq Insiemi\ Ricorsivi$.

1) $\mathcal{L}_{LBA} \subseteq Insiemi\ Ricorsivi$

Sia $L \in \mathcal{L}_{LBA} \overset{def}{\Leftrightarrow} \exists T = (Q, X, \delta, q_0, q_F)\ LBA : L(T) = L$.

Ts. L è ricorsivo $\overset{def}{\Leftrightarrow} c_L(x) = \begin{cases} 1 & x \in L \\ 0 & x \notin L \end{cases}$ è computabile.

Formalmente la funzione caratteristica di L $c_L : X^* \to \aleph$ e non $\aleph \to \aleph$, quindi devo "codificare" il dominio X^* delle parole su X con una funzione iniettiva $\alpha : X^* \to \aleph$ (α = ordinamento lessicografico) in modo che $c_L^*(x) \cong c_L \cdot \alpha^{-1}$.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Sia $w \in X^*$, se $w \in L$ allora $q_0\, w \overset{*}{\underset{T}{\vdash}} v\, q_F\, z$

se $w \in X^* - L$ posso osservare che un LBA può passare attraverso un numero finito di configurazioni istantanee distinte. Dunque modifico T in modo che <u>si fermi</u> non appena ritorna in una configurazione istantanea già incontrata.

Dunque $\mathcal{L}_{LBA} \subseteq$ *Insiemi Ricorsivi* .

2) $\mathcal{L}_{LBA} \neq$ *Insiemi Ricorsivi*

Basta considerare $L = \{x_i \mid x_i \notin L(G_i)\} \in$ *Insiemi Ricorsivi* $\setminus \mathcal{L}_{LBA}$

e usare l'equivalenza $\mathcal{L}_{LBA} \equiv \mathcal{L}_1$.

Sia $G = (X, V, S, P)$ una grammatica di Tipo 1.

Suppongo $|X| = 2$: ogni grammatica G può essere codificata in un alfabeto binario (stringa di o e 1) come segue:

$X = \left\{ \overset{a}{00}, \overset{b}{001} \right\}$ i terminali iniziano per 00

$V = \left\{ \overset{S}{01}, \overset{A}{011}, 0111, \ldots \right\}$ i NT iniziano per 0

$S = 01$

$P = v \to w \quad v, w \in (X \cup V)^*$ sono già codificati

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

$'\rightarrow' = 0011$

e ho bisogno di "separare" le produzioni tra loro: $'separatore' = 00111$.

Quindi se $\alpha \in (X \cup V)^*$, $\alpha \in \left\{00, 001, 0(1)^+\right\}^*$.

La codifica e la decodifica di una G sono univoche ed effettive.

Le parole su $\{0,1\}$ si possono enumerare effettivamente senza ripetizioni attraverso l'ordinamento lessicografico

$$0,\ 1,\ 00,\ 01,\ 10,\ 11,\ \ldots$$

$\exists$ una bigezione $\{0,1\}^* \rightarrow \aleph$ computabile w la sua inversa è computabile.

G_i di Tipo 1

1	$\lvert G_1 \rvert$	$\lvert G_1 \rvert = x_1 \in \{0,1\}^*$
2	$\lvert G_2 \rvert$	$\lvert G_2 \rvert = x_2 \in \{0,1\}^*$
$\vdots$	$\vdots$	$\vdots$
i	$\lvert G_i \rvert$	$\lvert G_i \rvert = x_i \in \{0,1\}^*$

<u>Esempio</u>

$\overset{01}{S} \overset{0011}{\rightarrow} \overset{011}{A}$	$\overset{00111}{\text{separatore}}$	viene	01:0011:011:00111:
$\overset{011}{A} \overset{0011}{\rightarrow} \overset{00}{a} \overset{0111}{B}$	$\overset{00111}{}$	codificato	011:0011:00:0111:00111:
$\overset{0111}{B} \overset{0011}{\rightarrow} \overset{001}{b}$	$\overset{00111}{}$	come	0111:0011:001:00111

Quindi, dato un intero i, posso ritrovare in modo univoco la i-esima stringa di $\{0,1\}^*$ che la rappresenta.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Definiamo:

$$L = \{x_i \mid x_i \notin L(G_i),\ \forall i > 0\} \text{ (metodo diagonale)}$$

Data una parola x_i, posso trovare la sua posizione i e determinare G_i e poi decidere se $x_i \in L(G_i)$ (il problema della derivabilità di una parola è decidibile in $\mathcal{L}_1$, vedi Teorema 9.5). Quindi L è ricorsivo, ma non è dipendente da contesto, infatti

se $L \in \mathcal{L}_1 \overset{def}{\Leftrightarrow} \exists\, j : L = L(G_j)$ con G_j j-esima grammatica di Tipo 1.

Ma allora se considero x_j = j-esima stringa, si avrebbe:

se $x_j \in L \Rightarrow x_j \in L(G_j) = L$	e, per definizione di L,	$\Rightarrow x_j \notin L(G_j)$ contraddizione
se $x_j \notin L \Rightarrow x_j \notin L(G_j) = L$	e, per definizione di L,	$\Rightarrow x_j \in L(G_j)$ contraddizione

Ho costruito L con il metodo diagonale:

"Fa sì che $L \neq L(G_n)$ in x_n $\forall n$"

c.v.d.

Il precedente risultato dipende dal seguente:

Teorema 9.5 (Decidibilità della derivabilità per grammatiche di Tipo 1)

Il problema '$w \in L(G)$' è decidibile per G di Tipo 1.

Dimostrazione 9.5

Sia $L \in \mathcal{L}_1 \overset{def}{\Leftrightarrow} \exists G \text{ Tipo } 1 : L = L(G)$.

Poiché $\mathcal{L}_1 \equiv \mathcal{L}_{monotone}$, considero $L = L(G)$ con G monotona.

Dunque per $\alpha, \beta \in (X \cup V)^*$, se $|\alpha| > |\beta|$ allora $a \underset{G}{\overset{*}{\not\Rightarrow}} \beta$, mentre se $|\alpha| \leq |\beta|$ allora esiste un numero finito di forme di frase di lunghezza k tali che $|\alpha| \leq k \leq |\beta|$. Supponiamo che tali forme di frase siano n, allora applico ad α tutte le regole di G in ogni possibile ordine. Si hanno 3 possibilità:

(a) $\alpha \underset{G}{\overset{t}{\Rightarrow}} \beta$, con t finito e rispondo SI;

(b) $\alpha \underset{G}{\overset{t}{\Rightarrow}} \gamma$ e $|\gamma| > |\beta|$, allora termino e rispondo NO;

(c) $\alpha \underset{G}{\overset{t_1}{\Rightarrow}} \gamma \underset{G}{\overset{t_2}{\Rightarrow}} \gamma$, cioè ritorno a produrre una forma di frase, allora termino e rispondo NO.

In ogni caso raggiungo lo stato finale.

c.v.d.

9.3 Insiemi ricorsivamente enumerabili

Teorema 9.6

$\mathcal{L}_0 \equiv \mathcal{L}_{MdT}$.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Dimostrazione 9.6 (Simile a quella per $\mathscr{L}_1 \equiv \mathscr{L}_{LBA}$)

Osservazione preliminare: le macchine di Turing hanno nastro, testina di lettura/scrittura e non possono essere descritte con l'apparato della teoria dei linguaggi formali. D'altro canto, le configurazioni istantanee sono stringhe di simboli e, come tali, possono essere manipolate da una grammatica formale.

$\mathscr{L}_{MdT} \subseteq \mathscr{L}_0$)

Sia $T = (Q, X, \delta, q_0, q_F)$ una m.d.T. : $L = L(T)$.

Sfruttiamo il carattere estremamente locale delle transizioni (mosse) di una m.d.T. per scrivere le produzioni di una grammatica che riscrivono una configurazione istantanea di T nella configurazione istantanea successiva.

Come fatto per $\mathscr{L}_{LBA} \subseteq \mathscr{L}_1$, iniziamo scrivendo le produzioni di una G di Tipo 1 per l'insieme di stringhe

$$w\$q_0w¢$$

dove

$w \in X^*$;

\$ è un nuovo simbolo terminale (speciale) di G;

¢ è un nuovo simbolo nonterminale (speciale) di G;

La stringa a destra di \$ rappresenta la configurazione iniziale della computazione su i.

Vogliamo scrivere le regole di produzione di G che corrispondono alle transizioni della macchina.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

(i) Se $\begin{array}{c}(q_j, c, D) \in \delta(q_0, a) \\ < q_0, a, q_j, c, D >\end{array}$ allora $q_0\, a \to c\, q_j \in P$

(ii) Se $< q_i, a, q_j, c, S >$ allora $\begin{array}{r} b\, q_i\, a \to q_j\, b\, c \in P \\ \forall\, b \in P \end{array}$

(iii) Se $< q_i, a, q_j, c, F >$ allora $q_i\, a \to q_j\, c \in P$

Poi, abbiamo bisogno di regole di produzione in G che estendono la porzione attiva del nastro.

(iv) $\$\, q_i \to \$\, b\, q_i$

(v) $q_i\, ¢ \to q_i\, b\, ¢$

Se la macchina si ferma, dobbiamo ripulire il nastro così da avere solo la stringa accettata.

Supponiamo che w sia accettata da T e che lo stato q_F sia derivato come parte di qualche forma di frase, cioè q_F si trova a destra di \$.

Aggiungiamo delle produzioni in G per "inghiottire" tutti gli altri simboli a destra di \$.

(vi) $q_F\, a \to q_F \quad \forall a \in X$

(vii) $a\, q_F \to q_F \quad \forall a \in X$

(viii) $\$\, q_F\, ¢ \to \lambda$

Si ha che,

se w porta T a fermarsi in q_F, le produzioni precedenti ci permettono di riscrivere $S \overset{*}{\underset{G}{\Rightarrow}} w\$\, q_0\, w¢$ attraverso le

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

produzioni che generano $\{w\$q_0\,w¢ \mid w \in X^*\} \overset{*}{\Rightarrow}$ tramite (i)-(v) $\overset{*}{\underset{G}{\Rightarrow}}\; w\$w_1\,q_F\,w_2¢ \underbrace{\overset{*}{\underset{G}{\Rightarrow}}}_{\substack{\text{tramite}\\ \text{(vi)}-\text{(viii)}}} w$ e quindi $w \in L(G)$;

se w porta T a non raggiungere mai lo stato q_F, le produzioni precedenti ci permettono di riscrivere $S \overset{*}{\underset{G}{\Rightarrow}} w\$w_1\,q'\,w_2¢ \overset{*}{\underset{G}{\Rightarrow}} \alpha¢$ (il simbolo ¢ rimane all'estrema destra comunque) e quindi $w \notin L(G)$.

