Le Avventure di Carpi

Parte 1 : Generalità

Parole e Linguaggi

Alfabeto:

Definizione 1 Chiameremo *alfabeto* un insieme finito non vuoto Σ di simboli. I suoi elementi sono detti *lettere*.

Per esempio, possiamo considerare gli alfabeti

$$\Sigma_0 = \{a,b\}, \quad \Sigma_1 = \{0,1\}, \quad \Sigma_2 = \{a,b,c\},$$

$$\Sigma_3 = \{0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F\}.$$

Parola:

Definizione 2 Ogni sequenza finita di lettere di Σ si dice *parola* sull'alfabeto Σ. L'insieme delle parole sull'alfabeto Σ sarà denotato con Σ^*.

Concatenazione di parole:

Definizione 3 Si considerino le parole $u = a_1a_2\cdots a_k$ e $v = b_1b_2\cdots b_h$ ($k,h \geq 0$, $a_1,a_2,\ldots,a_k,b_1,b_2,\ldots,b_h \in \Sigma$). La *concatenazione* di u e v è la parola

$$uv = a_1a_2\cdots a_kb_1b_2\cdots b_h\,.$$

Potenza di una parola:

Definizione 4 Sia $n \geq 0$. La *potenza n-esima* di una parola w si definisce induttivamente come segue:

$$u^0 = \varepsilon, \quad u^{n+1} = uu^n \;\; n \geq 0.$$

Fattore di una Parola:

Definizione 5 Diremo che una parola v è un *fattore* di una parola w se risulta $w = xvy$ per opportune parole x, y. Nel caso in cui $x = \varepsilon$ (risp., $y = \varepsilon$) il fattore v si dice *prefisso* (risp., *suffisso*) di w. Diremo che v è un fattore *proprio* se $v \neq w$.

Linguaggio formale:

Definizione 6 Ogni sottoinsieme di Σ^* si dice *linguaggio formale* (o, brevemente, *linguaggio*) sull'alfabeto Σ.

Definizione di Grammatica a Struttura di Frase:

Definizione 7 Una *grammatica a struttura di frase* è una quadrupla

$$G = \langle V, \Sigma, P, S \rangle,$$

ove

- V è un alfabeto finito, detto *vocabolario totale*,
- $\Sigma \subseteq V$ è l'alfabeto dei *simboli terminali*,
- $S \in N = V - \Sigma$ è il *simbolo iniziale* o *assioma*,
- P è un insieme finito di espressioni della forma

  $$\alpha \to \beta$$

  con $\alpha \in V^* - \Sigma^*$ e $\beta \in V^*$, detto insieme delle *produzioni*.

Conseguenza Diretta:

Definizione 8 Siano $\alpha, \beta \in V^*$. Diremo che β è una *conseguenza diretta* di α in G se esistono parole $\gamma_1, \gamma_2 \in V^*$ e una produzione $\gamma \to \gamma'$ in P tali che

$$\alpha = \gamma_1 \gamma \gamma_2, \quad \beta = \gamma_1 \gamma' \gamma_2 .$$

In tal caso scriveremo $\alpha \underset{G}{\Longrightarrow} \beta$.

Diremo che β si *deriva* (o è una *conseguenza*) di α in G se risulta $\alpha = \beta$, oppure esistono $n > 0$, $\alpha_0, \alpha_1, \ldots, \alpha_n \in V^*$ tali che

$$\alpha = \alpha_0 \underset{G}{\Longrightarrow} \alpha_1 \underset{G}{\Longrightarrow} \cdots \underset{G}{\Longrightarrow} \alpha_n = \beta .$$

In tal caso scriveremo $\alpha \overset{*}{\underset{G}{\Longrightarrow}} \beta$.

Forme Sentenziali e Linguaggio Generato:

Chiameremo *forme sentenziali* della grammatica G tutte le conseguenze del simbolo iniziale S. L'insieme delle forme sentenziali di G sarà denotato con $S(G)$.

Chiameremo *linguaggio generato* dalla grammatica G il linguaggio costituito dalle forme sentenziali di G che non contengono variabili. Il linguaggio generato dalla grammatica G sarà denotato con $L(G)$.

Si ha quindi

$$S(G) = \{\alpha \in V^* \mid S \overset{*}{\underset{G}{\Longrightarrow}} \alpha\}, \quad L(G) = S(G) \cap \Sigma^* .$$

Nel seguito, nei simboli $\underset{G}{\Longrightarrow}$ e $\overset{*}{\underset{G}{\Longrightarrow}}$, ometteremo l'indice G quando ciò non crea confusione.

Grammatiche Equivalenti:

Definizione 9 Due grammatiche si dicono *equivalenti* se generano lo stesso linguaggio.

Gerarchia di Chomsky

Grammatica di tipo 0:

Al livello 0 della gerarchia si trovano ovviamente le grammatiche a struttura di frase senza alcuna restrizione. I linguaggi generati da tali grammatiche sono i linguaggi *ricorsivamente enumerabili*, detti anche linguaggi di *tipo 0*.

Grammatica di tipo 1 - Contestuale:

Definizione 10 Una grammatica a struttura di frase $G = \langle V, \Sigma, P, S \rangle$ si dice *sensibile al contesto* (o *contestuale*) se tutte le produzioni hanno la forma[1]

$$\alpha_1 X \alpha_2 \to \alpha_1 \beta \alpha_2, \qquad \text{con } X \in N, \quad \alpha_1, \alpha_2, \beta \in V^*, \quad \beta \neq \varepsilon.$$

Un linguaggio generato da una grammatica sensibile al contesto si dice linguaggio *sensibile al contesto* o di *tipo 1*.

Grammatica Monotona:

Una nozione più generale è quella di *grammatica monotòna*: una grammatica si dice *monotòna* se tutte le produzioni hanno la forma

$$\alpha \to \beta \quad \text{con } |\alpha| \leq |\beta|.$$

Una grammatica è monotona se la lunghezza di α è minore o uguale alla lunghezza di β.

Teorema 1 *Un linguaggio è di tipo 1 se e solo se è generato da una grammatica monotòna.*

In breve, per riconoscere se una grammatica è di tipo 1 basta **guardare** se è **monotona**.

Grammatica di tipo 2 - Non Contestuale:

Definizione 11 Una grammatica a struttura di frase $G = \langle V, \Sigma, P, S \rangle$ si dice *non contestuale* (o *libera dal contesto*) se tutte le produzioni hanno la forma

$$X \to \beta, \quad \text{con } X \in N,\ \beta \in V^*.$$

Grammatica di tipo 3 - Lineare Destra

Definizione 12 Una grammatica a struttura di frase $G = \langle V, \Sigma, P, S \rangle$ è *di tipo 3* (o *lineare destra*) se tutte le produzioni hanno le forme

$$X \to aY, \quad X \to a, \qquad X, Y \in N,\ a \in \Sigma.$$

Retroattività delle grammatiche:

Tutte le grammatiche di **tipo 3** sono anche di **tipo 2**, e tutte le grammatiche di tipo 2 sono anche di tipo 1.

Esempi:

Grammatica che **NON** è **Monotona**, quindi **NON** è di **tipo 1**.

$$S \rightarrow N \qquad A \rightarrow N \qquad N \rightarrow a$$
$$S \rightarrow N,AF \qquad A \rightarrow N,A \qquad N \rightarrow b$$
$$,NF \rightarrow \&N \qquad N \rightarrow c$$

Grammatica **Monotona**, quindi è di **tipo 1**.

$$S \rightarrow NVS \qquad VN \rightarrow ,N \qquad N \rightarrow a \qquad U \rightarrow a$$
$$S \rightarrow U \qquad VU \rightarrow \&U \qquad N \rightarrow b \qquad U \rightarrow b$$
$$N \rightarrow c \qquad U \rightarrow c$$

Grammatica di **tipo 2**.

$$S \rightarrow N \qquad M \rightarrow N\&N \qquad N \rightarrow a$$
$$S \rightarrow M \qquad M \rightarrow N,M \qquad N \rightarrow b$$
$$N \rightarrow c$$

Grammatica di **tipo 3**.

$$S \rightarrow a \qquad S \rightarrow aR \qquad R \rightarrow \&N \qquad M \rightarrow aR \qquad N \rightarrow a$$
$$S \rightarrow b \qquad S \rightarrow bR \qquad R \rightarrow ,M \qquad M \rightarrow bR \qquad N \rightarrow b$$
$$S \rightarrow c \qquad S \rightarrow cR \qquad M \rightarrow cR \qquad N \rightarrow c$$

Parte 2 : Linguaggi Regolari

Automi a Stati Finiti

Automa Deterministico:

Definizione 13 Un *automa a stati finiti deterministico* è una quintupla

$$\mathscr{A} = \langle Q, \Sigma, \delta, q_0, F \rangle,$$

dove

- Q è un insieme finito, detto insieme degli *stati*,
- Σ è un alfabeto, detto *alfabeto di input*,
- $\delta : Q \times \Sigma \to Q$ è la *funzione di transizione*,
- $q_0 \in Q$ è lo *stato iniziale*,
- $F \subseteq Q$ è l'insieme degli *stati finali*.

