

Linguaggi e Computabilità

UniShare

Davide Cozzi
@dlcgold

Gabriele De Rosa
@derogab

Federica Di Lauro
@f_dila

Indice

1	Introduzione	2
1.1	Definizioni	2
1.1.1	Alberi Sintatici	14
1.1.2	Grammatiche ambigue	18
1.1.3	Grammatiche Regolari	20
1.1.4	Espressioni Regolari (Regex)	23
1.2	Automati	29
1.2.1	Automati deterministici	30
1.2.2	Automati non deterministici	36

Capitolo 1

Introduzione

Questi appunti sono presi a lezione. Per quanto sia stata fatta una revisione è altamente probabile (praticamente certo) che possano contenere errori, sia di stampa che di vero e proprio contenuto. Per eventuali proposte di correzione effettuare una pull request. Link: <https://github.com/dlclgold/Appunti>.

Grazie mille e buono studio!

1.1 Definizioni

- un **linguaggio** è un insieme di stringhe che può essere generato mediante un dato meccanismo con delle date caratteristiche; un linguaggio può essere riconosciuto, ovvero dando in input una stringa un meccanismo può dirmi se appartiene o meno ad un linguaggio. I meccanismi che generano linguaggi si chiamano *grammatiche*, quelli che li riconoscono *automi*. I linguaggi formali fanno parte dell'informatica teorica (*TCS*)
- si definisce **alfabeto** come un insieme finito e non vuoto di simbolo (come per esempio il nostro alfabeto o le cifre da 0 a 9). Solitamente si indica con Σ o Γ
- si definisce **stringa** come una sequenza finita di simboli (come per esempio una parola o una sequenza numerica). La stringa vuota è una sequenza di 0 simboli, e si indica con ε o λ
- si definisce **lunghezza di una stringa** il numero di simboli che la compone (ovviamente contando ogni molteplicità). Se si ha $w \in \Sigma^*$ è una stringa w con elementi da Σ^* (insieme di tutte le stringhe di tutte le lunghezze possibili fatte da Σ), allora $|w|$ è la lunghezza di w , inoltre $|\varepsilon| = 0$.

- si definisce **potenza di un alfabeto** Σ^k come l'insieme di tutte le sequenze (espressi come stringhe e non simboli) di lunghezza $k \in \mathbb{N}$, $k > 0$ ottenibili da quell'alfabeto (se Σ^2 si avranno tutte le sequenze di 2 elementi etc...). Se ho $k = 1$ si ha $\Sigma^1 \neq \Sigma$ in quanto ora ho stringhe e non simboli. Se ho $k = 0$ ho $\Sigma^0 = \varepsilon$. Dato k ho $|\Sigma|$ che è la cardinalità dell'insieme Σ (e non la sua lunghezza come nel caso delle stringhe); sia $w \in \Sigma^k = a_1, a_2, \dots, a_k$, $a_i \in \Sigma$ e $|\Sigma| = q$ ora:

$$|\Sigma^k| = q^k$$

- si definisce Σ^* come **chiusura di Kleene** che è l'unione infinita di Σ^k ovvero

$$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \dots \cup \Sigma^k$$

- si ha che Σ^+ è l'unione per $k \geq 1$ di Σ^k ovvero:

$$\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \dots \cup \Sigma^k = \Sigma^* - \Sigma^0$$

per esempio, per l'insieme $\{0, 1\}$ si ha:

$$\Sigma^* = \{\varepsilon, 0, 1, 00, 01, 10, 100, 000, \dots\}$$

- quindi un **linguaggio** L è un insieme di stringhe e:

$$L \subseteq \Sigma^*$$

si hanno sottoinsiemi particolari, come l'insieme vuoto, che resta però un linguaggio, il **linguaggio vuoto** e $\emptyset \in \Sigma^k$, $|\emptyset| = 0$ che è diverso dal linguaggio che contiene la stringa vuota $|\varepsilon| = 1$ (che conta come una stringa). Inoltre $\Sigma^* \subseteq \Sigma^*$ che ha lunghezza infinita. Posso concatenare due stringhe con un punto: $a \cdot b \cdot c = abc$ e $a \cdot \varepsilon = a$. Ovviamente la stringa concatenata è lunga come la somma delle lunghezze delle stringhe che la compongono. Vediamo qualche esempio di linguaggio:

- il linguaggio di tutte le stringhe che consistono in n 0 seguiti da n 1:

$$\{\varepsilon, 01, 0011, 000111, \dots\}$$

- l'insieme delle stringhe con un uguale numero di 0 e di 1:

$$\{\varepsilon, 01, 10, 0011, 0101, 1001, \dots\}$$

- l'insieme dei numeri binari il cui valore è un numero primo:

$$\{\varepsilon, 10, 11, 101, 111, 1011, \dots\}$$

- Σ^* è un linguaggio per ogni alfabeto Σ
- \emptyset , il linguaggio vuoto, e $\{\varepsilon\}$ sono un linguaggio rispetto a qualunque alfabeto

Prendiamo un alfabeto $\Sigma = \{0, 1\}$ con la sua chiusura di Kleen $\Sigma = \{0, 1\}^*$. Quando si ha un input si può avere un problema di decisione, P , che dia come output "si" o "no". Posso avere un problema di decisione (o *membership*) su $w \in \Sigma = \{0, 1\}^*$, con w stringa, che dia in output "si" o "no". Un linguaggio L sarà:

$$L = \{w \in \{0, 1\}^* \mid P(w) = \text{si}\}$$

quindi si ha che:

$$\Sigma^* \setminus L = \{P(w) = \text{no}\}$$

Vediamo ora un esempio di *Context Free Language (CFL)*, costruito a partire da una *Context Free Grammar (CFG)*:

Esempio 1. Sia $\Sigma = \{0, 1\}$ e $L_{pal} = \text{"stringhe palindrome binarie"}$. Quindi, per esempio, $0110 \in L$, $11011 \in L$ ma $10010 \notin L$. Si ha che ε , la stringa vuota, appartiene a L . Diamo una definizione ricorsiva:

- **base:** $\varepsilon, 0, 1 \in L_{pal}$
- **passo:** se w è palindroma allora $0w0$ è palindromo e $1w1$ è palindromo

una variabile generica S può sottostare alle regole di produzione di una certa grammatica. In questo caso si ha uno dei seguenti:

$$S \rightarrow \varepsilon, S \rightarrow 0, S \rightarrow 1, S \rightarrow 0S0, S \rightarrow 1S1$$

Si ha che una grammatica G è una quadrupla $G = (V, T, P, S)$ con:

- V simboli variabili
- T simboli terminali, ovvero i simboli con cui si scrivono le stringhe alla fine
- P regole di produzione
- S variabile di partenza *start*

riprendiamo l'esempio sopra:

Esempio 2.

$$G_{pal} = (V = \{S\}, T = \{0, 1\}, P, S)$$

con:

$$P = \{S \rightarrow \varepsilon, S \rightarrow 0, S \rightarrow 1, S \rightarrow 0S0, S \rightarrow 1S1\}$$

Si può ora costruire un algoritmo per creare una stringa palindroma a partire dalla grammatica G :

$$\underbrace{S}_{\text{start applico una regola}} \rightarrow 1S1 \rightarrow 01S10 \rightarrow \underbrace{01010}_{\text{sostituisco variabile}}$$

con S , $1S1$ e $01S10$ che sono forme sentenziali. Posso così ottenere tutte le possibili stringhe. Esiste anche una forma abbreviata:

$$S \rightarrow \varepsilon | 0 | 1 | 0S0 | 1S1$$

Non si fanno sostituzioni in parallelo, prima una S e poi un'altra

Si hanno 4 grammatiche formali, *gerarchia di Chomsky*:

- **tipo 0:** non si hanno restrizioni sulle regole di produzione, $\alpha \rightarrow \beta$. Sono linguaggi ricorsivamente numerabili e sono rappresentati dalle *macchine di Turing*, deterministiche o non deterministiche (la macchina di Turing è un automa)
- **tipo 1:** il lato destro della produzione ha lunghezza almeno uguale a quello sinistro. Sono grammatiche dipendenti dal contesto (*contestuali*) e come automa hanno *la macchina di Turing che lavora in spazio lineare*:

$$\alpha_1 A \alpha_2 \rightarrow \alpha_1 B \alpha_2$$

con α_1 e α_2 detti *contesto* e $\alpha_1, \alpha_2, \beta \in (V \cup T)^*$

- **tipo 2:** sono quelle libere dal contesto, context free. Come regola ha $A \rightarrow \beta$ con $A \in V$ e $\beta \in (V \cup T)^*$ e come automa ha gli *automi a pila non deterministici*
- **tipo 3:** sono le grammatiche *regolari*. Come regole ha $A \rightarrow \alpha B$ (o $A \rightarrow B\alpha$) e $A \rightarrow \alpha$ con $A, B \in V$ e $\alpha \in T$. Come automi ha gli *automi a stato finito deterministici o non deterministici*

Esempio 3. Sia $G = (V, T, O, E)$, con $V = \{E, I\}$ e $T = \{a, b, 0, 1, (,), +, *\}$ quindi ho le seguenti regole, è di tipo 3:

1. $E \rightarrow I$
2. $E \rightarrow E + E$
3. $E \rightarrow E * E$
4. $E \rightarrow (E)$
5. $I \rightarrow a$
6. $I \rightarrow b$
7. $I \rightarrow Ia$
8. $I \rightarrow Ib$
9. $I \rightarrow I0$
10. $I \rightarrow I1$

voglio ottenere $a*(a+b00)$ sostituisco sempre a destra (*right most derivation*)

$$E \rightarrow E * E \rightarrow E * (E) \rightarrow E * (E + E) \rightarrow E * (E + I) \rightarrow E + (E + I0) \\ \rightarrow R + (I + b00) \rightarrow E * (a + b00) \rightarrow I * (a + b00) \rightarrow a * (a + b00)$$

usiamo ora l'inferenza ricorsiva:

passo	stringa ricorsiva	var	prod	passo stringa impiegata
1	a	I	5	\
2	b	I	6	\
3	b0	I	9	2
4	b00	I	9	3
5	a	E	1	1
6	b00	E	1	4
7	a+b00	E	2	5,6
8	(a+b00)	E	4	7
9	a*(a+b00)	E	3	5, 8

definisco formalmente la derivazione \rightarrow :

Definizione 1. Prendo una grammatica $G = (V, T, P, S)$, grammatica CFG. Se $\alpha A \beta$ è una stringa tale che $\alpha, \beta \in (V \cup T)^*$, appartiene sia a variabili che terminali. Sia $A \in V$ e sia $a \rightarrow \gamma$ una produzione di G . Allora scriviamo:

$$\alpha A \beta \rightarrow \alpha \gamma \beta$$

con $\gamma \in (V \cup T)^*$.