$\mathcal{L}_0 \subseteq \mathcal{L}_{MdT}$) (simile a $\mathcal{L}_1 \subseteq \mathcal{L}_{LBA}$)

Sia $L \in \mathcal{L}_0 \overset{def}{\Leftrightarrow} \exists G$ di Tipo $0 : L = L(G)$.

Costruisco T in modo che accetti L.

Le transizioni in δ fanno sì che T analizzi sintatticamente w sul nastro.

$q_0 w$ configurazione iniziale per T

La testina si muove da sinistra a destra. Non appena scopre una sequenza di simboli che corrisponde alla parte destra di una produzione fa una "riduzione" sostituendo ad essa la parte sinistra. L'analisi è inerentemente non deterministica, ossia devo sempre prevedere mosse che continuano l'elaborazione come se la "riduzione" non fosse stata operata.

Se alla fine sul nastro rimane S, posso accettare passando in q_F.

c.v.d.

Che rapporto esiste tra $\mathcal{L}_0$ e Insiemi ricorsivamente enumerabili?

Teorema 9.7

$\mathcal{L}_0 \equiv$ *Insiemi r. e.*

Dimostrazione 9.7

$\mathcal{L}_0 \subseteq$ *Insiemi r. e.*)

Sia $L \in \mathcal{L}_0 \equiv \mathcal{L}_{MdT} \underset{\substack{\text{per il}\\ \text{Teorema}\\ \text{di Equiv.}}}{\Leftrightarrow} \exists T = (Q, X, \delta, q_0, q_F) : L(T) = L$ con

T m.d.T. e quindi $L(T) \overset{def}{=} \left\{ w \in X^* \;\middle|\; q_0\, w \overset{*}{\underset{T}{\vdash}} v\, q_F\, w' \right\} = L$.

Ts. L è ricorsivamente enumerabile $\overset{def}{\Leftrightarrow} f(x) = \begin{cases} 1 & x \in L \\ indef & x \notin L \end{cases}$ (la funzione caratteristica parziale) è computabile.

Formalmente la funzione caratteristica di L $f : X^* \to \aleph$ e non $\aleph \to \aleph$, quindi devo "codificare" il dominio X^* delle parole su X con una funzione iniettiva $\alpha : X^* \to \aleph$ (α = ordinamento lessicografico) in modo che $f^* = f \circ \alpha^{-1} : \aleph \to \aleph$. Devo dimostrare che f^* è computabile.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Sia $w \in X^*$,

$$\Phi_T(w) = \begin{cases} z & q_0\, w \overset{*}{\underset{T}{\vdash}} v\, q_F\, w' \ \text{ e } \ v\, w' = z \\ indef & \qquad altrimenti \end{cases}$$

Quindi: se $w \in L : \Phi_T(w) = z,\ z \in X^*$

se $w \notin L : \Phi_T(w)$ non è definita.

Allora f è computabile attraverso una m.d.T. T' che si comporta come T e, quando T si ferma, T' cancella z e lascia sul nastro / (codifica in alfabeto unario di 1), mentre se T non si ferma, T' non si ferma.

Per la Tesi di Church, segue che f è computabile ed L è ricorsivamente enumerabile.

Insiemi r. e. $\subseteq \mathcal{L}_0$)

Sia L un insieme r.e. $\overset{def}{\Leftrightarrow}$ $f(x) = \begin{cases} 1 & x \in L \\ indef & x \notin L \end{cases}$ è computabile.

$L \subseteq \aleph \qquad\qquad f : \aleph \to \aleph$.

Ts. Voglio costruire $T = (Q, X, d, q_0, q_F)$ m.d.T. : $L(T) = L$.

Ma $L \subseteq \aleph$, $\aleph$ è infinito, mentre X è finito.

Devo codificare i numeri interi in un alfabeto X.

Suppongo $X = \{/\}$, quindi

0 è codificato con	$/$
1	$//$
$\vdots$	$\vdots$
n	$\underbrace{//\ldots/}_{n+1}$

X^* è equipotente ad $\aleph$ (è effettivamente numerabile).

Con questa codifica, poiché f è computabile, per la Tesi di Church, deve $\exists$ T m.d.T. che la computa.

$$T = (Q, X, d, q_0, q_F) \text{ tale che } \Phi_T \cong f .$$

c.v.d.

<u>Esempio 9.2</u>

Dimostrare che

$$L_1 = \left\{ x_i \in X^* \mid x_i \in L(T_i) \right\}$$

è un linguaggio ricorsivamente enumerabile ma non ricorsivo.

La dimostrazione si basa sul fatto che $\tau = \{T \mid T \text{ m.d.T.}\}$ è effettivamente numerabile.

Posso infatti <u>codificare</u> ogni m.d.T. T con una parola su $\{0,1\}$.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

In particolare, consideriamo la quintupla di una m.d.T. in cui $X = \{/, b\}$ $(q_i, /, q_j, b, D)$ e poniamo:

$$\begin{array}{ll}
\underline{01}0000 & 0 \\
\underline{01}0001 & 1 \\
\vdots & \vdots \\
\underline{01}1001 & 9 \\
\underline{11}0000 & q \\
\underline{11}0001 & / \\
\underline{11}0010 & b \\
\underline{111}000 & S \\
\underline{111}001 & D \\
\underline{111}010 & F
\end{array}$$

e quindi, per esempio, $(q_2, /, q_{11}, b, S)$ diventa:

$$\underbrace{110000}_{q} : \underbrace{010010}_{2} : \underbrace{110001}_{/} : \underbrace{110000}_{q} : \underbrace{010001}_{1} : \underbrace{010001}_{1} : \underbrace{111000}_{S}$$

Quindi le m.d.T. T_i sono effettivamente enumerabili, nel senso che:

- dato un intero i (in binario), possiamo determinare univocamente la m.d.T. T_i (che ha come codifica la stringa i) cioè $T_i = \gamma^{-1}(i)$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

- data T_i (la m.d.T.), possiamo determinare univocamente la stringa su $\{0,1\}$ che rappresenta $i = \gamma(T)$

$$\begin{array}{lll} 1 & |T_1| & x_1 \in \{0,1\}^* \\ 2 & |T_2| & x_2 \in \{0,1\}^* \\ \vdots & & \vdots \\ i & |T_i| & x_i \in \{0,1\}^* \end{array}$$

Data una parola x_i su $\{0,1\}$, posso trovare la sua posizione i e determinare la m.d.T. T_i che, con input x_i, si ferma se $x_i \in L(T_i)$ ma non si ferma se $x_i \notin L(T_i)$.

Quindi

$$f(x_i) = \begin{cases} 1 & x_i \in L_i \\ indef & x_i \notin L_i \end{cases} = \begin{cases} 1 & x_i \in L(T_i) \\ indef & x_i \notin L(T_i) \end{cases}$$

è computabile attraverso la m.d.T. Universale T_U che prende in ingresso la codifica $|T_i|$ della m.d.T. T_i e la stringa x_i. T_U processa ogni stringa $(|T_i|, x_i)$ proprio come T_i processerebbe x_i.

Schematicamente: se $x_i \overset{*}{\underset{T_i}{\vdash}} x_i'$ allora $(|T_i|, x_i) \overset{*}{\underset{T_U}{\vdash}} x_i'$.

Quindi $f(x_i) = 1 \circ (\Phi_U(|T_i|, x_i))$, dove *1* lascia una / sul nastro.

$L_1 = \{x_i \mid x_i \in L(T_i)\} = \{i \mid \Phi_{T_i}(i) \downarrow\}$ (‘ $i \in Dom\ \Phi_{T_i}$ ’) non è ricorsivo perché se considero $L_2 = \{x_i \mid x_i \notin L(T_i)\}$, L_2 è non r.e. (vedi Esempio 9.3). Se L_2 non r.e. allora $\overline{L_2}$ non ricorsivo (se lo fosse, L_2 sarebbe ricorsivo), ma $\overline{L_2} = L_1$ e quindi L_1 non ricorsivo.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

9.4 Problema della derivabilità per grammatiche Tipo 0

Definizione 9.2 (Problema della derivabilità)

Sia $G = (X, V, S, P)$ una grammatica a struttura di frase. Il problema della derivabilità in $G \overset{def}{\Leftrightarrow} v \overset{*}{\underset{G}{\Rightarrow}} w,\ \ v, w \in (X \cup V)^*$ (date due stringhe su $(X \cup V)$, stabilire se w è derivabile da v in G).

Teorema 9.8

Il problema '$v \overset{*}{\underset{G}{\Rightarrow}} w$' è indecidibile se G è di Tipo 0.

Dimostrazione 9.8

Lo schema di dimostrazione è basato sulla seguente riduzione (molti-a-uno):

$$\text{'}S \overset{*}{\underset{G}{\Rightarrow}} z\text{'} \leq_m \text{'}v \overset{*}{\underset{G}{\Rightarrow}} w'\text{'}$$

Ts. $\quad S \overset{*}{\underset{G}{\Rightarrow}} z$ è indecidibile.

Quindi, poiché $\mathcal{L}_0 \equiv \mathcal{L}_{MdT}$, il problema è equivalente a quello di stabilire se la m.d.T. M corrispondente a G si ferma con input z (problema della fermata di una m.d.T.). Quindi costruisco M a partire da G con le transizioni necessarie a "ridurre" le parti di nastro che coincidono con le parti destre di produzioni, alle relative parti sinistre.

Ma il problema della fermata di una m.d.T. è indecidibile, quindi '$v \overset{*}{\underset{G}{\Rightarrow}} w$' è indecidibile.

