Sommario

1. **Gerarchia di Chomsky**
2. **Forma normale di Chomsky**
3. **Algoritmi per l'appartenenza**

Gerarchia di Chomsky

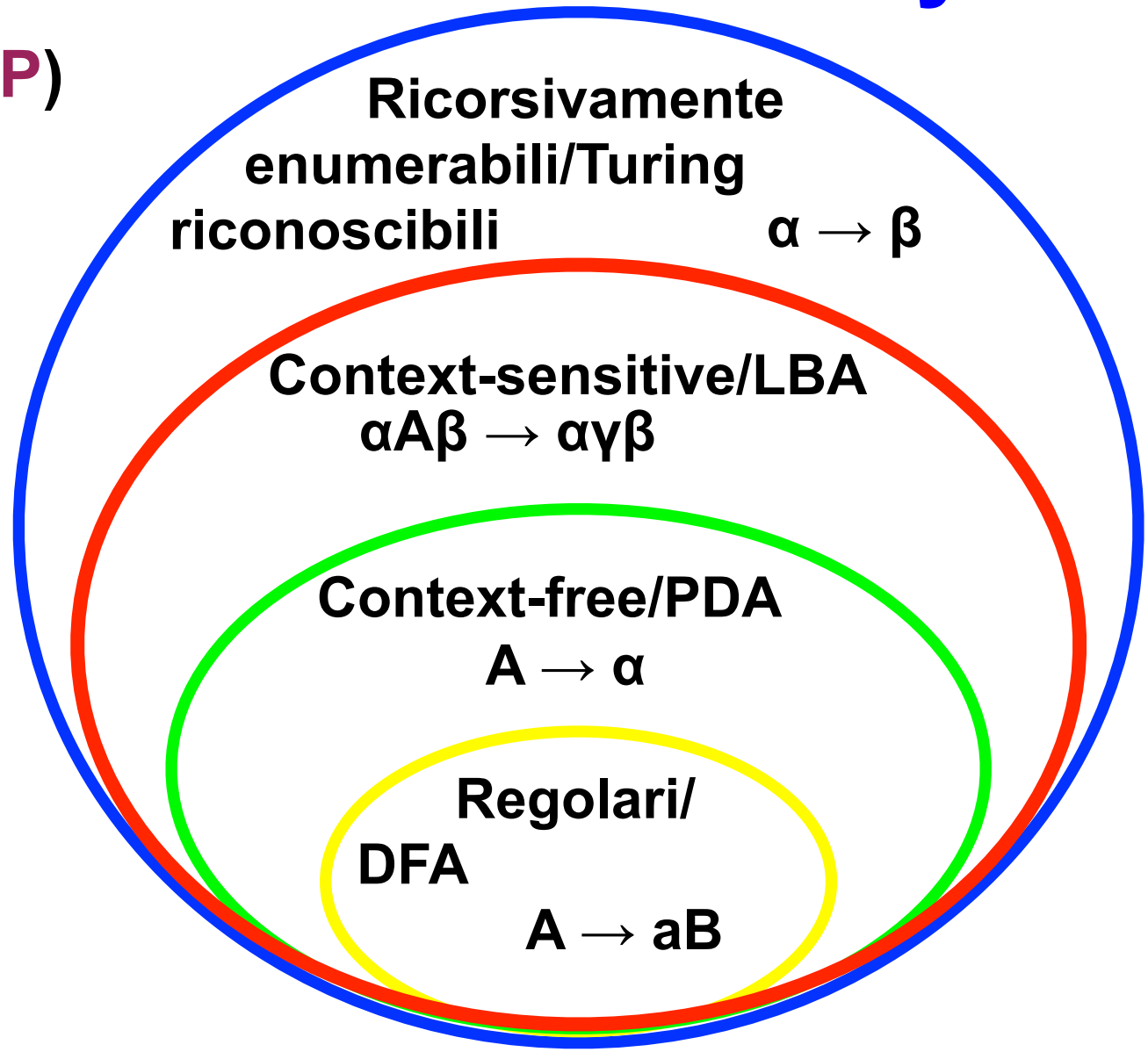
G = (T,V,S,P)

a in T∪{ε}

A,B in V,

α, β in $(T \cup V)^*$,

γ in $(T \cup V)^+$



Forme normali

Una forma normale per una CFG si ottiene imponendo delle restrizioni sulla forma delle produzioni, ma in modo tale da continuare a generare la stessa classe di linguaggi. Le forme normali rendono più facili le prove di alcuni risultati.

Forma normale di Chomsky

Una grammatica è detta in forma normale di Chomsky, brevemente CNF (da Chomsky Normal Form), se le produzioni sono della forma

$$A \rightarrow BC$$
$$A \rightarrow a$$

Si può costruire una CFG in forma normale di Chomsky equivalente a una data, a meno della parola vuota.