Le sostituzioni si fanno indipendentemente da α e β . Questa è quindi la definizione di derivazione.

Definizione 2. Definisco il simbolo \rightarrow_* , ovvero il simbolo di derivazioni in 0 o più passi. Può essere definito in modo ricorsivo. Per induzione sul numero di passi.

- la base dice che $\forall \alpha \in (V \cup T)^*, \alpha \rightarrow_* \alpha$
- il passo è: se $\alpha \rightarrow_G * \beta$ e $\beta \rightarrow_G * \gamma$ allora $\alpha \rightarrow_* \gamma$

Si può anche dire che $\alpha \rightarrow_G * \beta$ sse esiste una sequenza di stringhe $\gamma_1, \dots, \gamma_n$ con $n \geq 1$ tale che $\alpha = \gamma_1$, $\beta = \gamma_n$ e $\forall i, 1 < i < n - 1$ si ha che $\gamma_i \rightarrow \gamma_{i+1}$ la derivazione in 0 o più passi è la chiusura transitiva della derivazione

Definizione 3. avendo ora definito questi simboli possiamo definire una forma sentenziale. Infatti è una stringa α tale che:

$$\forall \alpha \in (V \cup T)^* \text{ tale che } S \rightarrow_G * \alpha$$

Definizione 4. data $G = (V, T, P, S)$ si ha che $L(G) = \{w \in T^* \mid S \rightarrow_G * w\}$ ovvero composto da stringhe terminali che sono derivabili o 0 o più passi.

Esempio 4. formare una grammatica CFG per il linguaggio:

$$L = \{0^n 1^n \mid n \geq 1\} = \{01, 0011, 000111, \dots\}$$

con x^n intendo una concatenazione di n volte x (che nel nostro caso sono 0 e 1).

posso scrivere:

$$0^n 1^n = 00^{n-1} 1^{n-1} 1$$

il nostro caso base sarà la stringa 01, Poi si ha: $G = (V, T, P, S)$, $T = \{0, 1\}$, $V = \{S\}$, il caso base $S \rightarrow 01$ e $S \rightarrow 0S1$ il caso passo è quindi: se $w = 0^{n-1} 1^{n-1} \in L$ allora $0w1 \in L$.

Ora voglio dimostrare che $000111 \in L$, ovvero $S \rightarrow_* 000111$:

$$S \rightarrow 0S1 \rightarrow 00S11 \rightarrow 000S111$$

Teorema 1. data la grammatica $G = \{V, T, P, S\}$ CFG e $\alpha \in (V \cup T)^*$. Si ha che vale $S \rightarrow * \alpha$ sse $S \rightarrow_{lm} * \alpha$ sse $S \rightarrow_{rm} * \alpha$. Con $\rightarrow_{lm} *$ simbolo di left most derivation e $\rightarrow_{rm} *$ simbolo di right most derivation

Esempio 5. formare una grammatica CFG per il linguaggio:

$$L = \{0^n 1^n | n \geq 0\} = \{\varepsilon, 01, 0011, 000111, \dots\}$$

stavolta abbiamo anche la stringa vuota. Il caso base stavolta è $S \rightarrow \varepsilon | 0S1$

Esempio 6. Fornisco una CFG per $L = \{a^n | n \geq 1\} = \{a, aa, aaa, \dots\}$. La base è a

il passo è che se $a^{n-1} \in L$ allora $a^{n-1}a \in L$ (o che $aa^{n-1} \in L$).

Si ha la grammatica $G = \{V, T, P, S\}$, $V = \{S\}$, $T = \{a\}$ e si hanno $S \rightarrow a | Sa$ (o $S \rightarrow a | aS$). Dimostro che $a^3 \in L$.

$$S \rightarrow Sa \rightarrow Saa \rightarrow aaa$$

oppure

$$S \rightarrow aS \rightarrow aaS \rightarrow aaa$$

Esempio 7. trovo una CFG per $L = \{(ab)^n | n \geq 1\} = \{ab, abab, ababab, \dots\}$

La base è ab

il passo è che se $(ab)^{n-1} \in L$ allora $(ab)^{n-1}ab \in L$.

Si ha la grammatica $G = \{V, T, P, S\}$, $V = \{S\}$, $T = \{a, b\}$ (anche se in realtà $T = \{ab\}$) e si hanno $S \rightarrow ab | Aab$. Poi dimostro come l'esempio sopra

Esempio 8. trovo una CFG per $L = \{a^n cb^n | n \geq 1\} = \{acb, aacbb, aaacbbb, \dots\}$

Il caso base è acb il passo è che se $a^{n-1}cb^{n-1} \in L$ allora $a^{n-1}cb^{n-1}acb \in L$

Si ha la grammatica $G = \{V, T, P, S\}$, $V = \{S\}$, $T = \{a, b, c\}$ e si hanno $S \rightarrow aSb | acb$.

dimostro che $aaaacbbbbb \in L$:

$$S \rightarrow aSb \rightarrow aaSbb \rightarrow aaaaSbbb \rightarrow aaaacbbbbb$$

provo a usare anche una grammatica regolare, con le regole $S \rightarrow aS | c$, $c \rightarrow cB$ e $B \rightarrow bB | b$;

$$S \rightarrow aS \rightarrow aaS \rightarrow aaC \rightarrow aacB \rightarrow aacb \dots$$

non si può dimostrare in quanto non si può imporre una regola adatta

Esempio 9. $L = \{a^n cb^{n-1} | n \geq 2\}$, con $a^n cb^{n-1} = a^{n-1}acb^{n-1}$. $S \rightarrow aSb | aacb$. Quindi:

$$S \rightarrow aSb \rightarrow aaacbb \in L$$

Esempio 10. cerco CFG per $L = \{a^n c^k b^n \mid n, k > 0\}$. a e b devono essere uguali, uso quindi una grammatica context free, mentre c genera un linguaggio regolare.

Si ha la grammatica $G = \{V, T, P, S\}$, $V = \{S, C\}$, $T = \{a, b, c\}$ e si hanno $S \rightarrow aSb \mid aCb$ e $C \rightarrow cC \mid c$. dimostro che $aaaccbbb \in L$, $n = 3$, $k = 2$:

$$S \rightarrow aSb \rightarrow aaSbb \rightarrow aaaCbbb \rightarrow aaacCbbb \rightarrow aaaccbbb$$

Esempio 11. scrivere CFG per $L = \{a^n b^n c^k b^k \mid n, k \geq 0\}$

$$= \{w \in \{a, b, c, d\}^* \mid a^n b^n c^k b^k \mid n, k \geq 0\}$$

quindi L concatena due linguaggi $L1$ e $L2$, $X = \{a^n b^n\}$ e $Y = \{c^k d^k\}$:

$$X \rightarrow aXb \mid \varepsilon$$

$$Y \rightarrow cYd \mid \varepsilon$$

$$S \rightarrow XY$$

voglio derivare $abcd$:

$$S \rightarrow XY \rightarrow XcYd \rightarrow aXbcYd \rightarrow aXbc\varepsilon d \rightarrow a\varepsilon bc\varepsilon d \rightarrow abcd$$

voglio derivare cd

$$S \rightarrow XY \rightarrow Y \rightarrow cYd \rightarrow cd$$

Quindi se ho $w \in L1, L2$, ovvero appartenente ad una concatenazione di linguaggi prima uso le regole di un linguaggio, poi dell'altro e infine ottengo il risultato finale.

Esempio 12. scrivere CFG per $L = \{a^n b^k c^k d^n \mid n > 0, k \geq 0\}$.

$$S \rightarrow aSd \mid aXd$$

$$X \rightarrow bXc \mid \varepsilon$$

derivo $aabcbdd$:

$$S \rightarrow aSd \rightarrow aaXdd \rightarrow aabXcdd \rightarrow aabcbdd$$

Esempio 13. scrivere CFG per $L = \{a^n c b^n c^m a d^m \mid n > 0, m \geq 1\}$.

$$S \rightarrow XY$$

$$X \rightarrow aXb \mid c$$

$$Y \rightarrow cUd \mid cad$$

$$S \rightarrow XY \rightarrow cY \rightarrow ccad$$

Esempio 14. scrivere CFG per $L = \{a^{n+m}xc^nyd^m \mid n, m \geq 0\}$. $a^{n+m} = a^n a^m$ o $a^m a^n$. Si hanno 2 casi:

$$1. L = \{a^n a^m x c^n y d^m \mid n, m \geq 0\}$$

$$2. L = \{a^m a^n x c^n y d^m \mid n, m \geq 0\}$$

ma solo $L = \{a^m a^n x c^n y d^m \mid n, m \geq 0\}$ può generare una CFG (dove non si possono fare incroci, solo concatenazioni e inclusioni/innesti).

$$S \rightarrow aSd \mid Y$$

$$Y \rightarrow Xy$$

$$X \rightarrow aXc \mid x$$

si può fare in 2:

$$S \rightarrow aSd \mid Xy$$

$$X \rightarrow aXc \mid x$$

derivo con $m = n = 1$, $aa x c y d$:

$$S \rightarrow aSd \rightarrow aXyd \rightarrow aaXcyd \rightarrow aa x c y d$$

Esempio 15. scrivere CFG per $L = \{a^n b^m \mid n \geq m \geq 0\}$.

$$L = \{\varepsilon, a, ab, aa, aab, aabb, aaa, aaab, aaabb, aaabbb, \dots\}$$

Se $n \geq m$ allora $\exists k \geq 0 \rightarrow n = m + k$. Quindi:

$$l = \{a^{m+k} b^m \mid m, k \geq 0\}$$

si può scrivere in 2 modi:

$$1. l = \{a^m a^k b^m \mid m, k \geq 0\} \text{ quindi con innesto}$$

$$2. l = \{a^k a^m b^m \mid m, k \geq 0\} \text{ quindi con concatenazione}$$

entrambi possibili per una CFG:

1.

$$S \rightarrow XY$$

$$X \rightarrow aX \mid \varepsilon \text{ si può anche scrivere } X \rightarrow Xa \mid \varepsilon$$

$$Y \rightarrow aYb \mid \varepsilon$$

oppure

$$S \rightarrow aS \mid X$$

$$X \rightarrow aXb \mid \varepsilon$$

2.