<u>Esempio 9.3</u>

Dimostrare che

$$L_2 = \{x_i \mid x_i \notin L(T_i)\}$$

è un linguaggio non ricorsivamente enumerabile.

Si può dimostrare per assurdo, infatti se L_2 fosse ricorsivamente enumerabile, allora, poiché $\mathcal{L}_{\mathrm{MdT}} \equiv$ *Insiemi r. e.*,

$$\exists T_j = (Q, X, \delta, q_0, q_F) \;\; m.d.T. : L(T_j) = L_2 .$$

Ma allora, presa x_j, codificata su $\{0,1\}$ di T_j, si avrebbero due possibili casi:

se $x_j \in L_2 \underset{def\ di\ L_2}{\Rightarrow} x_j \notin L(T_j) \underset{poiché\ L(T_j)=L_2}{\Rightarrow} x_j \notin L_2$ contraddizione;

se $x_j \notin L_2 \underset{def\ di\ L_2}{\Rightarrow} x_j \in L(T_j) \underset{poiché\ L(T_j)=L_2}{\Rightarrow} x_j \in L_2$ contraddizione.

In entrambi i casi si è giunti ad una contraddizione derivante dall'aver supposto L_2 ricorsivamente enumerabile.

c.v.d.

Relazioni tra Insiemi, Macchine, Linguaggi

Insiemi		Macchine		Linguaggi
Regolari	=	Automi a stati finiti (FSA) (det. e nondet. $\mathcal{L}_{FSL} = \mathcal{L}_{NDL}$, conesenza λ - transizioni, unidirezionali e bidirezionali) (1)	=	$\mathcal{L}_3 = \{L(G) \mid G \text{ Tipo } 3\}$
	potenza delle macchine	$\subsetneq$	inclusione di classi di linguaggi	
		Automi a (una) pila (PDA) deterministici (2)		$\mathcal{L}_{2\,det}$ (Ling. C.F. det.)
		$\subsetneq$		
		Automi a (una) pila (PDA) nondeterministici $\mathcal{L}_{PDA} = \mathcal{L}_{PDA_{1\,state}}$ (3)	=	$\mathcal{L}_2 = \{L(G) \mid G \text{ Tipo } 2\}$
		$\subsetneq$		
		Automi limitati lineari (LBA) (inerent. nondet.) (4)	=	$\mathcal{L}_1 = \{L(G) \mid G \text{ Tipo } 1\}$
		$\subsetneq$		
Ricorsivi (5)				
		$\subsetneq$		
Ricorsivamente enumerabili (6)	=	Macchine di Turing, URM, Automi con due pile	=	$\mathcal{L}_0 = \{L(G) \mid G \text{ Tipo } 0\}$
Non ricorsivamente enumerabili (7)				

312 - CENNI SU AUTOMI LIMITATI LINEARI E MACCHINE DI TURING

(1)
$$\{w \mid |w| = 4k, k \in \aleph\}$$
$$\{w \mid w \neq \alpha\ p2-1816\ \beta,\ \alpha, \beta \in X^*\}$$

(2)
$$\{a^n b^n \mid n > 0\}$$
$$\{wcw^R \mid w \in \{a,b\}^*\}$$
$$\{w \mid \#_a(w) = \#_b(w)\}$$

(3)
$$\{ww^R \mid w \in \{a,b\}^*\}$$

(4)
$$\{a^n b^n c^n \mid n > 0\}$$
$$\{wcw \mid w \in \{a,b\}^*\}$$
$$\{w\$q_0 w¢ \mid w \in X^*\}$$

(5)
$$L = \{x_i \mid x_i \notin L(G_i),\ G_i \text{ Tipo } 1, \forall i > 0\}$$

(6)
$$L = \{x_i \mid x_i \in L(T_i),\ T_i = i-\text{esima m.d.T.}, \forall i > 0\}$$
$$K = \{x \mid x \in Wx\}$$

(7)
$$L = \{x_i \mid x_i \notin L(T_i),\ T_i = i-\text{esima m.d.T.}, \forall i > 0\}$$

10. Analisi sintattica. Grammatiche LL(k) e LR(k)

10.1 Analisi sintattica (Parsing)

Data una grammatica libera da contesto (CFG) G ed una stringa w di caratteri terminali, analizzare sintatticamente w significa determinare se $w \in L(G)$ e, in tal caso, costruire l'albero sintattico di w (o gli alberi sintattici, qualora G sia ambigua).

Nel caso di linguaggi regolari, non si parla di analisi sintattica, ma soltanto di *riconoscimento* poiché la struttura delle frasi del linguaggio è nota a priori, trattandosi di alberi sintattici che crescono soltanto verso destra (o sinistra) se le regole sono del tipo $A \to aB \;\; (A \to Ba)$.

Il riconoscimento viene effettuato da un automa a stati finiti.

Generalmente, si è interessati all'analisi sintattica di linguaggi non contestuali, o addirittura di loro sottoclassi.

L'analisi di linguaggi più potenti, di tipo 1 o 0, è meno interessante, sia per la maggiore complessità cui si andrebbe incontro sia per la minore importanza applicativa.

Le caratteristiche degli algoritmi di analisi sintattica di interesse per il progettista di traduttori sintattici o compilatori sono: la velocità, la quantità di memoria occupata, la capacità di riconoscere ed elaborare errori sintattici, la generalità della classe

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

di linguaggi riconosciuti e la possibilità di costruzione automatica dell'analizzatore.

L'analisi sintattica può essere svolta da un automa a pila che, ogni volta che riconosce una produzione, emette il relativo numero d'ordine (trasduttore a pila).

Esempio 10.1

Si riconsideri la grammatica vista nell'esempio 8.6:

$$
\begin{array}{ll}
(1) & S \to 0SAB \\
(2) & S \to 1 \\
(3) & A \to 1A \\
(4) & A \to 1 \\
(5) & B \to 2B \\
(6) & B \to 2
\end{array}
$$

Il trasduttore a pila che effettua l'analisi sintattica di $L(G)$ si ottiene facilmente dall'automa a pila che effettua il riconoscimento di $L(G)$.

$$
\begin{aligned}
T &= (Q, X, \Gamma, \Omega, \delta, q_0, Z_0, F) = \\
&= (\{q_0\}, \{0,1,2\}, \{S, A, B\}, \{1,2,3,4,5,6\}, \delta, q_0, S, \varnothing)
\end{aligned}
$$

ove:

$$\delta : Q \times (X \cup \{\lambda\}) \times \Gamma \to 2^{Q \times \Gamma^* \times \Omega^*}$$

titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

$$\delta(q_0,0,S) = \{(q_0,SAB,1)\}$$
$$\delta(q_0,1,S) = \{(q_0,\lambda,2)\}$$
$$\delta(q_0,1,A) = \{(q_0,A,3),(q_0,\lambda,4)\}$$
$$\delta(q_0,2,B) = \{(q_0,B,5),(q_0,\lambda,6)\}$$

Durante il riconoscimento, in condizioni di pila vuota, di una stringa $x \in L(G)$, il trasduttore emette una stringa $\omega \in \Omega^*$ che rappresenta la sequenza delle regole applicate nella derivazione sinistra di *x*.

Per esempio, se x = 0111222, si ottiene ω = 123455.

Si è già detto che un analizzatore sintattico di importanza pratica deve essere veloce.

Nel caso dell'automa precedente si è in presenza di un automa non deterministico e ciò ha una influenza negativa sul tempo di analisi. Infatti, poiché non è possibile scegliere univocamente una transizione in ogni configurazione e, d'altra parte, poiché non è possibile elaborare contemporaneamente tutti i possibili tentativi di analisi conseguenti al non determinismo, in alcuni passi del processo di riconoscimento è necessario scegliere tra diverse alternative, col rischio che la scelta effettuata ad un certo istante risulti successivamente scorretta, e che quindi si debba ritornare indietro ad effettuare un'altra delle possibili scelte precedentemente inevase.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Per esempio, per la stringa x = 0111222, l'automa nella configurazione $(q_0, 11222, AB)$ potrebbe scegliere la transizione $\delta(q_0, 1, A) = (q_0, \lambda, 4)$, che genera la configurazione $(q_0, 1222, B)$ dalla quale non è definita alcuna mossa.

E' necessario ritornare alla configurazione precedente e scegliere la transizione alternativa $\delta(q_0, 1, A) = (q_0, A, 3)$.

Questo modo di procedere avanti-indietro (*backtracking*), caratteristico del comportamento non deterministico, è inaccettabile da un punto di vista pratico, poiché dà luogo a tempi di analisi sintattica che crescono in modo più che proporzionale con la lunghezza della stringa.

Nel campo della compilazione, in realtà, questi inconvenienti non sono molto frequenti, poiché le grammatiche che descrivono i convenzionali linguaggi di programmazione sono tali da consentire un'analisi sintattica deterministica.

Nel caso di un automa che ricostruisce l'albero sintattico della stringa dall'alto verso il basso e da sinistra a destra si parla di *analisi discendente* (equivale all'attraversamento dell'albero in ordine anticipato), in contrapposizione all'*analisi ascendente*, che ricostruisce l'albero sintattico dal basso verso l'alto (cioè in ordine differito).

Nell'analisi discendente l'algoritmo è predittivo, nel senso che quando si espande un non terminale A mediante la regola

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

$A \to A_1 A_2 \ldots A_k$ si predice che dovranno essere incontrate successivamente nella stringa da analizzare anche le sottostringhe derivate da $A_2 \ldots A_k$.

Al contrario, nell'analisi ascendente si effettuano riduzioni delle parti destre delle produzioni alle rispettive parti sinistre soltanto dopo che la parte destra è stata tutta esaminata.

Spesso il non determinismo degli analizzatori può essere eliminato se si consente la prospezione di due o più caratteri della stringa sorgente, prima di decidere la mossa dell'automa.