Definizione 14 Sia $\mathscr{A} = \langle Q, \Sigma, \delta, q_0, F \rangle$ un automa a stati finiti deterministico. Una parola $w \in \Sigma^*$ è *accettata* da $\mathscr{A}$ se $\widehat{\delta}(q_0, w) \in F$. L'insieme delle parole accettate da $\mathscr{A}$, cioè il linguaggio

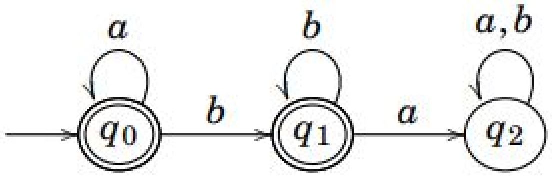
$$L(\mathscr{A}) = \{w \in \Sigma^* \mid \widehat{\delta}(q_0, w) \in F\}$$

si dice *linguaggio riconosciuto* (o *accettato*) da $\mathscr{A}$.

Un **Automa Deterministico** può essere rappresentato tramite un **Grafo Orientato** ed **Etichettato**. Un automa a stati finiti deterministico ha un grafo in cui **per ogni nòdo devono** uscire tante frecce quante sono le lettere dell'alfabeto Σ **una sola volta**.
Gli **Stati Finali** si indicano con un nòdo **cerchiato due volte** e lo **Stato Iniziale** con un nodo con una **freccia entrante**.

Esempio 8 Si consideri l'automa $\mathscr{A} = \langle Q, \Sigma, \delta, q_0, F \rangle$ con $Q = \{q_0, q_1, q_2\}$, $\Sigma = \{a, b\}$, $F = \{q_0, q_1\}$ e con δ definita dalla tabella seguente:

	a	b
q_0	q_0	q_1
q_1	q_2	q_1
q_2	q_2	q_2



Definizione 17 Sia $\mathscr{A} = \langle Q, \Sigma, \delta, q_0, F \rangle$ un automa a stati finiti deterministico. Uno stato $q \in Q$ si dice *accessibile* se $q = \delta(q_0, w)$ per qualche $w \in \Sigma^*$. L'automa si dice *ridotto* se tutti i suoi stati sono accessibili.

Automi Non Deterministici:

Definizione 15 Un *automa a stati finiti non deterministico* è una quintupla

$$\mathscr{A} = \langle Q, \Sigma, \delta, q_0, F \rangle,$$

dove Q, Σ, q_0, F sono come nella Definizione 13 e

$$\delta : Q \times \Sigma \to \mathscr{P}(Q)$$

è la *funzione di transizione*.

$\mathscr{P}(Q)$ è l'insieme delle parti di Q cioè, tutti i possibili sottoinsiemi di Q.

Un **Automa Non Deterministico** può essere rappresentato tramite un **Grafo Orientato** ed **Etichettato** dove da ogni nòdo **possono uscire più frecce** con la **stessa etichetta**. Non è necessario che da ogni nòdo escano tante frecce quante sono le lettere dell'alfabeto.

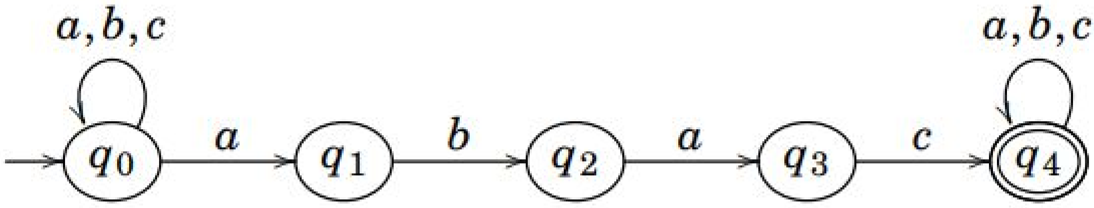
Linguaggio Accettato:

Definizione 16 Sia $\mathscr{A} = \langle Q, \Sigma, \delta, q_0, F \rangle$ un automa a stati finiti non deterministico. Una parola $w = a_1 a_2 \cdots a_n$, $a_1, a_2, \ldots, a_n \in \Sigma$, $n \geq 0$, è *accettata* da $\mathscr{A}$ se esistono stati $q_1, q_2, \ldots, q_n \in Q$ tali che

$$q_i \in \delta(q_{i-1}, a_i),\ 1 \leq i \leq n, \quad q_n \in F. \tag{1}$$

L'insieme delle parole accettate da $\mathscr{A}$ si dice *linguaggio accettato* (o *riconosciuto*) da $\mathscr{A}$ e si denota con $L(\mathscr{A})$.

Esempio Automa Non Deterministico:



Regola dello Scambia Bare:

Teorema 2 *Sia $\mathscr{A} = \langle Q, \Sigma, \delta, q_0, F\rangle$ un automa a stati finiti non deterministico. Esiste effettivamente un automa a stati finiti deterministico $\mathscr{A}'$ tale che*

$$L(\mathscr{A}) = L(\mathscr{A}').$$

DIMOSTRAZIONE:

$\mathscr{A}'$ = <Q', Σ, δ' ,q_0, *F'*>

- l'insieme degli stati è l'insieme $Q' = \mathscr{P}(Q)$ costituito dai sottoinsiemi di Q,
- lo stato iniziale è $s_0 = \{q_0\}$,
- gli stati finali sono tutti i sottoinsiemi di Q che intersecano F, cioè

$$F' = \{s \in \mathscr{P}(Q) \mid s \cap F \neq \emptyset\} \qquad (2)$$

- la funzione di transizione $\delta' : \mathscr{P}(Q) \times \Sigma \to \mathscr{P}(Q)$ è definita da

$$\delta'(s,a) = \bigcup_{q \in s} \delta(q,a), \quad s \in \mathscr{P}(Q),\ a \in \Sigma. \qquad (3)$$

Lemma 1 *Per ogni $w \in \Sigma^*$, $\delta'(s_0, w)$ è uguale all'insieme degli stati che, nel grafo dell'automa $\mathscr{A}$, si raggiungono da q_0 con un cammino etichettato w.*

Automa con ε-transizioni:

Definizione 18 Un *automa a stati finiti non deterministico con ε-transizioni* è una quintupla

$$\mathscr{A} = \langle Q, \Sigma, \delta, q_0, F\rangle,$$

dove Q, Σ, q_0, F sono come nella Definizione 13 e

$$\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to \mathscr{P}(Q)$$

è la *funzione di transizione*.

Linguaggio accettato con ε-transizioni:

Definizione 19 Sia $\mathscr{A} = \langle Q, \Sigma, \delta, q_0, F \rangle$ un automa a stati finiti non deterministico con ε-transizioni. Una parola $w \in \Sigma^*$ è *accettata* da $\mathscr{A}$ se esistono $n \geq 0$, $a_1, a_2, \ldots, a_n \in \Sigma \cup \{\varepsilon\}$ e $q_1, q_2, \ldots, q_n \in Q$ tali che

$$w = a_1 a_2 \cdots a_n, \quad q_i \in \delta(q_{i-1}, a_i),\ 1 \leq i \leq n, \quad q_n \in F. \tag{5}$$

L'insieme delle parole accettate da $\mathscr{A}$ si dice *linguaggio accettato* (o *riconosciuto*) da $\mathscr{A}$ e si denota con $L(\mathscr{A})$.

ε-Chiusura:

Definizione 20 Sia $\mathscr{A} = \langle Q, \Sigma, \delta, q_0, F \rangle$ un automa a stati finiti non deterministico con ε-transizioni. La *ε-chiusura* di uno stato $q \in Q$ è il più piccolo sottoinsieme $C_\varepsilon(q)$ di Q tale che

- $q \in C_\varepsilon(q)$,
- per ogni $p \in C_\varepsilon(q)$, $\delta(p, \varepsilon) \subseteq C_\varepsilon(q)$.