$$S \rightarrow aSb|\varepsilon$$

$$X \rightarrow aX|\varepsilon$$

Esempio 16. scrivere CFG per $L = \{a^n b^{m+n} c^h \mid m > h \geq 0, n \geq 0\}$.
Se $n > h$ allora $\exists k \rightarrow n = h + k$, quindi:

$$L = \{a^n b^{m+h+k} c^h \mid m > h \geq 0, n \geq 0\}$$

. ovvero:

$$L = \{a^n b^n b^k b^h c^h \mid m \geq 0, k > 0, h \geq 0\}$$

si ha:

$$S \rightarrow XYZ$$

$$X \rightarrow aXb|\varepsilon$$

$$Y \rightarrow Yb|b$$

$$Z \rightarrow bZc|\varepsilon$$

si può anche fare:

$$S \rightarrow XY$$

$$X \rightarrow aXb|\varepsilon$$

$$Y \rightarrow bYc|Z$$

$$Z \rightarrow bZ|b$$

Esempio 17. scrivere CFG per $L = \{a^n b^m c^k \mid k > n + m, n, m \geq 0\}$.
per $n = m = 0, k = 1$ avrò la stringa c . se $k > n + m$ allora $\exists l > 0 \rightarrow k = n + m + l$ quindi:

$$L = \{a^n b^m c^{n+m+l} \mid l > 0, n, m \geq 0\}$$

$$= L = \{a^n b^m c^n c^m c^l \mid l > 0, n, m \geq 0\}$$

sistemando:

$$= L = \{a^n b^m c^l c^m c^n c^l \mid l > 0, n, m \geq 0\}$$

quindi:

$$S \rightarrow aSc|X$$

$$X \rightarrow bXc|Y$$

$$Y \rightarrow cY|c$$

Esempio 18. scrivere CFG per $L = \{a^n x c^{n+m} y^h z^k d^{m+h} \mid n, m, k, h \geq 0\}$.
ovvero:

$$L = \{a^n x c^n c^m y^h z^k d^h d^m \mid n, m, k, h \geq 0\}$$

quindi avrò:

$$S \rightarrow XY$$

$$X \rightarrow aXc \mid x$$

$$Y \rightarrow cYd \mid W$$

$$W \rightarrow yWd \mid X$$

$$Z \rightarrow zZ \mid \varepsilon$$

Esempio 19. vediamo un esempio di grammatica dipendente dal contesto:

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

$G = \{V, T, P, S\} = \{(S, B, C, X)\} = \{(a, b, c), P, S\}$ ecco le regole di produzione (qui posso scambiare variabili a differenza delle context free):

$$1. S \rightarrow aSBC$$

$$2. S \rightarrow aBC$$

$$3. CB \rightarrow XB$$

$$4. XB \rightarrow XC$$

$$5. XC \rightarrow BC$$

$$6. aB \rightarrow ab$$

$$7. bB \rightarrow bb$$

$$8. bC \rightarrow bc$$

$$9. cC \rightarrow cc$$

vediamo un esempio di derivazione: per $n = 1$ ho abc ovvero:

$$S \rightarrow aBC \rightarrow abC \rightarrow abc$$

con $n = 2$ ho $aabbcc$: $S \rightarrow aSBC \rightarrow aaBCBC \rightarrow aaBXBC \rightarrow aaBXCC \rightarrow aaBBCC \rightarrow aabBCC \rightarrow aabbCC \rightarrow aabbccC \rightarrow aabbcc$

Esempio 20. vediamo un esempio di grammatica dipendente dal contesto:

$$L = \{a^n b^m c^n d^m \mid n, m \geq 1\}$$

Si ha:

$$G = (\{S, X, C, D, Z\}, \{a, b, c, d\}, P, S)$$

con le seguenti regole di produzione:

- $S \rightarrow aSc \mid aXc$
- $X \rightarrow bXD \mid bD$
- $DC \rightarrow CD$
- $DC \rightarrow DZ$
- $DZ \rightarrow CZ$
- $XZ \rightarrow CD$
- $bC \rightarrow bc$
- $cC \rightarrow cc$
- $cD \rightarrow cd$
- $dD \rightarrow dd$

provo a derivare $aabbccddd$ quindi con $n = 2, m = 3$:

$$\begin{aligned} S &\rightarrow aSC \rightarrow aaXCC \rightarrow aabXDCC \rightarrow aabbXDDCC \rightarrow \\ &aabbbDDDCC \rightarrow aabbbCCDDD \rightarrow aabbccddd \end{aligned}$$

Esempio 21. Sia $L = \{w \in \{a, b\}^* \mid w \text{ contiene lo stesso numero di } a \text{ e } b\}$:

$$S \rightarrow aSbS \mid bSaS \mid \varepsilon$$

dimostro per induzione che è corretto:

- **caso base:** $|w| = 0 \rightarrow w = \varepsilon$

- **caso passo:** si supponga che G produca tutte le stringhe (di lunghezza $< n$) di $\{a, b\}^*$ con lo stesso numero di a e b e dimostro che produce anche quelle di lunghezza n , sia:

$w \in \{a, b\}^* \mid |w| = n$ con a e b in egual numero, $m(a) = m(b)$ con $m()$ che indica il numero

quindi si ha che:

$$w = aw_1bw_2 \text{ o } w = bw_1aw_2$$

sia.

$$k_1 = m(a) \in w_1 = m(b) \in w_1$$

$$k_2 = m(a) \in w_2 = m(b) \in w_2$$

allora:

$$k_1 + k_2 + 1 = m(a) \in w = m(b) \in W$$

sapendo che $|w_1| < n$ e $|w_2| < n$ allora w_1 e w_2 sono egnerati da G per ipotesi induttiva

1.1.1 Alberi Sintatici

Definizione 5. Data una grammatica CFG, $G = \{V, T, P, S\}$ un **albero sintattico** per G soddisfa le seguenti condizioni:

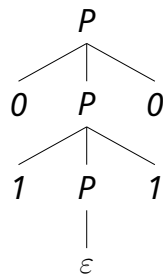
- ogni nodo interno è etichettato con una variabile
- ogni foglia è anch'essa etichettata con una variabile o col simbolo di terminale T o con la stringa vuota ε (in questo caso la foglia è l'unico figlio del padre)
- se un nodo interno è etichettato con A i suoi figli saranno etichettati con X_1, \dots, X_k e $A \rightarrow X_1, \dots, X_k$ sarà una produzione di G . Se un X_i è ε sarà l'unica figlio e $A \rightarrow \varepsilon$ sarà comunque una produzione di G

La concatenazione in ordine delle foglie viene detto **prodotto dell'albero**

Esempio 22. Usiamo l'esempio delle stringhe palindrome:

$$P \rightarrow 0P0 \mid 1P1 \mid \varepsilon$$

sia il seguente albero sintattico:

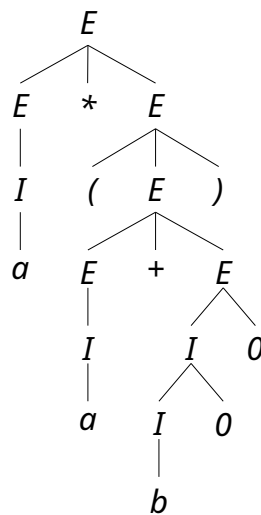


Esempio 23. Si ha:

$$E \rightarrow I \mid E + E \mid E * E \mid (E)$$

$$I \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1$$

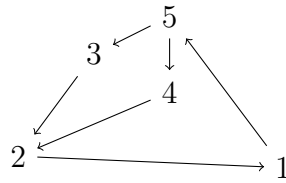
un albero sintattico per $a * (a + b00)$ può essere:



Data una CFG si ha che i seguenti cinque enunciati si equivalgono:

1. la procedura di inferenza ricorsiva stabilisce che una stringa w di simboli terminali appartiene al linguaggio $L(A)$ con A variabile
2. $A \rightarrow^* w$
3. $A \rightarrow_{lm}^* w$
4. $A \rightarrow_{rm}^* w$
5. esiste un albero sintattico con radice A e prodotto w

queste 5 proposizioni si implicano l'uni l'altra:



vediamo qualche dimostrazione di implicazione tra queste proposizioni:

da 1 a 5. si procede per induzione:

- **caso base:** ho un livello solo (una sola riga), $\exists A \rightarrow w$:

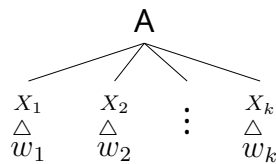
$$\begin{array}{c} A \\ \triangle \\ w \end{array}$$

- **caso passo:** suppongo vero per un numero di righe $\leq n$, lo dimostro per $n + 1$ righe:

$$A \rightarrow X_1, X_2, \dots, X_k$$

$$w = w_1, w_2, \dots, w_k$$

ovvero, in meno di $n + 1$ livelli:



□

da 5 a 3. procedo per induzione:

- **caso base (n=1):** $\exists A \rightarrow w$ quindi $A \rightarrow_{lm} w$, come prima si ha un solo livello:

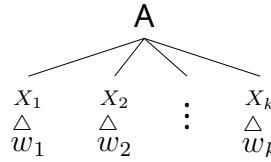
$$\begin{array}{c} A \\ \triangle \\ w \end{array}$$

- **caso passo:** suppongo che la proprietà valga per ogni albero di profondità minore uguale a n , dimostro che valga per gli alberi profondi $n + 1$:

$$A \rightarrow X_1, X_2, \dots, X_k$$

$$w = w_1, w_2, \dots, w_k$$

ovvero, in meno di $n + 1$ livelli:



$$A \rightarrow_{lm} X_1, X_2, \dots, X_k$$

$x_1 \rightarrow_{lm}^* w_1$ per ipotesi induttiva si ha un albero al più di n livelli quindi:

$$A \rightarrow_{lm} X_1, \dots, X_k \rightarrow_{lm}^* w_1, X_2, \dots, X_k \rightarrow_{lm}^* \dots \rightarrow_{lm}^* w_1, \dots, w_k = w$$

Esempio 24.

$$E \rightarrow I \rightarrow Ib \rightarrow ab$$

$$\alpha E \beta \rightarrow \alpha I \beta \rightarrow \alpha Ib \beta \rightarrow \alpha ab \beta, \quad \alpha, \beta \in (V \cup T)^*$$