Nell'esempio precedente il non determinismo che si ha quando si legge 1 in ingresso e A è posto in cima alla pila, può essere sciolto se si leggono due caratteri in ingresso. Infatti, se il secondo carattere è un 1, deve essere applicata la transizione $\delta(q_0, 1, A) = (q_0, A, 3)$; altrimenti, se esso è un 2, si utilizza $\delta(q_0, 1, A) = (q_0, \lambda, 4)$.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

10.2 Grammatiche LL(*k*)

Una grammatica LL(*k*)[18] genera un linguaggio non contestuale la cui analisi può essere fatta da un algoritmo operante in modo discendente che:

a) scandisce la stringa corrente da sinistra verso destra;
b) costruisce la derivazione sinistra corrispondente all'albero sintattico;
c) opera deterministicamente esaminando il carattere sotto la testina di lettura e non più di *k*-1 caratteri successivi e consecutivi della stringa sorgente.

Definizione 10.1 ($First_k$)

Sia $\alpha = X_1 X_2 \ldots X_i,\ i \geq 0,\ \alpha \in (V \cup X)^*$.

Definiamo $First_k(\alpha),\ k \geq 0$, come segue:

$$First_k(\alpha) = \left\{ w \in X^* \;\middle|\; \begin{array}{ll} |w| < k \ \text{ e } \ \alpha \overset{*}{\Rightarrow} w & \text{oppure} \\ |w| = k \ \text{ e } \ \alpha \overset{*}{\Rightarrow} wz, & \text{per qualche } z \end{array} \right\}$$

Se $\alpha \in X^*$, allora:

$$First_k(\alpha) = \left\{ X_1 X_2 \ldots X_k \;\middle|\; |\alpha| > k \right\} \cup \left\{ \alpha \;\middle|\; |\alpha| \leq k \right\}$$[19]

[18] la sigla LL(*k*) indica che la stringa viene esaminata dalla sinistra – *Left to right* – e che viene ricostruita la derivazione sinistra – *Leftmost derivation*.

[19] si noti che i due insiemi sono disgiunti.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Definizione 10.2 (Grammatica LL(k))

Una CFG G è LL(k), $k \geq 1$, se l'esistenza di due derivazioni:

$$1) \quad S \overset{*}{\underset{lm}{\Rightarrow}} wA\alpha \underset{lm}{\Rightarrow} w\beta\alpha \overset{*}{\underset{lm}{\Rightarrow}} wx$$

$$2) \quad S \overset{*}{\underset{lm}{\Rightarrow}} wA\alpha \underset{lm}{\Rightarrow} w\gamma\alpha \overset{*}{\underset{lm}{\Rightarrow}} wy$$

con $First_k(x) = First_k(y)$, implica $\beta = \gamma$.

Se G è LL(k), la regola che espande A deve essere identica in 1) e 2).

Informalmente, G è LL(k) se, data una stringa $wA\alpha \in (V \cup X)^*$ ed i primi k simboli terminali (se esistono) derivati da $A\alpha$, esiste al più una produzione che può essere applicata ad A per fornire una derivazione di una stringa terminale che inizia con w ed è seguita da questi k terminali.

Osservazione

Se G è LL(k), G è a maggior ragione $LL(k')$, per $k' > k$.

Dunque si pone il problema di determinare il minimo k per cui G è LL(k).

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Dalla definizione precedente si deduce immediatamente il seguente teorema.

Teorema 10.1

Una grammatica LL(k) non contiene ricorsioni sinistre.

Esempio 10.2

Si consideri la grammatica:

$$A \to Ab$$
$$A \to a$$

esistono allora le derivazioni:

$$1) \quad A \overset{*}{\underset{lm}{\Rightarrow}} Ab^j \underset{lm}{\Rightarrow} ab^j, \qquad j \geq 0$$
$$2) \quad A \overset{*}{\underset{lm}{\Rightarrow}} Ab^j \underset{lm}{\Rightarrow} Ab^{j+1} \underset{lm}{\Rightarrow} ab^{j+1}$$

Supponiamo, per assurdo, che G sia LL(k); preso allora $j \geq k$, le sottostringhe b^j e b^{j+1} coincidono per i primi k caratteri.

Ma, per la definizione di grammatica LL(k), la regola che espande A dovrebbe essere identica in 1) e 2). Si ha dunque una contraddizione.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

10.3 Esercizi

Esercizio 10.1

Verificare che la seguente grammatica G è LL(1):

$$S \to aAS \mid b$$
$$A \to bSA \mid a$$

Intuitivamente G è LL(1) perché, detto *'C'* il non terminale più a sinistra in ogni stringa e *'c'* il successivo simbolo di ingresso, c'è al più una produzione per C capace di derivare una stringa di terminali per c.

In base alla definizione di LL(k), si ha:

$$S \overset{*}{\underset{lm}{\Rightarrow}} wS\alpha \underset{lm}{\Rightarrow} w\underbrace{aAS}_{\beta}\alpha \overset{*}{\underset{lm}{\Rightarrow}} w\underbrace{ax'}_{x}$$

$$S \overset{*}{\underset{lm}{\Rightarrow}} wS\alpha \underset{lm}{\Rightarrow} w\underbrace{b}_{\gamma}\alpha \overset{*}{\underset{lm}{\Rightarrow}} w\underbrace{by'}_{y}$$

E' dunque impossibile che esistano due derivazioni distinte che generano una stringa che inizia per wa (wb).

Specificamente, se x e y iniziano per a allora è stata utilizzata la produzione $S \to aAS$ e $\beta = \gamma = aAS$.

Se x e y iniziano per b, la produzione usata deve essere $S \to b$ e $\beta = \gamma = b$.

Si osservi che $x = y = \lambda$ è impossibile, dato che λ non è derivabile da S in G.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Analogo ragionamento si può fare quando consideriamo le due derivazioni:

$$S \overset{*}{\underset{lm}{\Rightarrow}} wA\alpha \underset{lm}{\Rightarrow} w\beta\alpha \overset{*}{\underset{lm}{\Rightarrow}} wx \quad \text{e}$$

$$S \overset{*}{\underset{lm}{\Rightarrow}} wA\alpha \underset{lm}{\Rightarrow} w\gamma\alpha \overset{*}{\underset{lm}{\Rightarrow}} wy$$

<u>Esercizio 10.2</u>

Verificare che la seguente grammatica è LL(2):

$$S \to \overset{(1)}{\lambda} \mid \overset{(2)}{abA}$$

$$A \to \overset{(3)}{Saa} \mid \overset{(4)}{b}$$

Innanzitutto verifichiamo che G non sia LL(1).

Per fare ciò, dobbiamo determinare due derivazioni:

$$1) \quad S \overset{*}{\underset{lm}{\Rightarrow}} wA\alpha \underset{lm}{\Rightarrow} w\beta\alpha \overset{*}{\underset{lm}{\Rightarrow}} wx$$

$$2) \quad S \overset{*}{\underset{lm}{\Rightarrow}} wA\alpha \underset{lm}{\Rightarrow} w\gamma\alpha \overset{*}{\underset{lm}{\Rightarrow}} wy$$

con $First_1(x) = First_1(y)$ e $\beta \neq \gamma$.

Consideriamo le seguenti derivazioni:

$$1) \quad S \underset{(2)}{\Rightarrow} abA \underset{(3)}{\Rightarrow} \underbrace{ab}_{w}\, S\, \underbrace{aa}_{\alpha} \underset{(2)}{\Rightarrow} \underbrace{ab}_{w}\, \underbrace{abA}_{\beta}\, \underbrace{aa}_{\alpha} \underset{(4)}{\Rightarrow} \underbrace{ab}_{w}\, \underbrace{abbaa}_{x}$$

$$2) \quad S \underset{(2)}{\Rightarrow} abA \underset{(3)}{\Rightarrow} \underbrace{ab}_{w}\, S\, \underbrace{aa}_{\alpha} \underset{(1)}{\Rightarrow} \underbrace{ab}_{w}\, \underbrace{\quad}_{\gamma=\lambda}\, \underbrace{aa}_{\alpha} \underset{(4)}{\Rightarrow} \underbrace{ab}_{w}\, \underbrace{aa}_{y}$$

Dunque si ha:

$$First_1(x) = First_1(abbaa) = \{a\} = First_1(aa) = First_1(y)$$

mentre $\beta = abA \neq \lambda = \gamma$; e quindi G non è LL(1).

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Mostriamo ora che G è LL(2).

Per far ciò, mostreremo che se $wB\alpha$ è una qualunque stringa ottenuta attraverso una *leftmost derivation* e wx è una parola di $L(G)$, allora esiste al più una produzione $B \to \beta$ in G tale che: $First_2(\beta\alpha) \supseteq First_2(x)$.

Supponiamo che $B = S$ e che esistano le due derivazioni:

$$1) \quad S \overset{*}{\underset{lm}{\Rightarrow}} wS\alpha \underset{lm}{\Rightarrow} w\beta\alpha \overset{*}{\Rightarrow} wx$$

$$2) \quad S \overset{*}{\underset{lm}{\Rightarrow}} wS\alpha \underset{lm}{\Rightarrow} w\gamma\alpha \overset{*}{\Rightarrow} wy$$

con $First_2(x) = First_2(y)$.

<u>Consideriamo α.</u>

E' immediato osservare che α deve essere una stringa di soli terminali.

Più precisamente, si ha una delle seguenti due condizioni:

o $\quad w = \alpha = \lambda$

oppure l'ultima produzione usata nella derivazione $S \overset{*}{\underset{lm}{\Rightarrow}} wS\alpha$ è stata la (3) $A \to Saa$ (altrimenti S non potrebbe comparire nella stringa $wB\alpha$) e dunque α comincia per aa.

Supponiamo che si utilizzi la produzione:

$$(1) \; S \to \lambda$$

per andare da $wS\alpha$ a $w\beta\alpha$.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Dunque: $\beta = \lambda$ ed x o è la stringa vuota ($x = \lambda$) oppure x comincia per aa.

Parimenti, se (1) $S \to \lambda$ è utilizzata per andare da $wS\alpha$ a $w\gamma\alpha$, allora $\gamma = \lambda$ ed $y = \lambda$ oppure y comincia per aa.

Supponiamo che si utilizzi la produzione:

$$(2)\ S \to abA$$

per andare da $wS\alpha$ a $w\beta\alpha$.

Dunque: $\beta = abA$ ed x inizia per ab.