La *ε-chiusura* di un insieme di stati $S \subseteq Q$ è l'unione delle ε-chiusure dei singoli stati di S: $C_\varepsilon(S) = \bigcup_{q \in S} C_\varepsilon(q)$.

Per il calcolo effettivo della ε-chiusura di uno stato, si può osservare che $C_\varepsilon(q)$ è l'insieme degli stati accessibili da q nel grafo delle ε-transizioni dell'automa

$$G_\varepsilon = (Q, \{(r, s) \mid r \in Q,\ s \in \delta(p, \varepsilon)\}.$$

L'ε-chiusura di uno stato q è l'insieme degli stati che si possono raggiungere da q passando solo per ε-transizioni

---

Regola Scambia Bare con ε:

Teorema 3 *Sia $\mathscr{A}$ un automa a stati finiti non deterministico con ε-transizioni. Esiste effettivamente un automa a stati finiti deterministico $\mathscr{A}'$ tale che*

$$L(\mathscr{A}) = L(\mathscr{A}').$$

DIMOSTRAZIONE:

$\mathscr{A}'$ = <Q', Σ, δ' , s_0, *F'*>

$$Q' = \mathscr{P}(Q),$$

$$\delta'(s,a) = C_\varepsilon\left(\bigcup_{p\in s}\delta(p,a)\right), \qquad s \in \mathscr{P}(Q),\ a \in \Sigma$$

$$s_0 = C_\varepsilon(q_0),$$

$$F' = \{s \in \mathscr{P}(Q) \mid s \cap F \neq \emptyset\}.$$

Il risultato di δ' è: l'unione di tutti gli stati raggiungibili con a partendo da p, ai quali poi viene fatta l'ε chiusura.

Espressioni Regolari

Definizione Espressione Regolare:

Definizione 23 Dato un alfabeto Σ, consideriamo l'alfabeto $\widehat{\Sigma}$ ottenuto aggiungendo a Σ i nuovi simboli $\emptyset$, $+$, $*$, $($, $)$. Le *espressioni regolari* sull'alfabeto Σ sono le parole di $\widehat{\Sigma}^*$ definite dalle seguenti regole:

(i) Ogni lettera $a \in \Sigma$ è un'espressione regolare; $\emptyset$ è un'espressione regolare,

(ii) Se E e F sono espressioni regolari, allora $(E+F)$, (EF) e E^* sono espressioni regolari,

(iii) Solo le parole che si ottengono applicando un numero finito di volte le due regole precedenti sono espressioni regolari.

Linguaggio Regolare:

Definizione 24 Per ogni $a \in \Sigma$, l'espressione regolare a *denota* il linguaggio $\{a\}$; l'espressione regolare $\emptyset$ *denota* il linguaggio vuoto.

Date due espressioni regolari E, F e detti rispettivamente L_E e L_F i linguaggi denotati da E e F, le espressioni regolari $(E+F)$, (EF) e E^* *denotano* rispettivamente i linguaggi $L_E \cup L_F$, $L_E L_F$ e L_E^*.

Diremo che un linguaggio è *regolare* se esiste un'espressione regolare che lo denota.

Teorema di Kleene

Dato che i **Linguaggi** sono **insiemi di parole** possiamo operare su di essi con operazioni booleane di **Unione**, **Intersezione** e **Complemento**.

Concatenazione e Potenza:

Definizione 21 Siano L_1 e L_2 due linguaggi sull'alfabeto Σ. La *concatenazione* di L_1 e L_2 è il linguaggio

$$L_1L_2 = \{uv \mid u \in L_1,\ v \in L_2\}.$$

La *potenza n-esima* di un linguaggio L si definisce induttivamente come segue:

$$L^0 = \{\varepsilon\}, \qquad L^{n+1} = LL^n,\ n \geq 0.$$

Chiusura di Kleene:

Definizione 22 La *chiusura di Kleene* di un linguaggio L è il linguaggio

$$L^* = \bigcup_{n\geq 0} L^n = \{u_1u_2\cdots u_n \mid n \geq 0,\ u_1, u_2, \ldots, u_n \in L\}.$$

In breve la chiusura di kleene di un linguaggio sono tutte le possibili combinazioni che si possono fare con le parole del linguaggio.

L = {a, ab, abb} L* = {ε} U {a, ab, abb, aa, aab, aabb, aaa, aaab, ...}

Teorema di Kleene:

Teorema 4 (Kleene) *Un linguaggio è regolare se e soltanto se è riconosciuto da un automa a stati finiti.*

Si presentano quindi 2 problemi:

- **Sintesi**: dato un Linguaggio Regolare → costruire automa a stati finiti che lo riconosce
- **Analisi**: dato un automa a stati finiti → trovare espressione regolare relativa al linguaggio

SINTESI

Osserviamo innanzitutto che i linguaggi denotati dalle espressioni regolari $\emptyset$ e a, $a \in \Sigma$, sono riconosciuti rispettivamente dagli automi

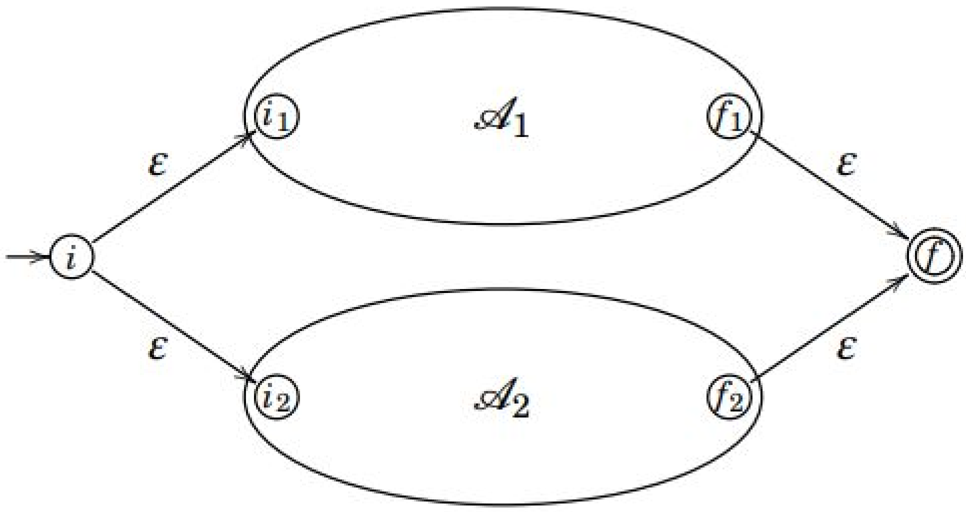
Sintesi Pratica:

Lemma 2 *Siano $\mathscr{A}_1$ e $\mathscr{A}_2$ due automi non deterministici con ε-transizioni provvisti di un unico stato finale e siano rispettivamente L_1 e L_2 i linguaggi accettati da tali automi. È effettivamente possibile costruire degli automi non deterministici con ε-transizioni provvisti di un unico stato finale che riconoscono rispettivamente i linguaggi $L_1 \cup L_2$, L_1L_2, L_1^*.*

L U L’:

$Q = Q_1 \cup Q_2 \cup \{i, f\}$, ove i, f sono due nuovi stati,

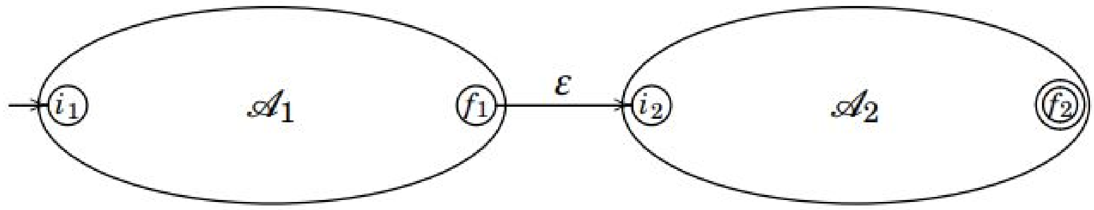
i e f sono, rispettivamente lo stato iniziale e l’unico stato finale dell’automa $\mathscr{A}$.