□

Esempio 25. Mostro l'esistenza di una derivazione sinistra dell'albero sintattico di $a * (a + b00)$:

$$\begin{aligned} E &\rightarrow_{lm}^* E * E \rightarrow_{lm}^* I * E \rightarrow_{lm}^* a * E \rightarrow_{lm}^* a * (E) \rightarrow_{lm}^* a * (E + E) \rightarrow_{lm}^* \\ a * (I + E) &\rightarrow_{lm}^* a * (a + E) \rightarrow_{lm}^* a * (a + I) \rightarrow_{lm}^* a + (a + I0) \rightarrow_{lm}^* a * (a + I00) \rightarrow_{lm}^* a * (a + b00) \end{aligned}$$

1.1.2 Grammatiche ambigue

Definizione 6. Una grammatica è definita ambigua se esiste una stringa w di terminali che ha più di un albero sintattico

Esempio 26. vediamo un esempio:

1. $E \rightarrow E + E \rightarrow E + E * E$ ovvero:



2. $E \rightarrow E * E \rightarrow E + E * E$ ovvero:



si arriva a due stringhe uguali ma con alberi diversi. Introduciamo delle categorie sintattiche, dei vincoli alla produzione delle regole:

1. $E \rightarrow T \mid E + T$
2. $T \rightarrow F \mid T * F$
3. $F \rightarrow I \mid (E)$
4. $I \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1$

Possono esserci più derivazioni di una stringa ma l'importante è che non ci siano alberi sintattici diversi. Capire se una CFG è ambigua è un problema indecidibile

Esempio 27. vediamo un esempio:

$$S \rightarrow \varepsilon \mid SS \mid iS \mid iSeS$$

con S =statement, i =if e e =else. Considero due derivazioni:

1. $S \rightarrow iSeS \rightarrow iiSeS \rightarrow iie$:



2. $S \rightarrow iS \rightarrow iiSeS \rightarrow iieS \rightarrow iie$:



Si ha quindi una grammatica ambigua

Teorema 2. Per ogni CFG, con $G = (V, T, P, S)$, per ogni stringa w di terminali si ha che w ha due alberi sintattici distinti sse ha due derivazioni sinistre da S distinte.

Se la grammatica non è ambigua allora esiste un'unica derivazione sinistra da S

Linguaggi inerentemente ambigui

Definizione 7. Un linguaggio L è inerentemente ambiguo se tutte le grammatiche CFG per tale linguaggio sono a loro volta ambigue

Esempio 28. Sia $L = \{a^n b^n c^m d^m \mid n, m \geq 1\} \cup \{a^n b m n c^m d^n \mid n, m \geq 1\}$ si ha quindi un CFL formato dall'unione di due CFL. L è inerentemente ambiguo e generato dalla seguente grammatica:

- $S \rightarrow AB \mid C$

- $A \rightarrow aAb \mid ab$
- $B \rightarrow cBd \mid cd$
- $C \rightarrow aCd \mid aDd$
- $D \rightarrow bDc \mid bc$

si possono avere due derivazioni:

1. $S \rightarrow_{lm} AB \rightarrow_{lm} aAbB \rightarrow_{lm} aabbB \rightarrow_{lm} aabbcBd \rightarrow_{lm} aabbccdd$
2. $S \rightarrow_{lm} C \rightarrow_{lm} aCd \rightarrow_{lm} aaBdd \rightarrow_{lm} aabBcdd \rightarrow_{lm} aabbccdd$

a generare problemi sono le stringhe con $n=m$ perché possono essere prodotte in due modi diversi da entrambi i sottolinguaggi. Dato che l'intersezione tra i due sottolinguaggi non è vuota si ha che L è ambiguo

1.1.3 Grammatiche Regolari

Sono le grammatiche che generano i linguaggi regolari (quelli del terzo tipo) che sono casi particolari dei CFL.

Si ha la solita grammatica $G = (V, T, P, S)$ con però vincoli su P :

- ε si può ottenere solo con $S \rightarrow \varepsilon$
- le produzioni sono tutte lineari a destra ($A \rightarrow aA$ o $A \rightarrow a$) o a sinistra ($A \rightarrow Ba$ o $A \rightarrow a$)

Esempio 29. $I \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1$ è una grammatica con le produzioni lineari a sinistra.

Potremmo pensarlo a destra $I \rightarrow a \mid b \mid aI \mid bI \mid 0I \mid 1I$.

Vediamo esempi di produzione con queste grammatiche:

- con $I \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1$ possiamo derivare $ab01b0$:

$$I \rightarrow I0 \rightarrow Ib0 \rightarrow I1b0 \rightarrow I01b0 \rightarrow Ib01b0 \rightarrow ab01b0$$

- con $I \rightarrow a \mid b \mid aI \mid bI \mid 0I \mid 1I$ invece non riusciamo a generare nulla:

$$I \rightarrow 0I \rightarrow 0a$$

definisco quindi un'altra grammatica (con una nuova categoria sintattica):

$$I \rightarrow aJ \mid bJ$$

$$J \rightarrow a \mid b \mid aJ \mid bJ \mid 0J \mid 1J$$

che però non mi permette di terminare le stringhe con 0 e 1, la modifico ancora ottenendo:

$$I \rightarrow aJ \mid bJ$$

$$J \rightarrow a \mid b \mid aJ \mid bJ \mid 0J \mid 1J \mid 0 \mid 1$$

e questo è il modo corretto per passare da lineare sinistra a lineare destra

Esempio 30. Sia $G = (\{S\}, \{0, 1\}, P, S)$ con $S \rightarrow \varepsilon \mid 0 \mid 1 \mid 0S \mid 1S$. Si ha quindi:

$$L(G) = \{0, 1\}^*$$

si hanno comunque due proposizioni ridondanti, riducendo trovo:

$$S \rightarrow \varepsilon \mid 0S \mid 1S$$

con solo produzioni lineari a destra. Con produzioni lineari a sinistra ottengo:

$$S \rightarrow \varepsilon \mid S0 \mid S1$$

Esempio 31. Trovo una grammatica lineare destra e una sinistra per $L = \{a^n b^m \mid n, m \geq 0\}$:

- **lineare a destra:** si ha $G = (\{S, B\}, \{a, b\}, P, S)$ e quindi:

$$S \rightarrow \varepsilon \mid aS \mid bB$$

$$B \rightarrow bB \mid b$$

ma non si possono generare stringhe di sole b , infatti:

$$S \rightarrow aS \rightarrow abB \rightarrow abbB \rightarrow abbb$$

ma aggiungere ε a B **non è lecito**. posso però produrre la stessa stringa da due derivazioni diverse:

$$S \rightarrow \varepsilon \mid aS \mid bB \mid b$$

$$B \rightarrow bB \mid b$$

che risulta quindi la nostra lineare a destra

- **lineare a sinistra:** si ha $G = (\{S, A\}, \{a, b\}, P, S)$ e quindi:

$$S \rightarrow \varepsilon \mid Sb \mid Ab \mid a$$

$$A \rightarrow Aa \mid a$$

Esempio 32. Trovo una grammatica lineare destra e una sinistra per $L = \{ab^n cd^m e \mid n \geq 0, m > 0\}$:

- **lineare a destra:** si ha $G = (\{S, A, B, E\}, \{a, b, c, d, e\}, P, S)$ e quindi:

$$S \rightarrow aA$$

$$A \rightarrow bA \mid cB$$

$$B \rightarrow dB \mid dE$$

$$E \rightarrow e$$

- **lineare a sinistra:** si ha $G = (\{S, X, Y, Z\}, \{a, b, c, d, e\}, P, S)$ e quindi:

$$S \rightarrow Xe$$

$$A \rightarrow Xd \mid Yd$$

$$B \rightarrow Zc$$

$$E \rightarrow a \mid Zb$$

quindi se per esempio ho la stringa "ciao" si ha:

- **lineare a destra:**

$$S \rightarrow Ao$$

$$A \rightarrow Ba$$

$$B \rightarrow Ei$$

$$E \rightarrow c$$

- **lineare a sinistra:**

$$S \rightarrow cA$$

$$A \rightarrow iB$$

$$B \rightarrow aE$$

$$E \rightarrow o$$

Esempio 33. A partire da $G = (\{S, T\}, \{0, 1\}, P, S)$ con:

$$S \rightarrow \varepsilon \mid 0S \mid 1T$$

$$T \rightarrow 0T \mid 1S$$

trovo come è fatto $L(G)$:

$$L(G) = \{w \in \{0, 1\}^* \mid w \text{ ha un numero di 1 pari}\}$$

Esempio 34. fornire una grammatica regolare a destra e sinistra per:

$$L = \{w \in \{0, 1\}^* \mid w \text{ ha almeno uno 0 o almeno un 1}\}$$

Si ha che tutte le stringhe tranne quella vuota contengono uno 0 o un 1 quindi $G = (\{S\}, \{0, 1\}, P, S)$:

- **lineare a destra:**

$$S \rightarrow 0 \mid 1 \mid 0S \mid 1S$$

- **lineare a sinistra:**

$$S \rightarrow 0 \mid 1 \mid S0 \mid S1$$

1.1.4 Espressioni Regolari (Regex)

le regex sono usate per la ricerca di un pattern in un testo o negli analizzatori lessicali. Una regex denota il linguaggio e non la grammatica. Si hanno le seguenti operazioni tra due linguaggi L e M :

- **unione:** dati $L, M \in \Sigma^*$, l'unione $L \cup M$ è l'insieme delle stringhe che si trovano in entrambi i linguaggi o solo in uno dei due

Esempio 35.

$$L = \{001, 10, 111\}$$

$$M = \{\varepsilon, 001\}$$

$$L \cup M = \{\varepsilon, 01, 10, 111, \varepsilon\}$$

si ha che:

$$L \cup M = M \cup L$$

- **concatenazione:** dati $L, M \in \Sigma^*$, la concatenazione $L \cdot M$ (o LM) è l'insieme di tutte le stringhe ottenibili concatenandone una di L a una di M

Esempio 36.