Similmente, se (2) $S \to abA$ è utilizzata per andare da $wS\alpha$ a $w\gamma\alpha$, allora $\gamma = abA$ ed y inizia per ab.

Per riassumere, non vi sono che tre possibilità:

a) $x = y = \lambda$

b) x e y iniziano per aa

c) x e y iniziano per ab

Qualunque altra possibilità sui primi due simboli di x e y rende impossibile una delle due o entrambe le derivazioni.

<u>Casi a) e b):</u>

(1) $S \to \lambda$ è utilizzata in entrambe le derivazioni e $\beta = \gamma = \lambda$.

<u>Caso c):</u>

(2) $S \to abA$ dev'essere utilizzata in entrambe le derivazioni e $\beta = \gamma = abA$.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Supponiamo ora che nella generica stringa $wB\alpha$ sia $B = A$ e che esistano le due derivazioni:

$$1) \quad S \overset{*}{\underset{lm}{\Rightarrow}} wA\alpha \underset{lm}{\Rightarrow} w\beta\alpha \overset{*}{\underset{lm}{\Rightarrow}} wx$$

$$2) \quad S \overset{*}{\underset{lm}{\Rightarrow}} wA\alpha \underset{lm}{\Rightarrow} w\gamma\alpha \overset{*}{\underset{lm}{\Rightarrow}} wy$$

con $First_2(x) = First_2(y)$.

Si osserva immediatamente che l'ultima produzione utilizzata in $S \overset{*}{\underset{lm}{\Rightarrow}} wA\alpha$ è necessariamente la (2) $S \to abA$.

Dunque, per α, si ha certamente $\alpha \in X^*$ ed inoltre:

($w = ab$ e) $\alpha = \lambda$ se la (2) $S \to abA$ è l'unica produzione applicata, ossia: $S \underset{lm}{\Rightarrow} abA$.

oppure:

α inizia per aa se sono state applicate prima le produzioni (2) $S \to abA$ e poi ripetutamente in sequenza le produzioni (3) $A \to Saa$ e (2) $S \to abA$:

$$S \underset{(2)}{\Rightarrow} abA \underset{(3)}{\Rightarrow} abSaa \underset{(2)}{\Rightarrow} (ab)^2 Aaa \overset{*}{\Rightarrow} \ldots \underset{(2)}{\Rightarrow}$$
$$\underset{(2)}{\Rightarrow} (ab)^{i+1} A(aa)^i$$

Supponiamo che si utilizzi la produzione:

$$(3) \quad A \to Saa$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

per andare da $wA\alpha$ a $w\beta\alpha$.

Dunque: $\beta = Saa$ e ricadiamo nel caso analizzato in precedenza, in quanto x sarà generato da S.

Pertanto o x inizia per aa (se è utilizzata $S \to \lambda$)

oppure x inizia per ab, per quanto visto in precedenza.

Similmente, se (3) $A \to Saa$ è utilizzata per andare da $wA\alpha$ a $w\gamma\alpha$, allora $\gamma = Saa$ e si ha che y inizia per aa oppure y inizia per ab, per quanto visto in precedenza.

Supponiamo che si utilizzi la produzione:

$$(4)\ A \to b$$

per andare da $wA\alpha$ a $w\beta\alpha$.

Dunque: $\beta = b$ ed x inizia per b (x può essere o solo b oppure iniziare per ba, per le considerazioni fatte su α).

Parimenti, se (4) $A \to b$ è utilizzata per andare da $wA\alpha$ a $w\gamma\alpha$, allora $\gamma =$ b e y inizia per b.

Per riassumere, le tre possibilità sono:

1. x e y iniziano per aa
2. x e y iniziano per ab
3. x e y iniziano per b

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Caso a):

la sequenza di produzioni (3) $A \to Saa$ e (1) $S \to \lambda$ è utilizzata necessariamente in entrambe le derivazioni e $\beta = \gamma = Saa$.

Caso b):

le produzioni (3) $A \to Saa$ e (2) $S \to abA$ sono usate necessariamente una o più volte in sequenza in entrambe le derivazioni e $\beta = \gamma = Saa$.

Caso c):

(4) $A \to b$ deve essere usata in entrambe le derivazioni e $\beta = \gamma = b$.

Esercizio 10.3

Verificare che la seguente grammatica è LL(2):

(1) $S \to aSA$

(2) $S \to aAS$

(3) $A \to cA$

(4) $A \to ba$

Occorre verificare innanzitutto che G non sia LL(1).

Anziché utilizzare direttamente la definizione di grammatica LL(k), possiamo usare la seguente caratterizzazione:

Teorema 10.2

Una CFG è LL(k) se e solo se vale la seguente condizione:

se $A \to \beta$ e $A \to \gamma$ sono produzioni distinte in P allora

$First_k(\beta\,\alpha) \cap First_k(\gamma\,\alpha) = \varnothing$ per ogni $wA\alpha$ tale che $S \overset{*}{\underset{lm}{\Rightarrow}} wA\alpha$.

Considero dunque le produzioni distinte:

$$(1) \quad S \to aSA$$
$$(2) \quad S \to aAS$$

e le derivazioni:

$$1) \quad S \overset{*}{\underset{lm}{\Rightarrow}} wS\alpha \underset{\substack{(1)\\ lm}}{\Rightarrow} w\underbrace{aSA}_{\beta}\alpha$$

$$2) \quad S \overset{*}{\underset{lm}{\Rightarrow}} wS\alpha \underset{\substack{(2)\\ lm}}{\Rightarrow} w\underbrace{aAS}_{\gamma}\alpha$$

Si ha:

$$First_1(\beta\alpha) = \{a\} = First_1(\gamma\alpha)\,.$$

Dunque $First_1(\beta\alpha)$ e $First_1(\gamma\alpha)$ non sono disgiunti e G non è LL(1).

Mostriamo ora che G è LL(2):

Utilizziamo nuovamente la caratterizzazione vista in precedenza.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Considero nuovamente le due produzioni distinte di G:

$$(1) \quad S \to aSA$$
$$(2) \quad S \to aAS$$

e le derivazioni:

$$1) \quad S \overset{*}{\underset{lm}{\Rightarrow}} wS\alpha \underset{\substack{(1)\\ lm}}{\Rightarrow} w\underbrace{aSA}_{\beta}\alpha$$

$$2) \quad S \overset{*}{\underset{lm}{\Rightarrow}} wS\alpha \underset{\substack{(2)\\ lm}}{\Rightarrow} w\underbrace{aAS}_{\gamma}\alpha$$

Calcolo:

$$First_2(\beta\alpha) = First_2(aSA\alpha) = \{aa\}$$

in quanto le derivazioni da $\beta\alpha$ in G sono solo dei seguenti due tipi:

$$\beta\alpha = aSA\alpha \underset{(1)}{\Rightarrow} \underline{\underline{aa}}SAA\alpha$$

$$\beta\alpha = aSA\alpha \underset{(2)}{\Rightarrow} \underline{\underline{aa}}ASA\alpha$$

Calcolo:

$$First_2(\gamma\alpha) = First_2(aAS\alpha) = \{ac, ab\}$$

in quanto le derivazioni da $\gamma\alpha$ in G sono solo dei seguenti due tipi:

$$\gamma\alpha = aAS\alpha \underset{(3)}{\Rightarrow} \underbrace{ac}AS\alpha$$

$$\gamma\alpha = aAS\alpha \underset{(4)}{\Rightarrow} \underbrace{ab}aS\alpha$$

Dunque:

$$First_2(\beta\alpha) = \{aa\} \cap \{ac, ab\} = First_2(\gamma\alpha) = \varnothing$$

Considero ora le due produzioni distinte che riscrivono A:

$$(3) \quad A \to cA$$
$$(4) \quad A \to ba$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

e le derivazioni:

$$1) \quad S \overset{*}{\underset{lm}{\Rightarrow}} wA\alpha \underset{\substack{(3)\\ lm}}{\Rightarrow} w\underbrace{cA}_{\beta}\alpha$$

$$2) \quad S \overset{*}{\underset{lm}{\Rightarrow}} wA\alpha \underset{\substack{(4)\\ lm}}{\Rightarrow} w\underbrace{ba}_{\gamma}\alpha$$

Calcolo $First_2(\beta\alpha)$ e $First_2(\gamma\alpha)$:

$$First_2(\beta\alpha) = First_2(cA\alpha) = \{cc, cb\}$$

in quanto tutte le stringhe derivabili in G da $\beta\alpha$ lo sono attraverso i due seguenti tipi di derivazioni:

$$\beta\alpha = cA\alpha \underset{(3)}{\Rightarrow} \underbrace{cc}A\alpha$$

$$\beta\alpha = cA\alpha \underset{(4)}{\Rightarrow} \underbrace{cb}a\alpha$$

e

$$First_2(\gamma\alpha) = First_2(ba\alpha) = \{ba\}$$

Dunque:

$$First_2(\beta\alpha) = \{cc, cb\} \cap \{ba\} = First_2(\gamma\alpha) = \varnothing$$

e G è LL(2), in quanto non vi sono altre coppie di produzioni di G da considerare per verificare la condizione della caratterizzazione delle grammatiche LL(k).

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Proposizioni

Data una grammatica non contestuale G, valgono le seguenti proposizioni:

- è **decidibile** il problema di stabilire se G è LL(k), fissato $k \geq 1$ (ossia esiste un algoritmo che decide se G è LL(k));
- è **indecidibile** il problema di stabilire l'esistenza di un k finito per cui G è LL(k);
- è **indecidibile** il problema di stabilire se, per una data G che non è LL(1), esiste G' equivalente a G ($L(G) = L(G')$) e che sia LL(1).

L'ultimo risultato ha notevole interesse pratico, in quanto gli analizzatori sintattici per grammatiche LL(1) sono particolarmente efficienti e dunque sarebbe stato importante stabilire l'esistenza di una procedura meccanica (algoritmo) che, se esiste, genera una grammatica G' di tipo LL(1) equivalente ad una data grammatica G.