LL':

$Q = Q_1 \cup Q_2$,

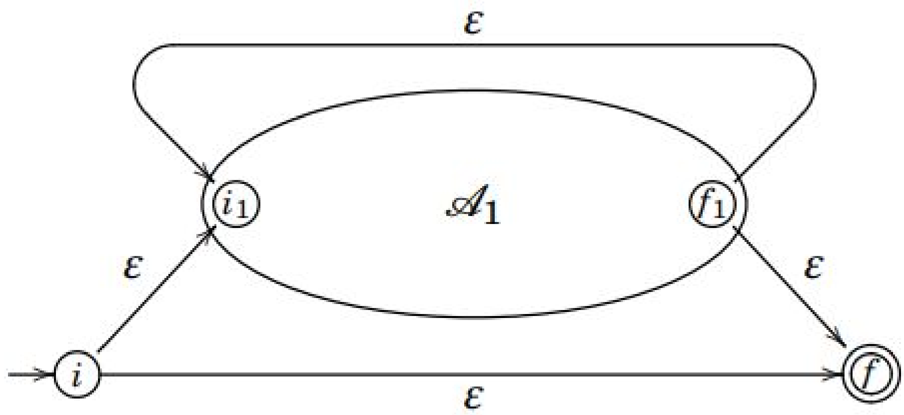
lo stato iniziale di $\mathscr{A}_1$ e lo stato finale di $\mathscr{A}_2$ sono, rispettivamente lo stato iniziale e l'unico stato finale dell'automa $\mathscr{A}'$.



L*:

$Q = Q_1 \cup \{i, f\}$, ove i, f sono due nuovi stati,

i e f sono, rispettivamente lo stato iniziale e l'unico stato finale dell'automa $\mathscr{A}''$.



In breve:

Corollario 1 *Data un'espressione regolare E, si può effettivamente costruire un automa a stati finiti che riconosce il linguaggio denotato da E. Si può inoltre supporre che tale automa abbia un unico stato finale.*

ANALISI

Lemma 3 *Per $i, j = 1, 2, \ldots, n$, $k = 0, 1, \ldots, n$ si può determinare effettivamente un'espressione regolare E_{ijk} che denota l'insieme delle etichette dei cammini nel grafo dell'automa $\mathscr{A}$ con origine in q_i, termine in q_j e che attraversano esclusivamente stati di indice $\leq k$.*

$$E_{ijk} = E_{ijk-1} + (E_{ikk-1})(E_{kkk-1})^*(E_{kjk-1}).$$

Figura 3: E_{ijk}

In breve:

Corollario 2 *Dato un automa (non deterministico con ε-transizioni) $\mathscr{A}$ si può effettivamente calcolare un'espressione regolare E che denota il linguaggio $L(\mathscr{A})$.*

Conseguenze del Teorema di Kleene:

Proposizione 1 *L'unione e l'intersezione di due linguaggi regolari sono linguaggi regolari. Il complemento di un linguaggio regolare è un linguaggio regolare.*

L'**Unione** di due **Linguaggi Regolari** sappiamo già che forma un Linguaggio Regolare (per Sintesi Pratica).

Il **Linguaggio Complementare** si ottiene con

$$\Sigma^* - L$$

e questo è riconosciuto dall'automa a stati finiti $\mathscr{A}' = \langle Q, \Sigma, \delta, q_0, Q - F \rangle$ quindi è un Linguaggio Regolare (per Teorema di Kleene).

L'**Intersezione** di due linguaggi regolari si ottiene con

$$L_1 \cap L_2 = \Sigma^* - ((\Sigma^* - L_1) \cup (\Sigma^* - L_2))$$

Proposizione 2 *Si può decidere effettivamente se due espressioni regolari denotano lo stesso linguaggio.*

Per far ciò basta controllare se il Linguaggio L_E è **incluso** nel Linguaggio L_F e **viceversa**.
Un Linguaggio L_E è incluso in un Linguaggio L_F se e solo se il Linguaggio $L_E \cap (\Sigma^* - L_F)$ è vuoto (L_E intersecato al complementare di L_F è vuoto).

Automa Minimo

Congruenza Destra/Sinistra:

Definizione 25 Sia Σ un alfabeto. Una relazione di equivalenza $\sim$ su Σ^* si dice una *congruenza destra* (risp., *sinistra*) se per ogni $u, v \in \Sigma^*$ tali che $u \sim v$ e per ogni lettera $a \in \Sigma$, si ha $ua \sim va$ (risp., $au \sim av$).

Una relazione che sia contemporaneamente una congruenza destra e una congruenza sinistra si dice *congruenza*.

Relazione di Nerode:

Definizione 26 Sia L un linguaggio sull'alfabeto Σ. L'*equivalenza di Nerode* associata al linguaggio L è la relazione $\mathcal{N}_L$ definita in Σ^* da

$$u \, \mathcal{N}_L \, v \quad \text{se per ogni } y \in \Sigma^*, \; uy \in L \Longleftrightarrow vy \in L .$$

In altri termini, due parole $u, v \in \Sigma^*$ sono nella relazione $\mathcal{N}_L$ se hanno gli stessi 'completamenti' a destra in L.

È evidente che la relazione $\mathcal{N}_L$ è un'equivalenza. Mostriamo ora che si tratta in realtà di una congruenza destra.

Lemma #:

Lemma 4 *Sia $L \subseteq \Sigma^*$ un linguaggio.*

1. *La relazione $\mathcal{N}_L$ è una congruenza destra,*
2. *L è unione di classi di equivalenza di $\mathcal{N}_L$,*

Teorema di Nerode:

Teorema 5 (Myhill-Nerode, 1958) *Sia L un linguaggio sull'alfabeto Σ. Le seguenti proposizioni sono equivalenti:*

(i) *L è regolare,*

(ii) *L è unione di classi di una congruenza destra su Σ^* di indice finito,*

(iii) *$\mathcal{N}_L$ ha indice finito.*

Automa Minimo di Nerode:

Proposizione 3 *Sia L un linguaggio regolare. L'automa di Nerode di L è un automa deterministico che riconosce L col minimo numero di stati possibile.*

Questo è utile per il problema della Minimizzazione: dato un Automa A costruire un Automa A' col minor numero di stati possibili. (guarda pagina 56 sopra Carpi)

Grammatica di Tipo 3

Definizione 27 Una grammatica a struttura di frase $G = \langle V, \Sigma, P, S\rangle$ è *di tipo 3* se tutte le produzioni hanno le forme

$$X \to aY, \quad X \to a, \qquad X, Y \in N,\ a \in \Sigma.$$

Nel caso in cui S non compaia nei lati destri delle produzioni è ammessa anche la produzione $S \to \varepsilon$.

Tipo 3 to Automa e viceversa:

Teorema 6 *Un linguaggio è accettato da un automa a stati finiti se e soltanto se è generato da una grammatica di tipo 3.*

Da Grammatica G = <V,Σ,P,S> ad Automa:
A = <Q,Σ,δ, q0,F>

- Q = N U {ε} (Gli stati sono le variabili della grammatica e la parola vuota).
- δ(X,a) = {Y ∈ N ∪{ε} | X → aY in P}, δ(ε,a) = ∅
- F = {ε} (L'unico stati finale è ε).

Da Automa A = <Q,Σ,δ, q0,F> a Grammatica:

- N = Q
- Le produzioni sono

  $X \to aY,\qquad$ per ogni $X \in Q,\ a \in \Sigma,\ Y \in \delta(X,a)$,

  $X \to a,\qquad$ per ogni $X \in Q,\ a \in \Sigma$ tali che $\delta(X,a) \cap F \neq \emptyset$.
- Il simbolo iniziale S = q_0

Teorema 7 *Un linguaggio è accettato da un automa a stati finiti se e solo se è generato da una grammatica lineare destra.*

Lemma di Iterazione

Lemma di Iterazione:

Teorema 8 (Lemma di iterazione) *Sia L un linguaggio regolare. Esiste un intero n tale che ogni parola $w \in L$ di lunghezza $|w| \geq n$ si può fattorizzare $w = xyz$ con $y \neq \varepsilon$ e $xy^*z \subseteq L$.*

Questo teorema è uno dei vari modi per verificare se un linguaggio è Regolare. Supponendo che un dato linguaggio sia regolare, se non rispetta la forma del teorema precedente, allora non è Regolare.