$$L = \{001, 10, 111\}$$

$$M = \{\varepsilon, 001\}$$

$$L \cdot M = \{001, 001001, 10, \dots\}$$

si ha che:

$$L \cdot M \neq M \cdot L$$

- si definiscono:

$$- L \cdot L = L^2, L \cdot L \cdot L = L^3 \text{ etc...}$$

$$- L^1 = L$$

$$- L^0 = \{\varepsilon\}$$

- **chiusura di Kleene:** dato $L \subseteq \Sigma^*$ si ha che la chiusura di Kleene di L è:

$$L^* = \bigcup_{i \geq 0} L^i$$

ricordando che $L^0 = \varepsilon$

Esempio 37. Sia $L = \{0, 11\}$, si ha:

$$L^0 = \varepsilon$$

$$L^1 = L = \{0, 11\}$$

$$L^2 = L \cdot L = \{00, 011, 110, 1111\}$$

$$L^3 = L \cdot L \cdot L = L^2 \cdot L = \{000, 0110, 1100, 11110, 0011, 01111, 11011, 111111\}$$

vediamo dei casi particolari:

- $L = \{0^n \mid n \geq 0\}$ implica $|L| = \infty$ e quindi, essendo $L^i = L$, $i \geq 1$ e quindi $|L^i| = \infty$, $|L^*| = \infty$. Si ha quindi:

$$L^* = L^0 \cup L^1 \cup \dots \cup L^i = L$$

- $L = \emptyset$ implica $L^0 = \{\varepsilon\}$, $L^2 = L \cdot L = \emptyset$ e così via per ogni concatenazione di L . Si ha quindi:

$$L^* = L^0 = \{\varepsilon\}$$

– $L = \{\varepsilon\}$ implica $L^0 = \{\varepsilon\} = L = L^1 = L^2 = \dots$, si ha quindi:

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

L'insieme vuoto e l'insieme contenente la stringa vuota hanno le uniche chiusure di Kleene finite

Definizione 8. *Si riporta la definizione ricorsiva di un'espressione regolare:*

• **casi base:** *si hanno tre casi base:*

1. ε e \emptyset sono espressioni regolari
2. se $a \in \Sigma$ a è un'espressione regolare, $L(a) = \{a\}$
3. le variabili che rappresentano linguaggi regolari sono espressioni regolari, $L(L) = L$

• **casi passo:** *si hanno i 4 casi passo:*

1. **unione:** *se E e F sono espressioni regolari allora anche $E + F = E \cup F$ è un'espressione regolare e si ha:*

$$L(E + F) = L(E) \cup L(F)$$

2. **concatenazione:** *se E e F sono espressioni regolari allora anche $EF = E \cdot F$ è un'espressione regolare e si ha:*

$$L(EF) = L(E) \cdot L(F)$$

3. **chiusura:** *se E è un'espressione regolare allora E^* è un'espressione regolare e si ha:*

$$L(E^*) = (L(E))^*$$

4. **parentesi:** *se E è un'espressione regolare allora (E) è un'espressione regolare e si ha:*

$$L((E)) = L(E)$$

Esempio 38. *trovo regex per l'insieme di stringhe in $\{0,1\}^*$ che consistono in 0 e 1 alternati:*

$$01 \rightarrow \{01\}$$

$$(01)^* \rightarrow \{\varepsilon, 01, 0101, 010101, \dots\}$$

$$(01)^* + (10)^* \rightarrow \{\varepsilon, 01, 10, 0101, 1010, \dots\}$$

ma posso volere diverse quantità di 0 e 1, sempre mantenendo l'alternanza, metto o uno 0 o un 1 davanti a quanto ottenuto appena sopra:

$$(01)^* + (10)^* + 0(10)^* + 1(01)^* \rightarrow \{\varepsilon, 01, 10, 010, 101, \dots\}$$

non è comunque l'unica soluzione, si può avere:

$$(\varepsilon + 1)(01)^*(\varepsilon + 0) \rightarrow \{\varepsilon, 01, 10, 010, 101, \dots\}$$

oppure ancora:

$$(\varepsilon + 0)(10)^*(\varepsilon + 1)$$

Si ha una precedenza degli operatori, in ordine di precedenza (si valuta da sinistra a destra):

1. chiusura di Kleene $*$
2. concatenazione \cdot , che è associativo $((E \cdot F) \cdot G = E \cdot (F \cdot G))$ ma non è commutativo $(E \cdot F \neq F \cdot E)$
3. unione $+$ che è associativa $((E+F)+G = E+(F+G))$ ed è commutativo $(E + F = F + E)$
4. infine le parentesi

si hanno anche delle proprietà algebriche:

- due espressioni regolari sono equivalenti se denotano lo stesso linguaggio
- due espressioni regolari con variabili sono equivalenti se lo sono \forall assegnamento alle variabili
- l'unione è commutativa e associativa, la concatenazione è solo associativa
- si definiscono:

- **identità:** ovvero un valore unito all'identità è pari a se stesso (elemento neutro della somma $0 + x = x + 0 = x$). \emptyset è identità per l'unione ($\emptyset + L = L + \emptyset = L$), $\{\varepsilon\}$ è identità per la concatenazione ($\varepsilon L = L\varepsilon = L$)
- **annichilitore:** ovvero un valore concatenato all'annichilitore da l'annichilitore (l'elemento nullo del prodotto $0x = x0 = 0$). \emptyset è l'annichilitore per la concatenazione ($\emptyset L = L\emptyset = \emptyset$)
- **distributività:** dell'unione rispetto alla concatenazione (che non è commutativa):
 - **distributività sinistra:** $L(M + N) = LM + LN$
 - **distributività destra:** $(M + N)L = ML + NL$
- **idempotenza:** $L + L = L$
- $(L^*)^* = L^*$
- $\emptyset^* = \varepsilon$ infatti $L(\emptyset) = \{\varepsilon\} \cup L(\emptyset) \cup L(\emptyset) \cdot L(\emptyset) \cup \dots = \{\varepsilon\} \cup \emptyset \cup \emptyset \dots = \varepsilon$
- $\varepsilon^* = \varepsilon$ infatti $L(\varepsilon) = \{\varepsilon\} \cup L(\varepsilon) \cup L(\varepsilon) = \{\varepsilon\} \cup \{\varepsilon\} \cup \dots = \{\varepsilon\} = L(\varepsilon)$
- $L^+ = L \cdot L^* = L^* \cdot L$ (quindi con almeno un elemento che non sia la stringa vuota)
- $L^* = l^+ + \varepsilon$

Esempio 39. *Ho $ER = (0 + 1)^*0^*(01)^*$:*

- *001 fa parte del linguaggio? Si: $\varepsilon \cdot 0 \cdot 01$*
- *1001 fa parte del linguaggio? Si: $1 \cdot 0 \cdot 01$*
- *0101 fa parte del linguaggio? Si: $\varepsilon \cdot \varepsilon \cdot 0101$*
- *0 fa parte del linguaggio? Si: $\varepsilon \cdot 0 \cdot \varepsilon$*
- *10 fa parte del linguaggio? Si: $1 \cdot 0 \cdot \varepsilon$*

$L((0+1)^*) = (L(0+1))^* = (L(0)+L(1))^* = (\{0\} \cup \{1\})^* = (\{0, 1\})^* = \{0, 1\}^*$
 ovvero tutte le combinazioni di 0 e 1

Si ricorda che:

$$(0 + 1)^* \neq 0^* + 1^*$$

Esempio 40. *ho* $ER = ((01)^* \cdot 10 \cdot (0 + 1)^*)^*$

- *0101 fa parte del linguaggio? No*
- *01000 fa parte del linguaggio? No*
- *01011 fa parte del linguaggio? No*
- *10111 fa parte del linguaggio? Si, $\varepsilon \cdot 10 \cdot 111$*
- *101010 fa parte del linguaggio? Si, prendo $10 \cdot 1010$*
- *101101 fa parte del linguaggio? Si, $\varepsilon \cdot 10 \cdot 1$ due volte*
- *0101100011 fa parte del linguaggio? Si, $0101 \cdot 10 \cdot 0011$ (0011 lo posso prendere da $(0 + 1)^*$)*

Esempio 41. *ho* $ER = ((01)^* \cdot 10 \cdot (0 + 1))^*$

- *0101 fa parte del linguaggio? No*
- *01000 fa parte del linguaggio? No*
- *01011 fa parte del linguaggio? No*
- *10111 fa parte del linguaggio? No*
- *101010 fa parte del linguaggio? No*
- *101101 fa parte del linguaggio? Si, $\varepsilon \cdot 10 \cdot 1$ due volte*
- *0101100011 fa parte del linguaggio? No*

Esempio 42. *Da* $L \subseteq \{0, 1\}^*$ *stringhe contenenti almeno una volta 01 quindi:*

$$(0 + 1)^* 01 (0 + 1)^*$$

Esempio 43. *ho* $ER = (00^*1^*)^*$, *quindi:*

$$L = \{\varepsilon, 0, 01, 000, 001, 010, 011\} = \{\varepsilon\} \cup \{w \in \{0, 1\}^* \mid w \text{ che inizia con } 0\}$$

Esempio 44. *ho* $ER = a(a + b)^*b$, *quindi:*

$$L = \{w \in \{a, b\}^* \mid w \text{ inizia con } a \text{ e termina con } b\}$$

Esempio 45. *ho* $ER = (0^*1^*)^*000(0 + 1)^*$, *quindi, sapendo che* $\{0, 1\}^*$ *mi permette tutte le combinazioni che voglio come* $(0 + 1)^*$:

$$L = \{w \in \{0, 1\}^* \mid w \text{ come voglio con tre } 0 \text{ consecutivi}\}$$

Esempio 46. ho $ER = a(a+b)^*c(a+b)^*c(a+b)^*b$, quindi:

$L = \{w \in \{a, b, c\}^* \mid w \text{ inizia con } a, \text{ termina con } b \text{ e contiene almeno due } c, \\ \text{eventualmente non adiacenti}\}$

Esempio 47. Da $L \subseteq \{0, 1\}^*$ ogni 1 è seguito da 0, a meno che non sia l'ultimo carattere, ovvero 11 non compare quindi:

$$(10 + 0)^*(\varepsilon + 1)^*$$

Esempio 48. cerco ER per $L \subseteq \{0, 1\}^*$ stringhe contenenti un numero pari di 1:

$$(0^*10^*1)^*0^*$$

oppure:

$$(0 + 10^*1)^*$$

Esempio 49. cerco ER per $L \subseteq \{0, 1\}^*$ stringhe contenenti un numero dispari di 1:

$$(0^*10^*)^*0^*10^*$$

oppure:

$$(0 + 10^*1)^*10^*$$

Esempio 50. cerco ER per $L \subseteq \{0, 1\}^*$ stringhe contenenti un numero divisibile per 3 di 0:

$$(1^*01^*01^*0)^*1^*$$

Esempio 51. cerco ER per $L \subseteq \{0, 1\}^*$ stringhe contenenti al più una coppia di 1 consecutivi:

$$(10 + 0)^*(11 + 1 + \varepsilon)(01 + 0)^*$$

Esempio 52. cerco ER per $L \subseteq \{a, b, c\}^*$ stringhe contenenti almeno una a e almeno una b :

$$c^*(a(a+c)^*b + b(b+c)^*a)(a+b+c)^*$$

1.2 Automi

un automa a stati finiti ha un insieme di stati e un controllo che si muove da stato a stato in risposta a input esterni. Si ha una distinzione:

- **automi deterministici:** dove l'automa non può essere in più di uno stato per volta
- **automi non deterministici:** dove l'automa può trovarsi in più stati contemporaneamente

1.2.1 Automi deterministici

Un automa a stati finiti deterministico (*DFA*), un automa che dopo aver letto una qualunque sequenza di input si trova in un singolo stato. Il termine *deterministico* concerne il fatto che per ogni input esiste un solo stato verso il quale l'automa passa dal suo stato corrente. Un automa a stati finiti deterministico consiste nelle seguenti parti:

- un insieme finito di stati, spesso indicato con Q
- un insieme finito di simboli di input, spesso indicato con Σ
- una funzione di transizione, che prende come argomento uno stato e un simbolo di input e restituisce uno stato. La funzione di transizione sarà indicata comunemente con δ . Nella rappresentazione grafica informale di automi δ è rappresentata dagli archi tra gli stati e dalle etichette sugli archi. Se q è uno stato e a è un simbolo di input, $\delta(q, a)$ è lo stato p tale che esiste un arco etichettato con a da q a p ²
- uno stato iniziale, uno degli stati in Q
- un insieme di stati finali, o accettanti, F . L'insieme F è un sottoinsieme di Q

Nel complesso un DFA è rappresentato in maniera concisa con l'enumerazione dei suoi elementi, quindi con la quintupla:

$$A = (Q, \Sigma, \delta, q_0, F)$$

con A nome del DFA, Q insieme degli stati, Σ rappresentante i simboli di input, δ la sua funzione di transizione, q_0 il suo stato iniziale e F l'insieme degli stati accettanti.

Vediamo come decidere se accettare o meno una stringa (sequenza di caratteri) in input mediante un DFA.

Ho una sequenza in input $a_1 \dots a_n$. Parto dallo stato iniziale q_0 , consultando la funzione di transizione δ , per esempio $\delta(q_0, a_1) = q_1$ e trovo lo stato in cui il DFA entra dopo aver letto a_1 . Poi passo a $\delta(q_1, a_2) = q_2$ e così via, $\delta(q_{i-1}, a_i) = q_i$ fino a ottenere q_n . Se q_n è elemento di F allora $a_1 \dots a_n$ viene accettato, altrimenti viene rifiutato.

Esempio 53. *specifico DFA che accetta tutte le stringhe binarie in cui compare la sequenza 01:*

$$L = \{w \mid w \text{ è della forma } x01y, \text{ con } x \text{ e } y \text{ pari a } 0 \text{ o } 1\} = \{01, 11010, 100011, \dots\}$$

o anche:

$$L = \{x01y | x, y \in \{0, 1\}^*\}$$

abbiamo quindi:

$$\Sigma = \{0, 1\}$$

ragioniamo sul fatto che A :

1. se ha "già visto" 01 , accetterà qualsiasi input
2. pur non avendo ancora visto 01 , l'input più recente è stato 0 , cosicché se ora vede un 1 avrà visto 01
3. non ha ancora visto 01 , ma l'input più recente è nullo (siamo all'inizio), in tal caso A non accetta finché non vede uno 0 e subito dopo un 1

la terza condizione rappresenta lo stato iniziale. All'inizio bisogna vedere uno 0 e poi un 1 . Ma se nello stato q_0 si vede per primo un 1 allora non abbiamo fatto alcun passo verso 01 , e dunque dobbiamo permanere nello stato q_0 , $\delta(q_0, 1) = q_0$. D'altra parte se nello stato iniziale vedo 0 siamo nella seconda condizione, uso quindi q_2 per questa condizione, si avrà quindi $\delta(q_0, 0) = q_2$. Vedo ora le transizioni di q_2 , se vedo 0 ho che 0 è l'ultimo simbolo incontrato quindi uso nuovamente q_2 , $\delta(q_2, 0) = q_2$, in attesa di un 1 . Se arriva 1 passo allo stato accertante q_1 corrispondente alla prima condizione, $\delta(q_2, 1) = q_1$. Ora abbiamo incontrato 01 quindi può succedere qualsiasi cosa e dopo qualsiasi cosa accada potremo nuovamente aspettarci qualsiasi cosa, ovvero $\delta(q_1, 0) = \delta(q_1, 1) = q_1$. Si deduce quindi che:

$$Q = \{q_0, q_1, q_2\} \text{ e } F = \{q_1\}$$

quindi:

$$A = \{\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_1\}\}$$

con in totale le seguenti transizioni:

$$\delta(q_0, 1) = q_0$$

$$\delta(q_0, 0) = q_2$$

$$\delta(q_2, 0) = q_2$$

$$\delta(q_2, 1) = q_1$$

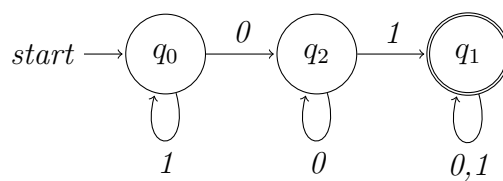
$$\delta(q_1, 0) = q_1$$

$$\delta(q_1, 1) = q_1$$

posso rappresentarle in maniera tabulare, con lo stato iniziale indicato da \rightarrow e quelli accettanti con $*$:

δ	0	1
$\rightarrow q_0$	q_1	q_0
$* q_1$	q_1	q_1
q_2	q_2	q_1

o col diagramma di transizione:



Esempio 54. Trovo automa per:

$$L = \{w \in \{a, b\}^* \mid w \text{ che contiene un numero pari di } b\}$$



ovvero se da q_0 vado a q_1 sono obbligato ab generare due b , dato che il nodo accettante è q_0 . In entrambi i nodi posso generare quante a voglio.

Esempio 55. Trovo automa per:

$$L = \{w \in \{a, b\}^* \mid w \text{ che contiene un numero dispari di } b\}$$



ovvero se da q_0 vado a q_1 sono obbligato ab generare una sola b , dato che il nodo accettante è q_1 . In entrambi i nodi posso generare quante a voglio e posso tornare da q_1 a q_0 per generare altre b .

Esempio 56. Trovo automa per:

$$L = \{w \in \{0, 1\}^* \mid w = 0^n 1^m\}$$

vediamo i vari casi: **Si ha che q_E è lo stato pozzo dove vanno le stringhe venute male**

- $n, m \geq 0$:



ovvero posso non generare nulla e uscire subito con q_0 , generare solo un 1 e passare a q_1 e uscire oppure generare 0 e 1 a piacere con l'ultimo stato o generare 0 a piacere dal primo e 1 a piacere dal secondo.

- $n \geq 0 \ m > 0$:



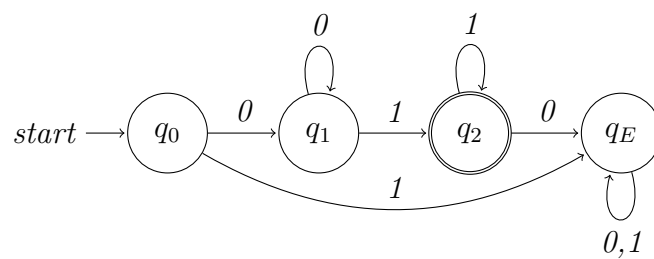
ovvero come l'esempio sopra solo che non posso uscire in q_0 in quanto almeno un 1 deve essere per forza generato

- $n > 0 \ m \geq 0$:



CHIARIRE

- $n, m > 0$:



CHIARIRE

Esempio 57. Trovo automa per:

$$L = \{w \in \{a, b\}^* \mid w \text{ che contiene un numero pari di } a \text{ e dispari di } b\}$$



Esempio 58. Trovo automa per:

$$L = \{w \in \{a, b\}^* \mid w \text{ che contiene un numero pari di } a \text{ seguito da uno dispari di } b\}$$

$$L = \{a^{2n}b^{2k+1} \mid j, k \geq 0\}$$



ovvero in tabella:

δ	a	b
$\rightarrow q_0$	q_1	q_2
q_1	q_0	q_E
$* q_2$	q_E	q_3
q_3	q_E	q_2
q_E	q_E	q_E

Esempio 59. Trovo automa per:

$$L = \{a^{2k+1}b^{2h} \mid h, k \geq 0\}$$



Esempio 60. Trovo automa per:

$$L = \{a^{2n+1}b^{2k+1} \mid n, k \geq 0\}$$



Esempio 61. Trovo automa per:

$$L = \{x010y \mid x, y \in \{0, 1\}^*\}$$



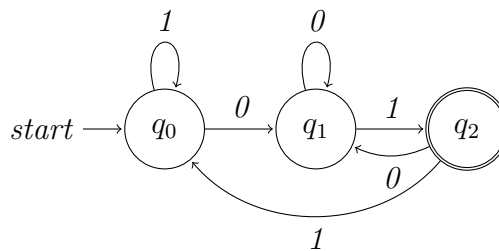
1.2.2 Automi non deterministici

Un automa a stati finiti non deterministici (*NFA*) può trovarsi in diversi stati contemporaneamente. Come i DFA accettano linguaggi regolari e spesso sono più semplici da trattare rispetto ai DFA.

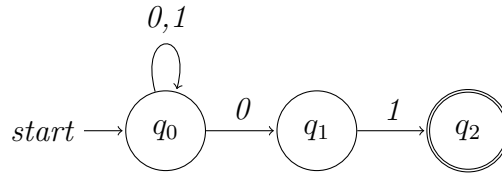
Un NFA è definito come un DFA ma si ha un diverso tipo di transizione δ , che ha sempre come argomenti uno stato e un simbolo di input ma restituisce zero o più stati.

Esempio 62. Sia $L = \{x01 \mid x \in \{0, 1\}^*\}$ ovvero il linguaggio formato da tutte le stringhe binarie che terminano in 01.