Questo risultato non esclude la possibilità di applicare metodi euristici per effettuare questa trasformazione, come illustrato dal seguente esercizio.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Esercizio 10.4

Sia data la seguente grammatica non contestuale che genera espressioni aritmetiche:

$$G_E = (X, V, S, P)$$

dove:

$X = \{i, (,), +, -, *, /\}$

$V = \{E, T, F\}$ $\quad E$ = espressione; T = termine; F = fattore

$S = E$

P :

$E \to T \mid E + T \mid E - T$

$T \to F \mid T * F \mid T / F$

$F \to i \mid (E)$

Determinare, se esiste, una grammatica equivalente a G_E e che sia LL(1).

G_E non è LL(1). (Dimostrarlo per esercizio).

Intuitivamente, G_E non è LL(1) perché esistono più parti destre nelle riscritture di uno stesso nonterminale (ossia nelle produzioni che hanno uno stesso nonterminale come parte sinistra) che iniziano con lo stesso carattere.

Per eliminare ciò, si opera una **fattorizzazione** (primo metodo euristico): essa consiste nell'utilizzare la proprietà distributiva della concatenazione di caratteri rispetto al simbolo " | " del metalinguaggio usato per descrivere le produzioni di una grammatica e nell'introdurre nuovi nonterminali.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Esempio: $\mathrm{E} \rightarrow \mathrm{E} + \mathrm{T} \mid \mathrm{E} - \mathrm{T}$ viene fattorizzato come segue: $\mathrm{E} \rightarrow \mathrm{E} \cdot (+T \mid -T)$ ed introducendo un nuovo nonterminale, si ha:

$$E \rightarrow EE'$$
$$E' \rightarrow +T \mid -T$$

Per fattorizzazione si ottiene una grammatica G'_E equivalente a G_E:

$$G'_E$$
$$E \rightarrow T \mid EE'$$
$$E' \rightarrow +T \mid -T$$
$$T \rightarrow F \mid TT'$$
$$T' \rightarrow *F \mid / F$$
$$F \rightarrow i \mid (E)$$

Trasformiamo ora G'_E in **forma normale priva di ricorsioni sinistre** (*NLR normal form*), eliminando le ricorsioni sinistre per *E* e per *T*, secondo l'algoritmo visto in precedenza:

$$A \rightarrow A\alpha \mid \beta$$

diventa:

$$A \rightarrow \beta B$$
$$B \rightarrow \alpha B \mid \lambda$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Otteniamo così G''_E equivalente a G'_E:

$$
\begin{aligned}
&E \to TE'' \\
&E'' \to E'E'' \mid \lambda \\
&E' \to +T \mid -T \\
&T \to FT'' \\
&T'' \to T'T'' \mid \lambda \\
&T' \to *F \mid / F \\
&F \to i \mid (E)
\end{aligned}
$$

G''_E è LL(1). (Dimostrarlo per esercizio).

Le regole empiriche sopra impiegate per trasformare la grammatica G_E, nella speranza di portarla in una forma LL(1), cioè la fattorizzazione e la eliminazione delle produzioni ricorsive a sinistra, sono gli strumenti fondamentali che si impiegano per operare la trasformazione desiderata.

10.4 Grammatiche LR(*k*)

Dopo aver visto le grammatiche LL(k), che consentono una analisi sintattica deterministica discendente (*top-down*), consideriamo una analoga classe di grammatiche, dette LR(k)[20], che consentono ancora una analisi deterministica, ma ascendente (*bottom-up*).

---

[20] la sigla LR(k) indica che, scandendo la stringa da analizzare da sinistra verso destra — *Left to right* —, ogni passo nella costruzione della derivazione destra — *Rightmost derivation* — è determinato esaminando i primi k caratteri della parte di stringa non ancora scandita.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Supporremo nel seguito che, come avviene di solito, un metodo di analisi sintattica bottom-up sia in grado di ottenere l'albero sintattico ricostruendo la derivazione destra, in forma di stringa di cifre corrispondente alla sequenza di produzioni applicate.

Se:

$$S = \alpha_0 \underset{rm}{\Rightarrow} \alpha_1 \underset{rm}{\Rightarrow} \ldots \underset{rm}{\Rightarrow} \alpha_i \underset{rm}{\Rightarrow} \alpha_{i+1} \underset{rm}{\Rightarrow} \ldots \underset{rm}{\Rightarrow} \alpha_n = w$$

è la derivazione destra della stringa w, l'analisi sintattica ascendente consiste nel determinare la produzione che permette di passare da α_{i+1} ad α_i per $i = n$-1, n-2, ..., 0.

Se $\alpha_i = \alpha A v$ e $\alpha_{i+1} = \alpha \beta v$, si dice che β viene ridotto ad A e che β è la *chiave* o *parte riducibile* di α_{i+1}.

Per determinare ad ogni passo della derivazione la chiave, si utilizza un metodo che scandisce i caratteri della stringa da analizzare da sinistra a destra e li sposta in cima ad una pila, finché sulla pila non compare la chiave, cui viene allora sostituita la parte sinistra.

Questo processo si ripete finché non è più possibile procedere perché la stringa non appartiene al linguaggio considerato (si è cioè rilevato un errore) o tutta la stringa è stata scandita e sulla pila è presente solo l'assioma S. In tal caso la stringa è accettata o riconosciuta.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Il processo suddetto prende il nome di **algoritmo** (di analisi sintattica) **a spostamento e riduzione**.

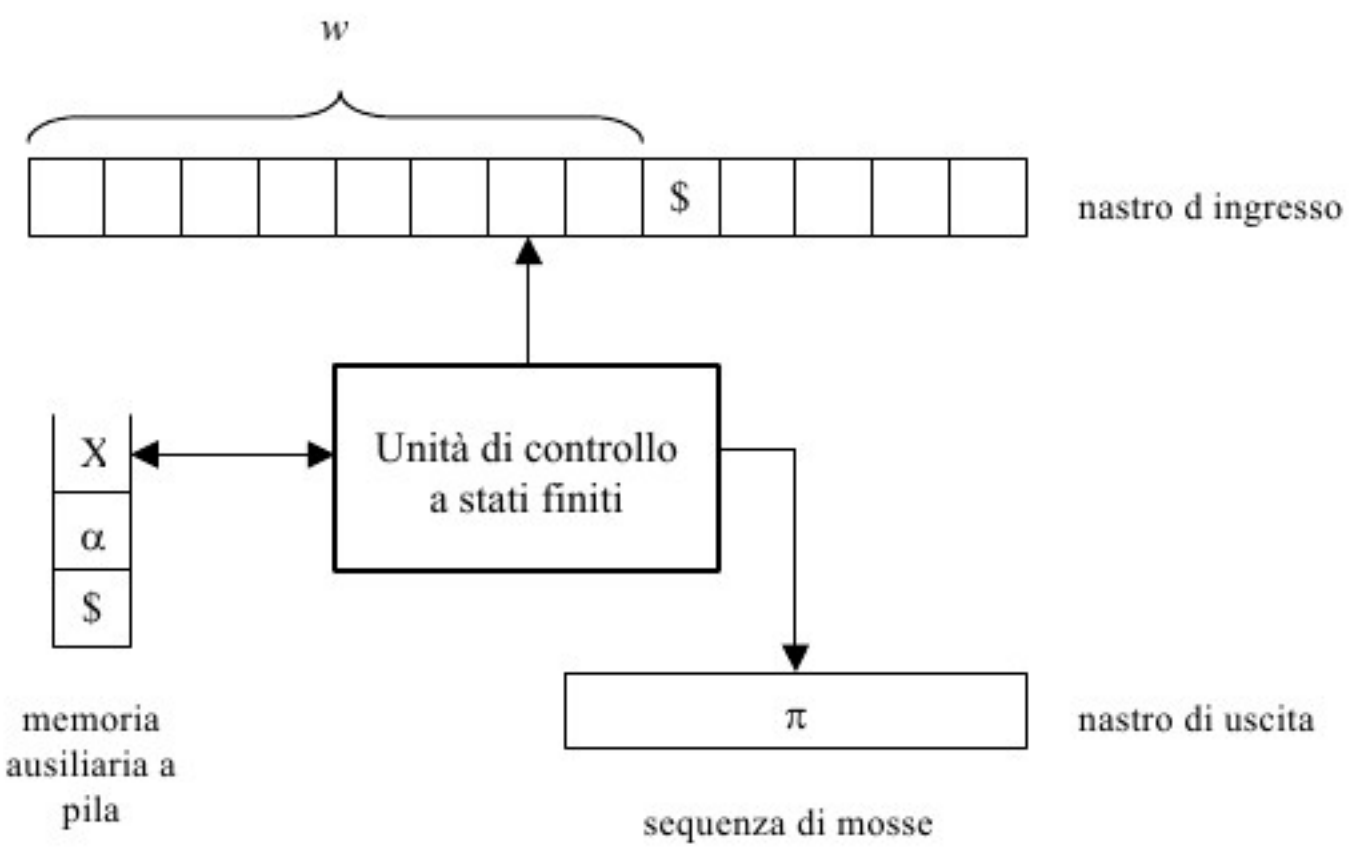
Formalmente, si dà la seguente:

Definizione 10.3 (Algoritmo a spostamento e riduzione)

Un algoritmo di analisi sintattica a spostamento e riduzione per una grammatica non contestuale $G = (X, V, S, P)$, le cui produzioni sono etichettate da 1 a p, è definito da una funzione parziale:

$$f : \left(\{\$\} \cdot (X \cup V)^*\right) \times \left(X^* \cdot \{\$\}\right) \rightarrow \{sposta, riduci\ 1, \ldots, riduci\ p\}$$

\$ funge da marcatore di fine pila e da marcatore di fine stringa sul nastro di ingresso.