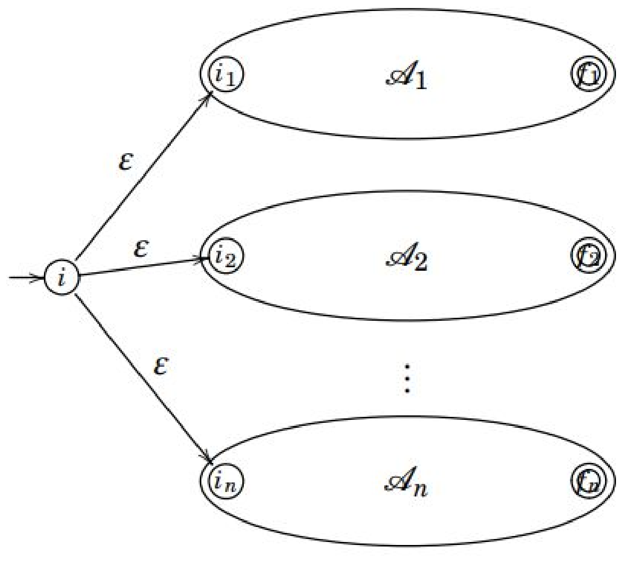
Analisi Lessicale

l'automa $\mathscr{A} = \langle Q, \Sigma, \delta, q_0, F \rangle$ con

$$Q = \{i_0\} \cup \bigcup_{j=0}^{n} Q_j, \quad F = \{f_j \mid 1 \leq j \leq n\},$$

$$\delta(q,a) = \begin{cases} \delta_j(q,a), & \text{se } q \in Q_j, 1 \leq j \leq n, a \in \Sigma \cup \{\varepsilon\}, \\ \{i_1, \ldots, i_n\}, & \text{se } q = i_0,\ a = \varepsilon, \\ \emptyset, & \text{se } q = i_0,\ a \in \Sigma. \end{cases}$$

In altri termini, l'automa $\mathscr{A}$ ha gli stati, le transizioni e gli stati finali di $\mathscr{A}_1, \mathscr{A}_2, \ldots, \mathscr{A}_n$ e inoltre un nuovo stato iniziale i_0 e delle ε-transizioni da i_0 agli stati iniziali originali degli automi $\mathscr{A}_1, \mathscr{A}_2, \ldots, \mathscr{A}_n$.



Per un esempio pratico guardare pagine 69 sopra carpi.

Parte III : Linguaggi non contestuali

Grammatiche di Tipo 2 - Non Contestuali

Definizione:

Definizione 11 Una grammatica a struttura di frase $G = \langle V, \Sigma, P, S \rangle$ si dice *non contestuale* (o *libera dal contesto*) se tutte le produzioni hanno la forma

$$X \to \beta, \quad \text{con } X \in N,\ \beta \in V^*.$$

Palindrome:

Esempio 33 Ricordiamo che una parola $w \in \Sigma$ si dice palindroma se si ha

$$w = a_1 a_2 \cdots a_n = a_n a_{n-1} \cdots a_1$$

per opportuni $n \geq 0$, $a_1, a_2, \ldots, a_n \in \Sigma$. Sia P il linguaggio delle palindrome sull'alfabeto $\Sigma = \{a, b\}$. Osserviamo che:

1. ε, a, b sono palindrome,
2. per ogni palindroma w, le parole awa e bwb sono palindrome,
3. ogni palindroma su Σ si ottiene applicando un numero finito di volte le regole 1. e 2.

Ne segue che il linguaggio P 'soddisfa' l'identità

$$P = \varepsilon + a + b + aPa + bPb \tag{12}$$

Inclusione di Classe:

Proposizione 4 *La classe dei linguaggi regolari è strettamente inclusa in quella dei linguaggi non-contestuali.*

Alberi di Derivazione

Albero di Derivazione:

Definizione 28 Sia $G = \langle V, \Sigma, P, S \rangle$ una grammatica non contestuale e T un albero. Diremo che T è un albero di derivazione della grammatica G se verifica le seguenti condizioni:

1. le etichette dei nodi interni sono elementi di N,
2. le etichette delle foglie sono elementi di $\Sigma \cup \{\varepsilon\}$,
3. le foglie con etichetta ε sono 'figli unici',
4. se un nodo con etichetta X ha figli etichettati nell'ordine $\alpha_1, \ldots, \alpha_k$, allora $X \to \alpha_1 \cdots \alpha_k$ è una produzione di G.

La parola che si ottiene leggendo, nell'ordine, le etichette delle foglie è la *parola associata* a T.

Alberi e Derivazioni:

Proposizione 5 *Sia $G = \langle V, \Sigma, P, S \rangle$ una grammatica non contestuale e siano $A \in N$ e $w \in \Sigma^*$. Si ha $A \overset{*}{\Longrightarrow} w$ se e solo se esiste un albero di derivazione di G associato a w con la radice etichettata A.*

In particolare, w appartiene a $L(G)$ se e solo se esiste un albero di derivazione di G associato a w con la radice etichettata S.

Grammatica Non Ambigua:

Definizione 29 Una grammatica si dice *non ambigua* se a ogni parola del linguaggio generato corrisponde un'unico albero di derivazione.

Grammatica Inerentemente Ambigua:

Definizione 30 Un linguaggio non contestuale si dice *inerentemente ambiguo* se non è generato da nessuna grammatica non contestuale non ambigua.

In modo più semplice, una Grammatica è **Inerentemente Ambigua** se è **generata solo** da Grammatiche di **tipo 2 Ambigue**.

Semplificazioni

ε produzioni:

Definizione 31 Sia $G = \langle V, \Sigma, P, S\rangle$ una grammatica non contestuale. Le produzioni della forma $X \to \varepsilon$ si dicono ε-produzioni. Le produzioni della forma $X \to Y$, $X, Y \in N$ si dicono produzioni 1-arie.

Grammatica 2 senza ε produzioni (1):

Proposizione 6 *Ogni linguaggio non contestuale è generato da una grammatica non contestuale priva di ε-produzioni, eccettuata, eventualmente, la produzione $S \to \varepsilon$, ove S è il simbolo iniziale. Inoltre, si può assumere che il simbolo iniziale non compaia nei lati destri delle produzioni.*

Grammatica 2 senza ε produzioni (2):

Proposizione 7 *Ogni linguaggio non contestuale è generato da una grammatica non contestuale priva di produzioni 1-arie e di ε-produzioni, eccettuata, eventualmente, la produzione $S \to \varepsilon$, ove S è il simbolo iniziale. Inoltre, si può assumere che il simbolo iniziale non compaia nei lati destri delle produzioni.*

Variabili improduttive ed inaccessibili:

Definizione 32 Sia $G = \langle V, \Sigma, P, S\rangle$ una grammatica non contestuale. Una variabile $X \in N = V - \Sigma$ si dice *produttiva* (o *utile*) se esiste $w \in \Sigma^*$ tale che $X \stackrel{*}{\Longrightarrow} w$. In caso contrario, X si dice *improduttiva* (o *inutile*).

Una variabile $X \in N$ si dice *accessibile* se esistono $\alpha, \beta \in V^*$ tali che $S \stackrel{*}{\Longrightarrow} \alpha X \beta$. In caso contrario, X si dice *inaccessibile*.

Grammatica Ridotta:

Definizione 33 Una grammatica non contestuale $G = \langle V, \Sigma, P, S\rangle$ si dice *ridotta* se tutte le variabili $X \neq S$ sono sia produttive che accessibili.

Il simbolo iniziale S **è sempre Accessibile** ed è sempre **Produttiva** a meno che il linguaggio generato da G sia vuoto.

Grammatica Ridotta in pratica:

Proposizione 8 *Ogni linguaggio non contestuale è generato da una grammatica non contestuale ridotta.*

Per determinare una Grammatica Ridotta data una Grammatica G si deve:
- Determinare le variabili produttive
- Eliminare le variabili improduttive e le produzioni che contengono tali variabili
- Determinare le variabili accessibili della grammatica ottenuta
- Eliminare le variabili inaccessibili e le produzioni che contengono tali variabili

L'algoritmo **va eseguito** in questo preciso ordine in quanto l'eliminazione di variabili improduttive potrebbe creare delle nuove variabili inaccessibili.

La forma normale (finale) di Chomsky:

Definizione 34 Una grammatica non contestuale $G = \langle V, \Sigma, P, S \rangle$ si dice in *forma normale di Chomsky* se ha solo produzioni dei tipi

- $X \to YZ$, con $X, Y, Z \in N$,
- $X \to a$, con $X \in N$, $a \in \Sigma$,
- $S \to \varepsilon$, ma solo a condizione che S non compaia nei lati destri delle produzioni.