Avremo il seguente automa deterministico:



che diventa il seguente NFA:



quindi con:

$$\delta(q_0, 0) = \{q_0, q_1\}$$

$$\delta(q_0, 1) = \{q_0\}$$

$$\delta(q_1, 0) = \emptyset$$

$$\delta(q_1, 1) = \{q_2\}$$

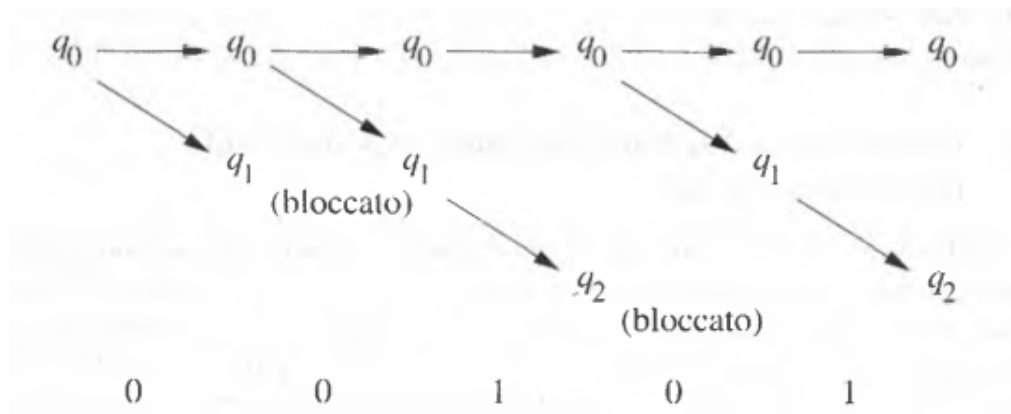
$$\delta(q_2, 0) = \emptyset$$

$$\delta(q_2, 1) = \emptyset$$

in forma tabulare:

δ	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_1\}$
q_1	\emptyset	$\{q_2\}$
$*q_0$	\emptyset	\emptyset

vediamone la simulazione per la stringa 00101:



ovvero si parte dallo stato iniziale, quando viene letto 0 si passa a q_0 e q_1 , poi viene letto il secondo 0 quindi q_0 va nuovamente verso q_0 e q_1 mentre il primo q_1 muore non avendo transizioni su 0. Arriva poi l'1 quindi q_0 va solon verso q_0 e q_1 verso q_2 e sarebbe accettante ma l'input non è finito. Ora arriva 0 e q_2 si blocca mentre q_0 va sia in q_0 che in q_1 . Arriva infine un 1 che manda q_0 in q_0 e q_1 in q_2 che è accettante e non avendo altri input si è dimostrata l'appartenenza della stringa al linguaggio

definisco quindi un NFA come una quintupla:

$$A = (Q, \Sigma, \delta, q_0, F)$$

con, a differenza dei DFA:

$$\delta : Q \times F \rightarrow 2^Q$$

Possiamo ora definire δ , delta cappuccio che prende in ingresso uno stato e l'intera stringa w . Definisco ricorsivamente:

- **caso base:** se $|w| = 0$ ovvero se $W = \varepsilon$ si ha:

$$\hat{\delta}(q, \varepsilon) = \{q\}$$

- **caso passo:** se $|w| > 0$, allora $W = xa$, $a \in \Sigma$ e $x \in \Sigma^*$. Posto $\hat{\delta}(q, x) = \{p_1, \dots, p_k\}$ si ha:

$$\hat{\delta}(q, w) = \cup \delta(p_i, a)$$

Per il linguaggio L accettato dall'automa si ha:

$$L(A) = \{w \in \Sigma^* \mid \hat{\delta}(q_0, w) \cap F \neq \emptyset\}$$

Esempio 63. Automa per $L = \{x010y \mid x, y \in \{0, 1\}^*\}$ ovvero tutte le stringhe con dentro la sequenza 010:



Troviamo ora un algoritmo che trasformi un NFA in un DFA. Dall'ultimo esempio ricavo:

	0	1
\emptyset	\emptyset	\emptyset
$\rightarrow \{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_1\}$	\emptyset	$\{q_2\}$
$* \{q_2\}$	\emptyset	\emptyset
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$* \{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0\}$
$* \{q_1, q_2\}$	\emptyset	$\{q_2\}$
$* \{q_0, q_1, q_2\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$

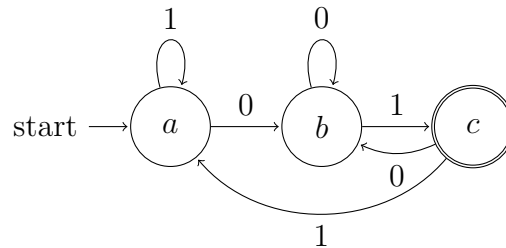
ovvero:



che è il DFA che si era anche prima ottenuto. Si hanno però dei sottoinsiemi mai raggiungibili. Si ha quindi:

	0	1
$\rightarrow \{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0\}$

e definendo $\{q_0\} = a$, $\{q_0, q_1\} = b$ e $\{q_0, q_2\} = c$ si avrà:



Definiamo questo algoritmo che avrà:

- come input un NFA $N = (Q_n, \Sigma, \delta_N, q_0, F_N)$
- come output un DFA $D = (Q_D, \Sigma, \delta_D, \{q_0\}, F_D)$ tale che $L(D) = L(N)$

con:

- $Q_D = 2^{Q_N}$ (quindi se $Q_N = n$ si ha $|Q_D| = 2^n$)
- $F_D = \{S \subseteq Q_n \mid S \cap F_N \neq \emptyset\}$

- $\forall S \subseteq Q_N$ e $\forall a \in \Sigma$:

$$\delta_D(S, a) = \cup \delta_n(p, a)$$

per esempio:

$$\delta_D(\{q_0, q_1, q_2\}, 0) = \delta_N(q_0, 0) \cup \delta_N(q_1, 0) \cup \delta_N(q_2, 0)$$

Si definisce l' ε -NFA, l'automa a stati finiti non deterministici con ε transizioni. Con la transizione ε posso saltare i nodi, ovvero avanza senza aggiungere caratteri

Esempio 64. Si ha $ER = a^*b^*c^*$, che genera:

$$L = \{a^n b^m c^k \mid n, m, k \geq 0\}$$

si ha:



ovvero con ε posso per esempio generare quante a voglio da q_0 e passare a q_2 , uscendo senza generare altro

Si definisce la funzione $ECLOSE : Q \rightarrow 2^Q$, con $ECLOSE(q)$ insieme degli stati raggiungibili da q tramite ε -mosse. Nell'esempio precedente si avrebbe:

$$ECLOSE(q_0) = \{q_0, q_1, q_2\}$$

$$ECLOSE(q_1) = \{q_1, q_2\}$$

$$ECLOSE(q_2) = \{q_2\}$$

si ha inoltre che:

- $ECLOSE 2^Q \rightarrow 2^Q \quad P \subseteq Q$
- $ECLOSE(P) = \cup ECLOSE(p)$
- $ECLOSE(\emptyset) = \emptyset$

mettendo in tabella l'esempio precedente si ha:

	a	b	c
$* \rightarrow \{q_0, q_1, q_2\}$	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$
$*\{q_1, q_2\}$	\emptyset	$\{q_1, q_2\}$	$\{q_2\}$
$*\{q_2\}$	\emptyset	\emptyset	$\{q_2\}$
\emptyset	\emptyset	\emptyset	\emptyset

riscrivendo:

- $a = \{q_0, q_1, q_2\}$
- $b = \{q_1, q_2\}$
- $c = \{q_2\}$
- $d = \emptyset$

ovvero:

$$\begin{aligned}
 \delta_D(\{q_0, q_1, q_2\}, a) &= ECLOSE(\delta_N(q_0, a) \cup \delta_N(q_1, a) \cup \delta_N(q_2, a)) \\
 &= ECLOSE(\{q_0\} \cup \emptyset \cup \emptyset) = ECLOSE(\{q_0\}) \\
 &= ECLOSE(q_0) = \{q_0, q_1, q_2\}
 \end{aligned}$$

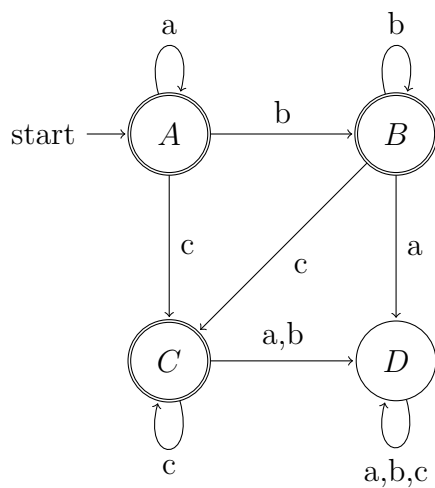
e

$$\begin{aligned}
 \delta_D(\{q_0, q_1, q_2\}, b) &= ECLOSE(\delta_N(q_0, b) \cup \delta_N(q_1, b) \cup \delta_N(q_2, b)) \\
 &= ECLOSE(\emptyset \cup \{q_1\} \cup \emptyset) = ECLOSE(\{q_1\}) \\
 &= ECLOSE(q_1) = \{q_1, q_2\}
 \end{aligned}$$

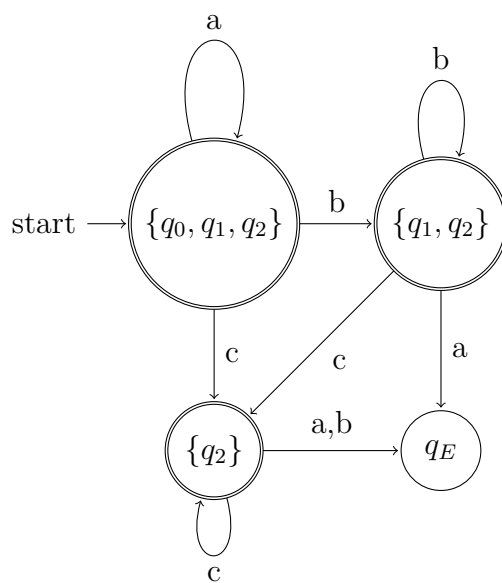
e

$$\begin{aligned}
 \delta_D(\{q_0, q_1, q_2\}, c) &= ECLOSE(\delta_N(q_0, c) \cup \delta_N(q_1, c) \cup \delta_N(q_2, c)) \\
 &= ECLOSE(\emptyset \cup \emptyset \cup \{q_2\}) = ECLOSE(\{q_2\}) \\
 &= ECLOSE(q_2) = \{q_2\}
 \end{aligned}$$

si ottiene quindi il seguente NFA:



che diventa il seguente DFA:



esercizi

Esempio 65. automa DFA per $w = x010y$, $x, y \in \{0, 1\}^*$:
la stringa più corta è 010