Intuitivamente, si ha $f = riduci\ i$, se sulla pila è presente come chiave la parte destra della produzione i-esima di G.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Sia x la stringa da analizzare, delimitata a destra da \$; una configurazione dell'algoritmo è definita da una tripla:

$$(\$\alpha,\ w\$,\ \pi)$$

dove:

\$	è il simbolo inizialmente presente sulla pila;
$\$\alpha = \$X_1X_2\ldots X_m$	è il contenuto della pila (con in cima X_m), dove $X_i \in X \cup V,\ i = 1,2,\ldots,m,\ m > 0$;
$w = a_1a_2\ldots a_n$	è la parte di stringa ancora da analizzare (con a_1 sotto la testina di lettura), dove $a_i \in X$, $i = 1,2,\ldots,n,\ n \geq 0$;
$\pi = p_1p_2\ldots p_q$	è la stringa associata alla sequenza delle etichette delle produzioni impiegate nella derivazione $\alpha w \overset{*}{\underset{rm}{\Rightarrow}} x$, dove $p_i \in \{1,2,\ldots,p\}$ per $1 \leq i \leq q$, $q \geq 0$.

La configurazione:

(\$, x\$, λ)	è detta iniziale
(\$$S$, \$, π)	è detta finale

Il passaggio da una configurazione ad una successiva è definito dalle seguenti operazioni:

- se $f(\$\alpha, w\$)$ = *sposta* allora $(\$\alpha, w\$, \pi) \Rightarrow (\$\alpha a_1, w'\$, \pi)$ dove $w = a_1w'$.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

- se $f(\$\alpha, w\$) = riduci\ i$ allora $(\$\alpha, w\$, \pi) \Rightarrow (\$\alpha' A, w\$, \pi\, i)$ dove $\alpha = \alpha'\beta$ e $i : A \to \beta \in P$;
- se $f(\$\alpha, w\$)$ non è definita, l'algoritmo si ferma.

Una stringa w è accettata se $(\$, w\$, \lambda) \overset{*}{\Rightarrow} (\$S, \$, \pi)$.

In tal caso: $S \underset{\substack{\pi^R \\ rm}}{\overset{*}{\Rightarrow}} w$ cioè π è l'inversa della stringa associata alla derivazione destra di w.

Un algoritmo a spostamento e riduzione è corretto per una grammatica G se

$$L(G) = \{w \mid w \text{ è accettata dall'algoritmo}\}.$$

<u>Esempio 10.3</u>

Si consideri la grammatica:

$$\begin{array}{ll} (1) & E \to T \\ (2) & E \to E + T \\ (3) & T \to d \\ (4) & T \to T * d \end{array}$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

L'algoritmo a spostamento e riduzione è il seguente:

supponiamo:

$$\alpha \in \{+,*,d,E,T\}^*$$
$$y \in \{+,*,d\}^* \cdot \{\$\}$$
$$a \in \{+,*,\$\}$$

Configurazione	f
($, dy)	*sposta*
($$d$, ay)	*riduci* 3
($$\alpha T$, $+y$)	*riduci* 1
($$\alpha+d$, ay)	*riduci* 3
($$\alpha T*d$, ay)	*riduci* 4
($$\alpha*$, dy)	*sposta*
($$\alpha T$, $*y$)	*sposta*
($$\alpha+$, dy)	*sposta*
($$\alpha E$, $+y$)	*sposta*
($$T$, $)	*riduci* 1
($$E+T$, $+y$)	*riduci* 2
($$E+T$, $)	*riduci* 2

Vediamo l'analisi della stringa $d+d$:

$$(\$, d+d\$, \lambda) \Rightarrow (\$d, +d\$, \lambda) \Rightarrow (\$T, +d\$, 3) \Rightarrow (\$E, +d\$, 31) \Rightarrow$$
$$\Rightarrow (\$E, d\$, 31) \Rightarrow (\$E+d, \$, 31) \Rightarrow (\$E+T, \$, 313) \Rightarrow$$
$$\Rightarrow (\$E, \$, 3132)$$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Dunque, in corrispondenza della derivazione destra:

$$E \underset{rm}{\overset{(2)}{\Rightarrow}} E + T \underset{rm}{\overset{(3)}{\Rightarrow}} E + d \underset{rm}{\overset{(1)}{\Rightarrow}} T + d \underset{rm}{\overset{(3)}{\Rightarrow}} d + d$$

si ha la stringa associata:

$$2313 = (3132)^R$$

In generale, per la determinazione della mossa dell'algoritmo a spostamento e riduzione, può essere necessario consultare l'intero contenuto della pila e quanto rimane della stringa da analizzare.
Per rendere ragionevolmente efficiente l'algoritmo, si cerca di ridurre al minimo possibile le parti di pila e stringa da consultare.

<u>Definizione 10.4</u> (Grammatica LR(k))
Una CFG G è LR(k), $k \geq 1$, se l'esistenza di due derivazioni:

$$1) \quad S \underset{rm}{\overset{*}{\Rightarrow}} \alpha A w \underset{rm}{\Rightarrow} \alpha\beta w$$

$$2) \quad S \underset{rm}{\overset{*}{\Rightarrow}} \gamma B x \underset{rm}{\Rightarrow} \alpha\beta y$$

con $First_k(y) = First_k(w)$ implica $\alpha = \gamma$, $A = B$ e $x = y$.

<u>Definizione 10.5</u> (*Augmented grammar*)
Sia $G = (X, V, S, P)$ una CFG.
La grammatica $G' = (X, V \cup \{S'\}, S', P \cup \{S' \to S\})$ è, per definizione, la *augmented grammar* derivata da G.

La augmented grammar G' è semplicemente G con una nuova produzione di partenza $S' \to S$, ove S' è un nuovo assioma.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Aggiungiamo tale produzione di partenza in modo che quando è richiesta una riduzione che utilizza questa produzione, possiamo interpretare questa *"riduzione"* come un segnale di accettazione.

<u>Esercizio 10.5</u>

Dimostrare che la seguente grammatica è LR(1):

$$S \to aS \mid bA$$
$$A \to cA \mid c$$

Poiché S appare a destra nella produzione $S \to aS$, considero l'augmented grammar derivata da G:

$$G' = (X, V \cup \{S'\}, S', P')$$
$$P':$$
$$S' \to S$$
$$S \to aS \mid bA$$
$$A \to cA \mid c$$

Per definizione:

$$G \text{ è LR(1)} \overset{def.}{\Leftrightarrow} \left.\begin{array}{ll} \text{(i)} & S' \overset{*}{\underset{rm}{\Rightarrow}} \alpha A' w \underset{rm}{\Rightarrow} \alpha\beta w \\ \text{(ii)} & S' \overset{*}{\underset{rm}{\Rightarrow}} \gamma B x \underset{rm}{\Rightarrow} \alpha\beta y \\ \text{(iii)} & First_1(y) = First_1(w) \end{array}\right\} \Rightarrow \begin{array}{l} \alpha = \gamma \\ A' = B \\ x = y \end{array}$$

<u>1° caso)</u>

Considero $A' = S'$ allora $\alpha = w = \lambda$.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Inoltre da S' derivo solo S ($S' \to \underbrace{S}_{\beta}$) in quanto

$$\underbrace{S'}_{\alpha} \underbrace{\Rightarrow}_{w} \underset{rm}{} \underbrace{S}_{\alpha}\underbrace{}_{\beta}\underbrace{}_{w} .$$

Poiché $First_1(y) = First_1(w) = \{\lambda\}$ si ha: $y = \lambda$.

Considero: $\gamma Bx \underset{rm}{\Rightarrow} \underbrace{S}_{\alpha}\underbrace{}_{\beta}\underbrace{}_{y}$

S può essere prodotto da $S' \to S$ oppure $S \to aS$.

Ma nel secondo caso $\beta = S$ dovrebbe essere preceduto da *"a"*. Contraddizione.

Dunque l'unica produzione possibile è $S' \to S$ allora

$$B = S' = A' \text{ e } \gamma Bx \underset{rm}{\Rightarrow} \underbrace{S}_{\alpha}\underbrace{}_{\beta}\underbrace{}_{y}$$

da cui: $\gamma = \lambda = \alpha$ e $x = \lambda = y$.

<u>2° caso)</u> $A' = S$

Si ha:

$$S' \overset{*}{\underset{rm}{\Rightarrow}} \underbrace{a^k}_{\alpha}\underbrace{S}_{A'}\underbrace{}_{w} \qquad \alpha = a^k,\ k \geq 0,\ w = \lambda$$

Poiché $First_1(w) = \{\lambda\} = First_1(y)$ si ha $y = \lambda$ e ad S posso applicare $S \to aS$ oppure $S \to bA$.

i) Se considero $S \to \underbrace{aS}_{\beta}$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

$$S' \overset{*}{\underset{rm}{\Rightarrow}} \underbrace{a^k}_{\alpha} \underbrace{S}_{A'} \underbrace{\quad}_{w} \underset{rm}{\Rightarrow} \underbrace{a^k}_{\alpha} \underbrace{aS}_{\beta} \underbrace{\quad}_{w}$$

Considero:

$$S' \overset{*}{\Rightarrow} \gamma Bx \Rightarrow \underbrace{a^k}_{\alpha} \underbrace{aS}_{\beta} y$$

β si può ottenere solo da $S \to aS$ perché se usassi $S' \to S$ allora dovrei avere $\beta = S$, quindi $B = S = A'$ e $S' \overset{*}{\underset{rm}{\Rightarrow}} \gamma Sx \Rightarrow \underbrace{a^k}_{\alpha} \underbrace{aS}_{\beta} \underbrace{\quad}_{y}$ da cui: $\gamma = a^k = \alpha$ e $x = \lambda = y$.

ii) Se considero $S \to \underbrace{bA}_{\beta}$:

$$S' \overset{*}{\underset{rm}{\Rightarrow}} \underbrace{a^k}_{\alpha} \underbrace{S}_{A'} \underbrace{\quad}_{w} \underset{rm}{\Rightarrow} \underbrace{a^k}_{\alpha} \underbrace{bA}_{\beta} \underbrace{\quad}_{w}$$

$$S' \underset{rm}{\Rightarrow} \gamma Bx \underset{rm}{\Rightarrow} \underbrace{a^k}_{\alpha} \underbrace{bA}_{\beta} \underbrace{\quad}_{y}$$

A può essere prodotto da $S \to bA$ oppure da $A \to cA$, ma nel secondo caso A non sarebbe preceduto da "b" quindi $B = S = A'$ allora:

$$S' \overset{*}{\underset{rm}{\Rightarrow}} \gamma Sx \underset{rm}{\Rightarrow} \underbrace{a^k}_{\alpha} \underbrace{bA}_{\beta} \underbrace{\quad}_{y}$$

da cui:

$$\gamma = a^k = \alpha \quad \text{e} \quad x = \lambda = y.$$

<u>3° caso)</u> $A' = A$

$$S' \overset{*}{\underset{rm}{\Rightarrow}} a^k S \underset{rm}{\Rightarrow} a^k bA \overset{*}{\underset{rm}{\Rightarrow}} \underbrace{a^k bc^n}_{\alpha} \underbrace{A}_{A'} \underbrace{\quad}_{w} \qquad k,n \geq 0$$

allora $\alpha = a^k bc^n$, $k,n \geq 0$, $w = \lambda$.