Teorema 9 *Ogni linguaggio non contestuale è generato da una grammatica non contestuale in forma normale di Chomsky.*

La forma normale di Greibach:

Definizione 35 Una grammatica non contestuale G si dice in *forma normale di Greibach* se ha solo produzioni dei tipi

- $X \to a\gamma$, con $a \in \Sigma$ e $\gamma \in N^*$,
- $S \to \varepsilon$, ma solo a condizione che S non compaia nei lati destri delle produzioni.

La caratteristica di una grammatica in forma normale di Greibach G è che ogni derivazione di una parola $w \in L(G)$ è costituita esattamente di $|w|$ passi.

Teorema 10 *Ogni linguaggio non contestuale è generato da una grammatica non contestuale in forma normale di Greibach.*

Come anche per i precedenti ci assicura che se abbiamo un linguaggio di tipo 2 allora possiamo ricavare una grammatica in forma normale di Greibach (Chomsky e ridotta).

Lemma di Iterazione:

Lemma 5 (di Iterazione) *Sia L un linguaggio non contestuale. Esiste un intero n tale che ogni parola $w \in L$ di lunghezza $|w| > n$ si può fattorizzare $w = xuyvz$ con $uv \neq \varepsilon$ e $xu^k yv^k z \in L$ per ogni $k \geq 0$.*

Questo lemma è utile per verificare se un linguaggio non è di tipo 2.

Problemi di Ricognizione e Parsing:

Sia $G = \langle V, \Sigma, P, S \rangle$ una grammatica non-contestuale. Vogliamo considerare i due seguenti problemi:

1. *Ricognizione*: Data una parola $w \in \Sigma^*$, decidere se $w \in L(G)$.
2. *Parsing*: Data una parola $w \in L(G)$ costruire un albero di derivazione di G associato a w.

Algoritmo di Cocke-Kasami-Younger:

Questo algoritmo risolve il problema della Ricognizione e del Parsing.
Si applica alle grammatiche in forma normale di Chomsky.
L'algoritmo costruisce una tabella in $O(n^3)$ e dice se la parola ω passata appartiene a L(G).

La voce Xij della tabella è l'insieme delle variabili A tali che $A \overset{*}{\Rightarrow} a_i a_{i+1} \cdots a_j$.
In particolare ci interessa sapere se S è nell'insieme X_{1n}. Se S appartiene a questo insieme allora w appartiene a L(G). X_{1n} è l'apice della tabella (in questo caso X_{15}.)

X_{15}				
X_{14}	X_{25}			
X_{13}	X_{24}	X_{35}		
X_{12}	X_{23}	X_{34}	X_{45}	
X_{11}	X_{22}	X_{33}	X_{44}	X_{55}
a_1	a_2	a_3	a_4	a_5

Parsing Top Down:

Il problema del parsing può essere risolto anche per grammatiche che non rispettano la forma normale di Chomsky. Quindi non si può utilizzare CKY ma ci sono 2 algoritmi diversi:

- Parsing Top - Down: si parte dall'assioma S e si va a cercare un cammino dell'albero delle produzioni che vada a dare come risultato la parola w.
- Parsing Bottom - Up: si parte dalla parola w e si va a cercare un cammino che ci riporti all'assioma S

Nel Top - Down la parola in input viene divisa in 2 parti:

- Analisi: la parte già analizzata
- Predizione: la parte da analizzare

input	aa	bb
	aa	BB
	analisi	predizione

Procedura per Parsing Top - Down:

1. sostituiamo la variabile più a sinistra della predizione con il lato sinistro di una sua produzione;
2. verifichiamo che i terminali eventualmente presenti all'inizio della predizione ottenuta siano un fattore iniziale dell'input non ancora analizzato;
3. in caso positivo spostiamo tali terminali nell'input analizzato; in caso negativo l'esecuzione fallisce.

Questa procedura non è deterministica.

Automi a Pila:

Definizione 36 Un *automa a pila* è una settupla

$$\mathscr{A} = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F \rangle$$

dove

- Q è un insieme finito, detto *insieme degli stati*,
- Σ è un alfabeto, detto *alfabeto di input*,
- Γ è un alfabeto, detto *alfabeto di pila*,
- $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to \mathscr{P}_F(Q \times \Gamma^*)$ è la *funzione di transizione*,
- $q_0 \in Q$ è lo *stato iniziale*,
- $Z_0 \in \Gamma$ è il *simbolo iniziale della pila*,
- $F \subseteq Q$ è l'insieme degli *stati finali*.

L'automa si trova inizialmente nello stato iniziale e la pila contiene il simbolo iniziale della pila. Supponiamo che a un dato istante della computazione l'automa si trova nello stato p, riceve a in input e Z dall'operazione di pop. Se $\delta(p,a,Z)$ contiene una coppia (q,γ), allora l'automa si può portare nello stato q eseguendo il push di γ sulla pila.

Descrizione Istantanea (conseguenza diretta):

Definizione 37 Una *descrizione istantanea* è una tripla $(q,\, u, \gamma) \in Q \times \Sigma^* \times \Gamma^*$. Nell'insieme delle descrizioni istantanee si introduce la relazione $\underset{\mathscr{A}}{\vdash}$ definita, da

$$(q, aw, Z\alpha) \underset{\mathscr{A}}{\vdash} (p, w, \gamma\alpha) \qquad \text{se} \qquad (p,\gamma) \in \delta(q,a,Z),$$

$q, p \in Q,\ a \in \Sigma,\ w \in \Sigma^*,\ Z \in \gamma,\ \alpha, \gamma \in \Gamma^*$

In altri termini, avremo $D \underset{\mathscr{A}}{\vdash} D'$ se D e D' sono due possibili descrizioni istantanee consecutive di $\mathscr{A}$ in una computazione. Scriveremo poi $D \overset{*}{\underset{\mathscr{A}}{\vdash}} D'$ se esiste una sequenza finita di descrizioni istantanee $D_0, D_1, \ldots, D_n$ tale che

$$D = D_0 \underset{\mathscr{A}}{\vdash} D_1 \underset{\mathscr{A}}{\vdash} \cdots \underset{\mathscr{A}}{\vdash} D_n = D'.$$

Si avrà quindi $D \overset{*}{\underset{\mathscr{A}}{\vdash}} D'$ se esiste una computazione che porta l'automa $\mathscr{A}$ dalla descrizione D alla descrizione D'.

Parola Accettata da Automa a Pila:

Definizione 38 Sia $\mathscr{A} = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F\rangle$ un automa a pila e $w \in \Sigma^*$. Diremo che una parola $w \in \Sigma^*$ è *accettata per stato finale* dall'automa a pila $\mathscr{A}$ se si ha

$$(q_0, w, Z_0) \vdash_{\mathscr{A}}^{*} (f, \varepsilon, \gamma), \quad f \in F,\ \gamma \in \Gamma^*.$$

Invece diremo che w è *accettata per pila vuota* da $\mathscr{A}$ se si ha

$$(q_0, w, Z_0) \vdash_{\mathscr{A}}^{*} (p, \varepsilon, \varepsilon), \quad p \in Q.$$

Infine diremo che w è *accettata per stato finale e pila vuota* da $\mathscr{A}$ se se si ha

$$(q_0, w, Z_0) \vdash_{\mathscr{A}}^{*} (f, \varepsilon, \varepsilon), \quad f \in F.$$

L'insieme delle parole accettate per stato finale (risp., per pila vuota, per stato finale e pila vuota) sarà chiamato il *linguaggio riconosciuto da $\mathscr{A}$ per stato finale* (risp., *per pila vuota, per stato finale e pila vuota*) e sarà denotato con $L_F(\mathscr{A})$ (risp., $L_P(\mathscr{A})$, $L(\mathscr{A})$).

Automa a Pila Deterministico:

Definizione 39 Un automa a pila $\mathscr{A} = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F\rangle$ è *deterministico* se per ogni $q \in Q$, $a \in \Sigma$, $Z \in \Gamma$ l'insieme $\delta(q, a, Z) \cup \delta(q, \varepsilon, Z)$ contiene al più un elemento.

Un linguaggio accettato per stato finale da un automa a pila deterministico si dirà *deterministico*.