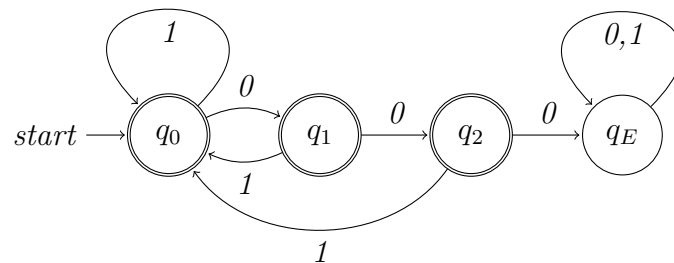
Esempio 66. automa DFA per $a^{2k+1}b^{2h}$, $h, k \geq 0$:
concatenazione di a dispari e b pari:



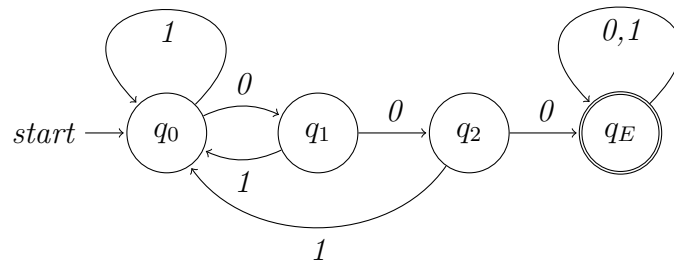
Esempio 67. cerco DFA per stringhe inizianti con a e finenti con b , con
occorrenze di b singole o a coppie, nessuna regola per c .
per esempio $abbcb$ è nel linguaggio



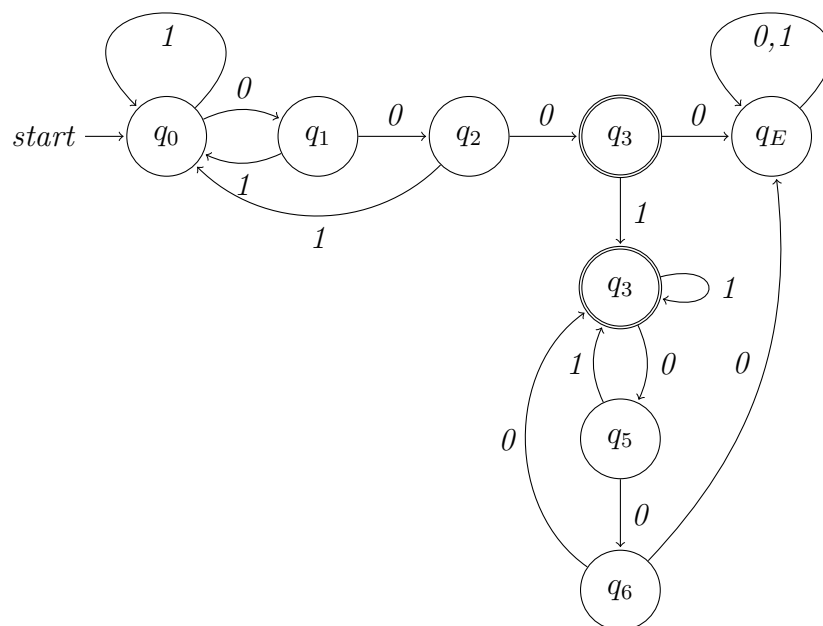
Esempio 68. *cerco DFA per stringhe di bit non contegano 000*



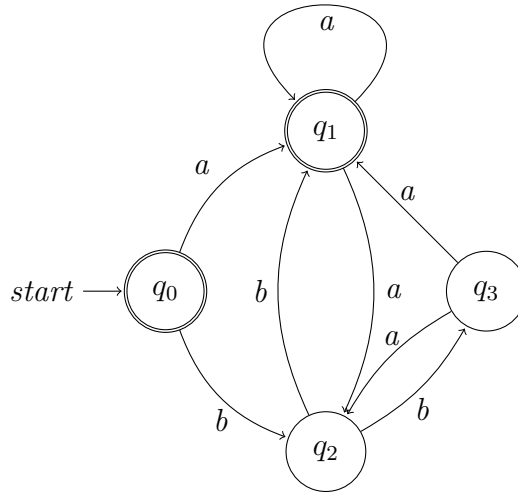
Esempio 69. *cerco DFA per stringhe di bit non contegano 000 almeno una volta*



Esempio 70. *cerco DFA per stringhe di bit che contengono 000 solo una volta*



Esempio 71. Trasformare il seguente NFA in un DFA:



abbiamo quindi:

$$\delta_D(\{q_0\}, a) = \delta_N(q_0, a) = \{q_1, q_2\}$$

$$\delta_D(\{q_0\}, b) = \delta_N(q_0, b) = \emptyset$$

$$\delta_D(\{q_1, q_2\}, a) = \delta_N(q_1, a) \cup \delta_N(q_2, a) = \{q_1, q_2\} \cup \emptyset = \{q_1, q_2\}$$

$$\delta_D(\{q_1, q_2\}, b) = \delta_N(q_1, b) \cup \delta_N(q_2, b) = \emptyset \cup \{q_1, q_3\} = \{q_1, q_3\}$$

...

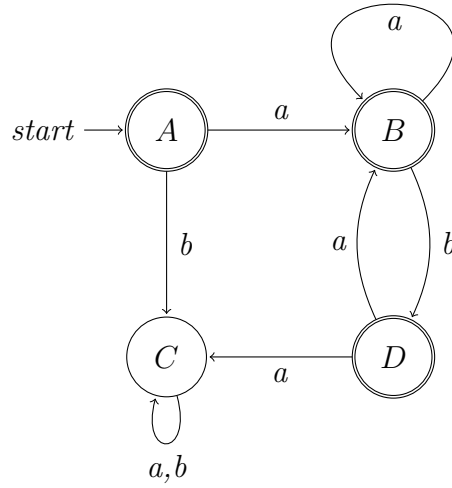
ottengo quindi:

DFA	a	b
$* \rightarrow \{q_0\}$	$\{q_1, q_2\}$	\emptyset
$*\{q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_1, q_3\}$
\emptyset	\emptyset	\emptyset
$*\{q_1, q_3\}$	$\{q_1, q_2\}$	\emptyset

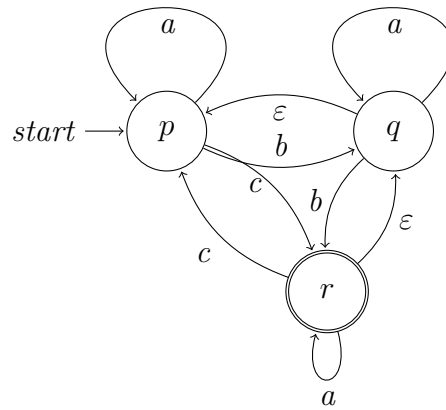
Posso ora rinominare:

- $A = \{q_0\}$
- $B = \{q_1, q_2\}$
- $C = \emptyset$
- $D = \{q_1, q_3\}$

ottengo quindi il seguente DFA:



Esempio 72. Trasformare il seguente ε -NFA in un DFA:



vediamo le $ECLOSE$:

$$ECLOSE(p) = \{p\}$$

$$ECLOSE(q) = \{p, q\}$$

$$ECLOSE(r) = \{p, q, r\}$$

si ottiene quindi:

	a	b	c
$to \{p\}$	$\{p\}$	$\{p, q\}$	$\{p, q, r\}$
$\{p, q\}$	$\{p, q\}$	$\{p, q, r\}$	$\{p, q, r\}$
$*\{p, q, r\}$	$\{p, q, r\}$	$\{p, q, r\}$	$\{p, q, r\}$

infatti, per esempio:

$$\delta_D(\{p\}, a) = ECLOSE(\delta(p, a)) = ECLOSE(\{p\}) = ECLOSE(p) = \{p\}$$

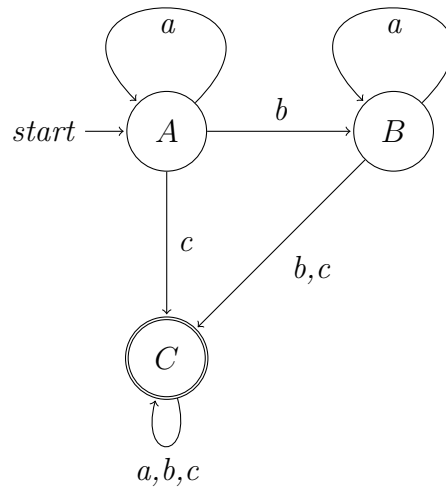
$$\begin{aligned} \delta_D(\{p, q\}, a) &= ECLOSE(\delta_N(p, a) \cup \delta_N(q, a)) = \\ ECLOSE(\{p\} \cup \{q\}) &= ECLOSE(p) \cup ECLOSE(q) = \{p\} \cup \{p, q\} = \{p, q\} \end{aligned}$$

...

si hanno quindi le seguenti rinominazioni:

- $A = \{p\}$
- $b = \{p, q\}$
- $C = \{p, q, r\}$

ovvero:



torniamo a dare bene qualche definizione per δ in un DFA:

$$\delta: Q \times \Sigma^* \rightarrow Q$$

con: $q \in Q, w \in \Sigma^*$ e $\delta(p, w) = q$

- **caso base:** $w = \varepsilon \rightarrow |w| = 0 \rightarrow \delta(q, \varepsilon) = q$
caso passo: $|w| \neq 0 \rightarrow w = ax$ con $a \in \Sigma, x \in \Sigma^*$: $\delta(q, w) = \delta(q, ax) = \delta(\delta(q, a), x)$
- **caso base:** $w = \varepsilon \rightarrow |w| = 0 \rightarrow \delta(q, w) = \delta(q, \varepsilon) = q$
caso passo: $|w| \neq 0 \rightarrow w = xa$ con $a \in \Sigma, x \in \Sigma^*$: $\delta(q, w) = \delta(q, xa) = \delta(\delta(q, x), a)$

in un NFA si ha:

•