Inoltre $First_1(w) = First_1(y) = \{\lambda\} \quad \Rightarrow \quad y = \lambda$.

Da A posso applicare $A \to cA$ oppure $A \to c$

i) Considero $A \to \underbrace{cA}_{\beta}$

$$S' \overset{*}{\underset{rm}{\Rightarrow}} \underbrace{a^k bc^n}_{\alpha} \underbrace{A}_{A'} \underbrace{\quad}_{w} \underset{rm}{\Rightarrow} \underbrace{a^k bc^n}_{\alpha} \underbrace{cA}_{\beta} \underbrace{\quad}_{w}$$

Considero:

$$S' \overset{*}{\underset{rm}{\Rightarrow}} \gamma Bx \underset{rm}{\Rightarrow} \underbrace{a^k bc^n}_{\alpha} \underbrace{cA}_{\beta} \underbrace{\quad}_{y}$$

A può essere prodotto da $A \to cA$ o da $S \to bA$, ma nel secondo caso $\beta \neq cA$ perché A sarebbe preceduto da *"b"*, quindi posso applicare solo $A \to cA$, cioè $B = A = A'$.

$$S' \overset{*}{\underset{rm}{\Rightarrow}} \gamma Ax \underset{rm}{\Rightarrow} \underbrace{a^k bc^n}_{\alpha} \underbrace{cA}_{\beta} \underbrace{\quad}_{y}$$

da cui $\gamma = a^k bc^n = \alpha$ e $x = \lambda = y$.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

ii) Considero $A \to \underbrace{c}_{\beta}$

$$S' \underset{rm}{\overset{*}{\Rightarrow}} \underbrace{a^k bc^n}_{\alpha} \underbrace{A}_{A'} \underbrace{\quad}_{w} \underset{rm}{\Rightarrow} \underbrace{a^k bc^n}_{\alpha} \underbrace{c}_{\beta} \underbrace{\quad}_{w}$$

Considero:

$$S' \underset{rm}{\overset{*}{\Rightarrow}} \gamma Bx \underset{rm}{\Rightarrow} \underbrace{a^k bc^n}_{\alpha} \underbrace{c}_{\beta} \underbrace{\quad}_{y}$$

"c" può essere prodotto da $A \to c$ oppure da $A \to cA$. Ma nel secondo caso "c" sarebbe seguito da A. Dunque posso applicare solo $A \to c$.

Allora $B = A = A'$ e $S' \underset{rm}{\overset{*}{\Rightarrow}} \gamma Ax \underset{rm}{\Rightarrow} \underbrace{a^k bc^n}_{\alpha} \underbrace{c}_{\beta} \underbrace{\quad}_{y}$ da cui:

$$\gamma = a^k bc^n = \alpha \text{ e } x = \lambda = y.$$

Per cui G è LR(1).

10.5 Esercizi proposti

<u>Esercizio 1</u>

Sia data la seguente grammatica:

$G = (X, V, S, P)$

ove $X = \{a, b, c, d\}$

$V = \{S, A, B\}$

$P = \{ \; S \to SaA \mid A,$

$A \to AbB \mid B,$

$B \to dSd \mid c \; \}$

Descrivere il linguaggio $L(G)$ generato da G.

Indicare se G è LL(k) per un certo valore di k.

Si trasformi G in una grammatica G' priva di ricorsioni sinistre.

Indicare se G' è LL(k) per un certo valore di k.

Esercizio 2

Sia data la seguente grammatica:

$G = (X, V, E, P)$

ove $X = \{Id, (,), +, *\}$

$V = \{E, E', T, T', F\}$

$P = \{ \quad E \rightarrow TE',$

$E' \rightarrow +TE' \mid \lambda,$

$T \rightarrow FT',$

$T' \rightarrow *FT' \mid \lambda,$

$F \rightarrow Id \mid (E) \qquad \}$

Descrivere il linguaggio $L(G)$.

Indicare se G è LL(k) per un certo valore di k.

Giustificare formalmente la precedente risposta.

Descrivere un algoritmo di parsing con stack esplicito per la grammatica G.

Esercizio 3

Sia data la seguente grammatica:

$G = (X, V, S, P)$

ove $X = \{a, b, c, d, e\}$

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

$$V = \{S, A, B\}$$

$$P = \{ \quad S \to aA \mid cB \ ,$$

$$A \to Sb \mid b,$$

$$B \to Bd \mid e \qquad \}$$

Costruire l'albero di derivazione per la parola $w = aacedddbb$.

Indicare se G è LL(k) per un certo valore di k.

348 - ANALISI SINTATTICA. GRAMMATICHE LL(K) E LR(K)

Bibliografia

Capitolo 1

Mauri, G., *Informatica teorica*, in Enciclopedia delle Scienze, Elettronica - Informatica - Comunicazioni, De Agostini, Novara, 1984.

Rayward-Smith, V.J., *A First Course in Formal Language Theory*, Blackwell Scientific Publications, Oxford, United Kingdom, 1983.

Moll, R.N., Arbib, M.A., Kfoury, A.J., *An Introduction to Formal Language Theory*, Springer-Verlag, New York, 1988.

Capitolo 2

Crespi-Reghizzi, S., Della Vigna, P.L., Ghezzi, C., *Linguaggi Formali e Compilatori*, ISEDI, Petrini Editore, Milano, 1985.

Moll, R.N., Arbib, M.A., Kfoury, A.J., *An Introduction to Formal Language Theory*, Springer-Verlag, New York, 1988.

Di Martino, L., Tamburini, M.C., *Appunti di ALGEBRA*, CLUED, Milano, 1983.

Borzacchini, L., *Dispense del corso di Logica Matematica*, Bari, 1985.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Capitolo 3

Moll, R.N., Arbib, M.A., Kfoury, A.J., *An Introduction to Formal Language Theory*, Springer-Verlag, New York, 1988.

Crespi-Reghizzi, S., Della Vigna, P.L., Ghezzi, C., *Linguaggi Formali e Compilatori*, ISEDI, Petrini Editore, Milano, 1985.

Capitolo 4

Moll, R.N., Arbib, M.A., Kfoury, A.J., *An Introduction to Formal Language Theory*, Springer-Verlag, New York, 1988.

Hopcroft, J.E., Ullman, J.D., *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, Reading, Massachussetts, 1979.

Rayward-Smith, V.J., *A First Course in Formal Language Theory*, Blackwell Scientific Publications, Oxford, United Kingdom, 1983.

Crespi-Reghizzi, S., Della Vigna, P.L., Ghezzi, C., *Linguaggi Formali e Compilatori*, ISEDI, Petrini Editore, Milano, 1985.

Aho, A.V., Ullman, J.D., *The Theory of Parsing, Translation, and Compiling*, Volume I: Parsing, Prentice-Hall, Englewood Cliffs, N.J., 1972.

Titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Capitolo 5

Moll, R.N., Arbib, M.A., Kfoury, A.J., *An Introduction to Formal Language Theory*, Springer-Verlag, New York, 1988.

Crespi-Reghizzi, S., Della Vigna, P.L., Ghezzi, C., *Linguaggi Formali e Compilatori*, ISEDI, Petrini Editore, Milano, 1985.

Bartezzaghi, S., *Accavallavacca - inventario di parole da gioco*, Bompiani, Milano, 1992.

Capitolo 6

Moll, R.N., Arbib, M.A., Kfoury, A.J., *An Introduction to Formal Language Theory*, Springer-Verlag, New York, 1988.

Rayward-Smith, V.J., *A First Course in Formal Language Theory*, Blackwell Scientific Publications, Oxford, United Kingdom, 1983.

Crespi-Reghizzi, S., *Sintassi, semantica e tecniche di compilazione*, Volume primo: metodi sintattici, CLUP, Milano, 1985.

Capitolo 7

Moll, R.N., Arbib, M.A., Kfoury, A.J., *An Introduction to Formal Language Theory*, Springer-Verlag, New York, 1988.

titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Hopcroft, J.E., Ullman, J.D., *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, Reading, Massachussetts, 1979.

Manna, Z., *Teoria Matematica della Computazione*, Boringhieri, Torino, 1978.

Crespi-Reghizzi, S., Della Vigna, P.L., Ghezzi, C., *Linguaggi Formali e Compilatori*, ISEDI, Petrini Editore, Milano, 1985.

Capitolo 8

Moll, R.N., Arbib, M.A., Kfoury, A.J., *An Introduction to Formal Language Theory*, Springer-Verlag, New York, 1988.

Hopcroft, J.E., Ullman, J.D., *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, Reading, Massachussetts, 1979.

Crespi-Reghizzi, S., Della Vigna, P.L., Ghezzi, C., *Linguaggi Formali e Compilatori*, ISEDI, Petrini Editore, Milano, 1985.

Brookshear, J.G., *Theory of Computation - Formal Languages, Automata, and Complexity*, The Benjamin/Cummings Publishing Company, Redwood City, California, 1989.

Capitolo 9

Moll, R.N., Arbib, M.A., Kfoury, A.J., *An Introduction to Formal Language Theory*, Springer-Verlag, New York, 1988.

titolo: Elementi di TEORIA dei LINGUAGGI FORMALI

Hopcroft, J.E., Ullman, J.D., *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, Reading, Massachussetts, 1979.

Capitolo 10

Crespi-Reghizzi, S., Della Vigna, P.L., Ghezzi, C., *Linguaggi Formali e Compilatori*, ISEDI, Petrini Editore, Milano, 1985.