Da Stato Finale e Pila Vuota:

Proposizione 9 *Sia* $\mathscr{A} = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F\rangle$ *un automa a pila. Esiste effettivamente un automa a pila* $\mathscr{A}'$ *tale che*

$$L(\mathscr{A}) = L_P(\mathscr{A}') = L_F(\mathscr{A}') = L(\mathscr{A}').$$

Formalmente, $\mathscr{A}' = \langle Q \cup \{q_0', f\}, \Sigma, \Gamma \cup \{\#\}, \delta', q_0', \#, \{f\}\rangle$ ove la funzione di transizione δ' è definita da:

$$\delta'(q,a,Z) = \delta(q,a,Z), \quad \text{per } (q,a,Z) \in Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma,$$

$$\delta'(q,\varepsilon,\#) = \begin{cases} (q_0, Z_0\#) & \text{se } q = q_0', \\ (f,\varepsilon) & \text{se } q \in F, \\ \emptyset & \text{se } q \in Q - F, \end{cases}$$

e $\delta(q,a,Z) = \emptyset$ in tutti i casi rimanenti.

Dalla costruzione risulta chiaro che una computazione di $\mathscr{A}'$ termina con la pila vuota se e soltanto se raggiunge lo stato finale f. Questo implica che

$$L_P(\mathscr{A}') = L_F(\mathscr{A}') = L(\mathscr{A}').$$

Da Pila Vuota:

Proposizione 10 *Sia* $\mathscr{A} = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F\rangle$ *un automa a pila. Esiste effettivamente un automa a pila* $\mathscr{A}'$ *tale che*

$$L_P(\mathscr{A}) = L_P(\mathscr{A}') = L_F(\mathscr{A}') = L(\mathscr{A}').$$

Si ottiene semplicemente con $\mathscr{A}'$ = <Q, $\boldsymbol{\Sigma}$, $\boldsymbol{\Gamma}$, $\boldsymbol{\delta}$, Z_0, Q>.

Linguaggio Deterministico:

Proposizione 11 *Un linguaggio riconosciuto per pila vuota (o per pila vuota e stato finale) da un automa a pila deterministico è un linguaggio deterministico.*

Da Stato Finale:

Proposizione 12 *Sia $\mathscr{A} = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F\rangle$ un automa a pila. Esiste effettivamente un automa a pila $\mathscr{A}'$ tale che*

$$L_F(\mathscr{A}) = L_P(\mathscr{A}') = L_F(\mathscr{A}') = L(\mathscr{A}').$$

Formalmente, definiamo $\mathscr{A}'' = \langle Q \cup \{f\}, \Sigma, \Gamma, \delta', q_0, Z_0, F \cup \{f\}\rangle$ ove la funzione di transizione δ' è data da:

$$\delta'(q,a,Z) = \begin{cases} \delta(q,a,Z), & \text{se } q \in Q, \\ \emptyset, & \text{se } q = f, \end{cases}$$

$$\delta'(q,\varepsilon,Z) = \begin{cases} \delta(q,\varepsilon,Z) \cup (f,Z), & \text{se } q \in F, \\ \delta(q,\varepsilon,Z), & \text{se } q \in Q - F, \\ (f,\varepsilon), & \text{se } q = f \end{cases}$$

($a \in \Sigma$, $Z \in \Gamma$).

Famiglie tutte uguali:

Teorema 11 *La famiglia dei linguaggi accettati da automi a pila per pila vuota, la famiglia dei linguaggi accettati da automi a pila per stato finale e la famiglia dei linguaggi accettati da automi a pila per stato finale e pila vuota coincidono.*

Teorema di Caratterizzazione

Derivazione Sinistra:

Definizione 40 Sia data una grammatica non contestuale $G = \langle V, \Sigma, P, S \rangle$. Siano $\alpha, \beta \in V^*$. Scriveremo $\alpha \underset{L}{\Longrightarrow} \beta$ se esistono $u \in \Sigma^*$, $\sigma \in V^*$ e una produzione $X \to \gamma$ di G tali che

$$\alpha = uX\sigma, \quad \beta = u\gamma\sigma.$$

Una *derivazione sinistra* è una sequenza

$$\alpha = \alpha_0 \underset{L}{\Longrightarrow} \alpha_1 \underset{L}{\Longrightarrow} \cdots \underset{L}{\Longrightarrow} \alpha_n = \beta.$$

In altri termini una derivazione sinistra è una derivazione in cui si riscrive sempre la variabile più a sinistra. Dato che in una grammatica non contestuale l'ordine in cui vengono eseguite le derivazioni descritte da un albero di derivazione è inessenziale, possiamo enunciare il seguente

Lemma 6 *Data una grammatica non contestuale G, ogni parola di $L(G)$ si ottiene dall'assioma S con una derivazione sinistra.*

Uguaglianza Per Pila Vuota e Grammatica:

Proposizione 13 *Sia $\mathscr{A}$ un automa a pila con un solo stato. Esiste effettivamente una grammatica non contestuale G tale che*

$$L_P(\mathscr{A}) = L(G).$$

Sia $\mathscr{A} = \langle \{q\}, \Sigma, \Gamma, \delta, q, Z_0, \{q\} \rangle$ un automa a pila con un unico stato. Possiamo supporre, senza perdita di generalità, $\Gamma \cap \Sigma = \emptyset$. Definiamo la grammatica G prendendo

- L'alfabeto di pila Γ come insieme delle variabili,
- L'alfabeto di input Σ come insieme dei simboli terminali,
- Il simbolo iniziale della pila Z_0 come assioma,
- Le produzioni

$$Z \to a\gamma, \quad \text{con } (q, \gamma) \in \delta(q, a, Z),\ a \in \Sigma \cup \{\varepsilon\},\ Z \in \Gamma.$$

Uguaglianza tra Grammatica 2 e Pila Vuota:

Proposizione 14 *Sia G una grammatica non contestuale. Esiste effettivamente un automa a pila $\mathscr{A}$ con un solo stato tale che*

$$L_P(\mathscr{A}) = L(G).$$

DIMOSTRAZIONE: Senza perdita di generalità, possiamo supporre che tutte le produzioni di G abbiano la forma

$$Z \to a\gamma, \qquad Z \in \Gamma,\ a \in \Sigma \cup \{\varepsilon\},\ \gamma \in \Gamma^*.$$

Basta osservare, per esempio, che le produzioni delle grammatiche in forma normale di Chomsky hanno questa forma e applicare il Teorema 9. A questo punto possiamo definire l'automa a pila con un unico stato $\mathscr{A} = \langle \{q\}, \Sigma, \Gamma, \delta, q, Z_0, \{q\} \rangle$ come segue:

- L'alfabeto di input Σ è l'insieme dei simboli terminali di G,
- L'alfabeto di pila Γ è l'insieme delle variabili di G,
- Il simbolo iniziale della pila Z_0 è l'assioma di G,
- La funzione di transizione è definita da

$$\delta(q, a, Z) = \{(q, \gamma) \mid Z \to a\gamma \text{ in } P\}.$$

La Proposizione 13 ci assicura che $L_P(\mathscr{A})$ è generato da una grammatica non contestuale. Ma, ripercorrendo la costruzione di tale grammatica, ci si convince facilmente che essa è esattamente la nostra grammatica G. Ciò dimostra l'asserto. $\square$

Uguaglianza Pila Vuota e Pila Vuota con 1 stato finale:

Proposizione 15 *Sia $\mathscr{A} = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F\rangle$ un automa a pila. Esiste effettivamente un automa a pila $\mathscr{A}'$ a un solo stato tale che*

$$L_P(\mathscr{A}) = L_P(\mathscr{A}').$$

DIMOSTRAZIONE: L'idea della dimostrazione è di simulare l'automa dato con un automa a un solo stato che registra nella pila anche lo stato dell'automa simulato. Più precisamente definiamo l'automa a pila

$$\mathscr{A}' = \langle \{q\}, \Sigma, \Gamma', \delta', q, Z_0', \{q\}\rangle$$

con

- alfabeto di pila $\Gamma' = Q \times \Gamma \times Q$,
- simbolo iniziale della pila $Z_0' = (q_0, Z_0, q_0)$,
- funzione di transizione δ' definita come segue: se $(p, \varepsilon) \notin \delta(s, a, Z)$ allora

  $$\begin{aligned}\delta'(a,(s,Z,p)) = \{&(t, Z_k, p_{k-1})(p_{k-1}, Z_k, p_{k-2}) \cdots (p_1, Z_1, p) \mid \\ &(t, Z_k \cdots Z_1) \in \delta(s,a,Z),\ p_1,\dots,p_{k-1} \in Q\},\end{aligned}$$

  se invece $(p, \varepsilon) \in \delta(s, a, Z)$ allora

  $$\begin{aligned}\delta'(a,(s,Z,p)) = \{&(t, Z_k, p_{k-1})(p_{k-1}, Z_k, p_{k-2}) \cdots (p_1, Z_1, p) \mid \\ &(t, Z_k \cdots Z_1) \in \delta(s,a,Z),\ p_1,\dots,p_{k-1} \in Q\} \cup \{\varepsilon\},\end{aligned}$$

  $s, p \in Q,\ a \in \Sigma \cup \{\varepsilon\},\ Z \in \Gamma.$

Automa a Pila e Linguaggio 2:

Teorema 12 *Un linguaggio è non contestuale se e solo se è riconosciuto da un automa a pila.*

Parsing Top - Down Deterministico

First(A):

Definiamo la funzione FIRST : $V^+ \to \mathscr{P}(\Sigma)$ ponendo

$$\mathrm{FIRST}(\alpha) = \{a \in \Sigma \mid \alpha \stackrel{*}{\Longrightarrow} a\beta,\ \beta \in V^*\}$$

Detto in altri termini, FIRST(α) è l'insieme dei terminali che possono comparire come prima lettera di una conseguenza di α.

Classe LL(1):

$\gamma \in$ V*

Definizione 41 Una grammatica non-contestuale G priva di ε-produzioni si dice di classe $LL(1)$ se per ogni coppia di produzioni distinte $X \to \alpha \mid \gamma$ si ha $\mathrm{FIRST}(\alpha) \cap \mathrm{FIRST}(\gamma) = \emptyset$.

In altri termini, una grammatica $LL(1)$ è una grammatica la cui tabella di parsing contiene al più una produzione in ogni ingresso. Questo assicura che il parsing può essere effettuato in maniera deterministica.

Il termine $LL(1)$ indica che è possibile la costruzione efficiente di una derivazione sinistra (Left-most) esaminando l'input da sinistra (Left) a destra e esaminando solo 1 lettera dell'input non ancora presente nell'albero di derivazione.

Come calcolare i FIRST:

1. se la prima lettera di α è un simbolo terminale b, allora $\mathrm{FIRST}(\alpha) = b$;
2. se la prima lettera di α è una variabile X, allora per ogni produzione $X \to \beta$, i terminali di FIRST(β) sono anche terminali di FIRST(α),
3. Tutti gli elementi di FIRST(α) con α lato destro di qualche produzione, si trovano applicando un numero finito di volte le regole precedenti.

Le regole precedenti ci forniscono una procedura effettiva per il calcolo di FIRST(α) per tutti gli α che sono lati destri delle produzioni. Inizialmente, porremo $\mathrm{FIRST}(\alpha) = \emptyset$ per tutti gli α. Poi ricalcoliamo tali insiemi usando le regole 1–2 e i valori attuali degli insiemi FIRST(β). Iteriamo quest'ultimo passo, arrestandoci solo quando nessuno degli insiemi FIRST(α) subisce modifiche.

Follow(A):

Consideriamo ora il caso in cui la nostra grammatica abbia anche delle ε-produzioni. In tal caso, definiamo le funzioni FIRST : $V^* \to \mathscr{P}(\Sigma \cup \{\varepsilon\})$ e FOLLOW : $N \to \mathscr{P}(\Sigma)$ ponendo FIRST$(\varepsilon) = \{\varepsilon\}$ e

$$\begin{aligned} \text{FIRST}(\alpha) &= \{a \in \Sigma \mid \alpha \stackrel{*}{\Longrightarrow} a\beta,\ \beta \in V^*\}, && \alpha \in V^*,\ \alpha \stackrel{*}{\not\Longrightarrow} \varepsilon, \\ \text{FIRST}(\alpha) &= \{a \in \Sigma \mid \alpha \stackrel{*}{\Longrightarrow} a\beta,\ \beta \in V^*\} \cup \{\varepsilon\}, && \alpha \in V^*,\ \alpha \stackrel{*}{\Longrightarrow} \varepsilon, \\ \text{FOLLOW}(X) &= \{a \in \Sigma \mid S\# \stackrel{*}{\Longrightarrow} \beta X a \gamma,\ \beta, \gamma \in V^*\}, && X \in N. \end{aligned}$$

Detto in altri termini, FIRST(α) è l'insieme dei terminali che possono comparire come prima lettera di una conseguenza di α, aumentato della parola ε nel caso in cui α sia annullabile. Invece, FOLLOW(X) è l'insieme dei terminali che possono seguire la variabile X in una forma sentenziale.

Come calcolare i Follow(A):

Per il calcolo di FOLLOW iniziamo con le seguenti osservazioni:

1. L'insieme FOLLOW(S) contiene #.
2. Se c'è una produzione $X \to \alpha Y \beta$ con $X, Y \in N$, $\alpha, \beta \in V^*$, allora tutti i terminali contenuti in FIRST(β) sono contenuti anche in FOLLOW(Y).
3. Se inoltre $\varepsilon \in$ FIRST(β), cioè $\beta \stackrel{*}{\Longrightarrow} \varepsilon$, allora tutti i terminali contenuti in FOLLOW(X) sono contenuti anche in FOLLOW(Y).
4. Tutti gli elementi di FOLLOW(Y), $Y \in N$, si trovano applicando un numero finito di volte le regole precedenti.

LL(1) con ε produzioni:

Definizione 42 Una grammatica non-contestuale G si dice di classe $LL(1)$ se per ogni coppia di produzioni distinte $X \to \alpha$, $X \to \gamma$ sono soddisfatte le seguenti condizioni:

1. $\text{FIRST}(\alpha) \cap \text{FIRST}(\gamma) = \emptyset$.
2. Se $\alpha \stackrel{*}{\Longrightarrow} \varepsilon$, allora $\gamma \stackrel{*}{\not\Longrightarrow} \varepsilon$ e $\text{FOLLOW}(X) \cap \text{FIRST}(\gamma) = \emptyset$.

Per poter costruire la tabella di parsing nel caso in cui siano presenti anche delle ε-transizioni, dobbiamo calcolare le funzioni FIRST and FOLLOW in questo caso più generale. Iniziamo con le seguenti osservazioni:

1. se la prima lettera di α è un simbolo terminale b, allora $\text{FIRST}(\alpha) = b$;
2. se la prima lettera di α è una variabile X, allora per ogni produzione $X \to \beta$, i terminali di $\text{FIRST}(\beta)$ sono anche terminali di $\text{FIRST}(\alpha)$,
3. se la prima lettera di α è una variabile annullabile, allora le due condizioni precedenti sono verificate anche dalla seconda lettera di α; se quest'ultima è una variabile annullabile, allora anche dalla terza, e così via;
4. se tutte le lettere di α sono variabili annullabili, allora $\varepsilon \in \text{FIRST}(\alpha)$;
5. Tutti gli elementi di $\text{FIRST}(\alpha)$ con α lato destro di qualche produzione, si trovano applicando un numero finito di volte le regole precedenti.

Le regole precedenti ci forniscono quindi una procedura effettiva per il calcolo di $\text{FIRST}(\alpha)$ per tutti gli α che sono lati destri delle produzioni. Inizialmente, porremo $\text{FIRST}(\alpha) = \emptyset$ per tutti gli α. Poi ricalcoliamo tali insiemi usando le regole 1–4 e i valori attuali degli insiemi $\text{FIRST}(\beta)$. Iteriamo quest'ultimo passo, arrestandoci solo quando nessuno degli insiemi $\text{FIRST}(\alpha)$ subisce modifiche.

Propietà di chiusura

Proposizione 16 *La classe dei linguaggi non contestuali è chiusa per unione, concatenazione e chiusura di Kleene.*

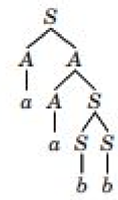
Proposizione 17 *L'intersezione di un linguaggio non contestuale con un linguaggio regolare è un linguaggio non contestuale.*

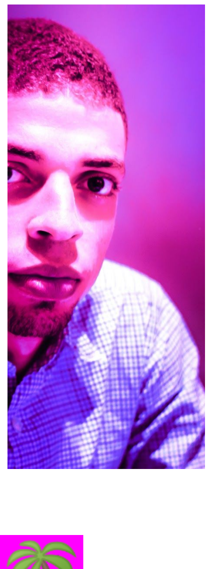
Proposizione 18 *La classe dei linguaggi non contestuali è chiusa per omomorfismi.*

Teorema di Chomsky Schutzenberger:

Teorema 13 (Chomsky, Schützenberger) *Un linguaggio L è non contestuale se e soltanto se esistono un intero $k > 0$, un linguaggio regolare R e un morfismo f tali che*

$$L = f(D_k \cap R).$$





🌴La poesia d🌅lla pagina 93🌴