$G1 : S \rightarrow aSb \mid \varepsilon$ $G2 : S \rightarrow AD$ $L(G1) - \{ \varepsilon \} = L(G2)$

$$D \rightarrow SB$$
$$A \rightarrow a$$
$$B \rightarrow b$$

Algoritmo per l'appartenenza

Problema A_{CFG}: data una grammatica contex-free G e una parola x sul suo alfabeto dei terminali, x è generata da G?

Osserviamo che se G è in CNF ogni parola di lunghezza n è generata in 2n-1 passi: n-1 passi per generare a partire dal simbolo iniziale una parola di variabili di lunghezza n e n passi per trasformare ogni non terminale in un terminale.

Allora basta considerare tutte le possibili derivazioni di lunghezza 2n-1 e verificare se una di queste genera x!

Complessità: $O(2^n)$

Algoritmo per l'appartenenza

John Cocke (1970)

Daniel Younger (1966)

Tadao Kasami (1065)

indipendentemente hanno elaborato un algoritmo, chiamato CYK, per l'appartenenza che ha complessità di tempo

$$O(n^3)$$

dove n è la lunghezza della parola cercata.

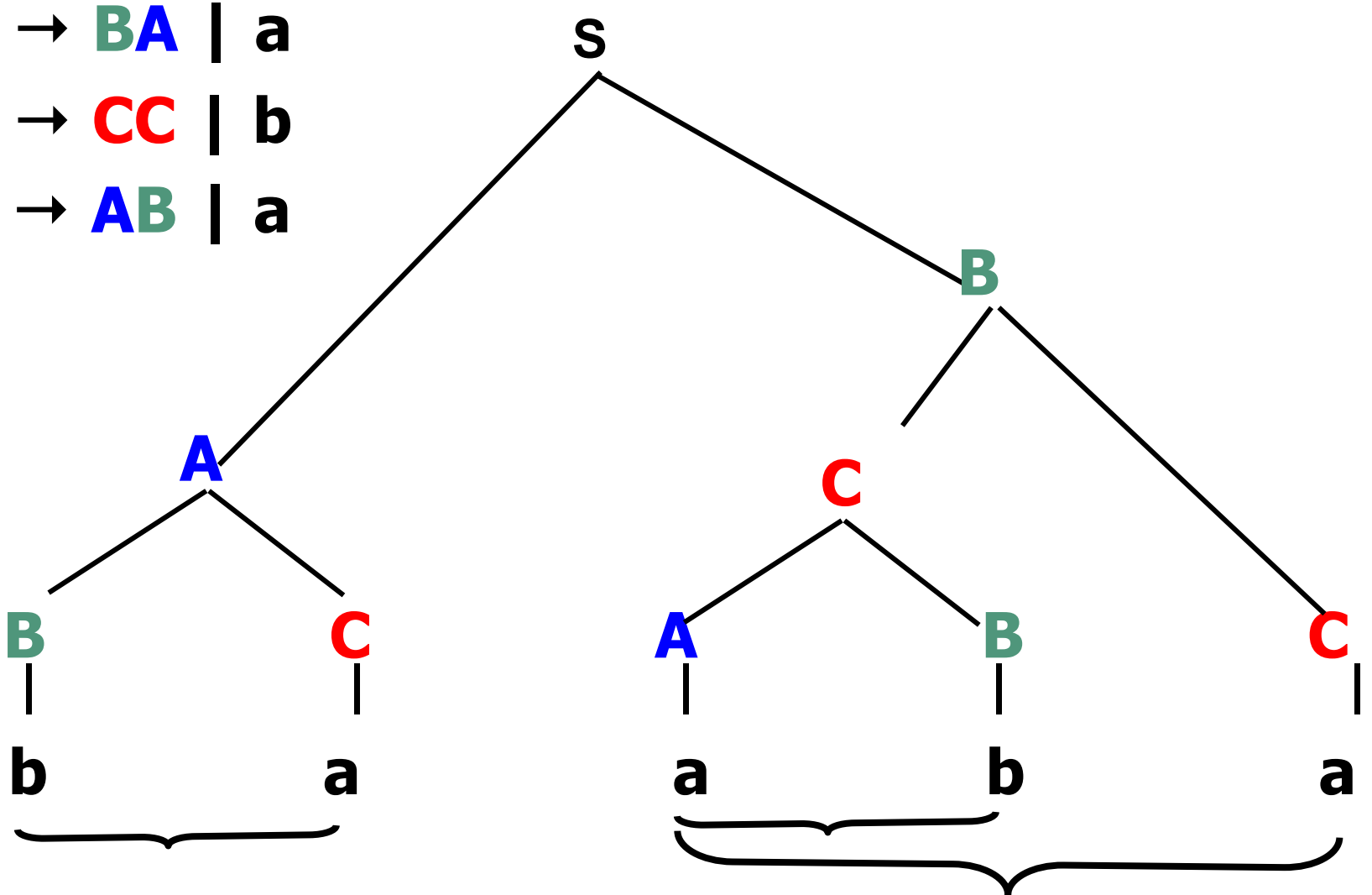
1 - Algoritmo CYK: l'idea

S → AB | BC

A → BA | a

B → CC | b

C → AB | a



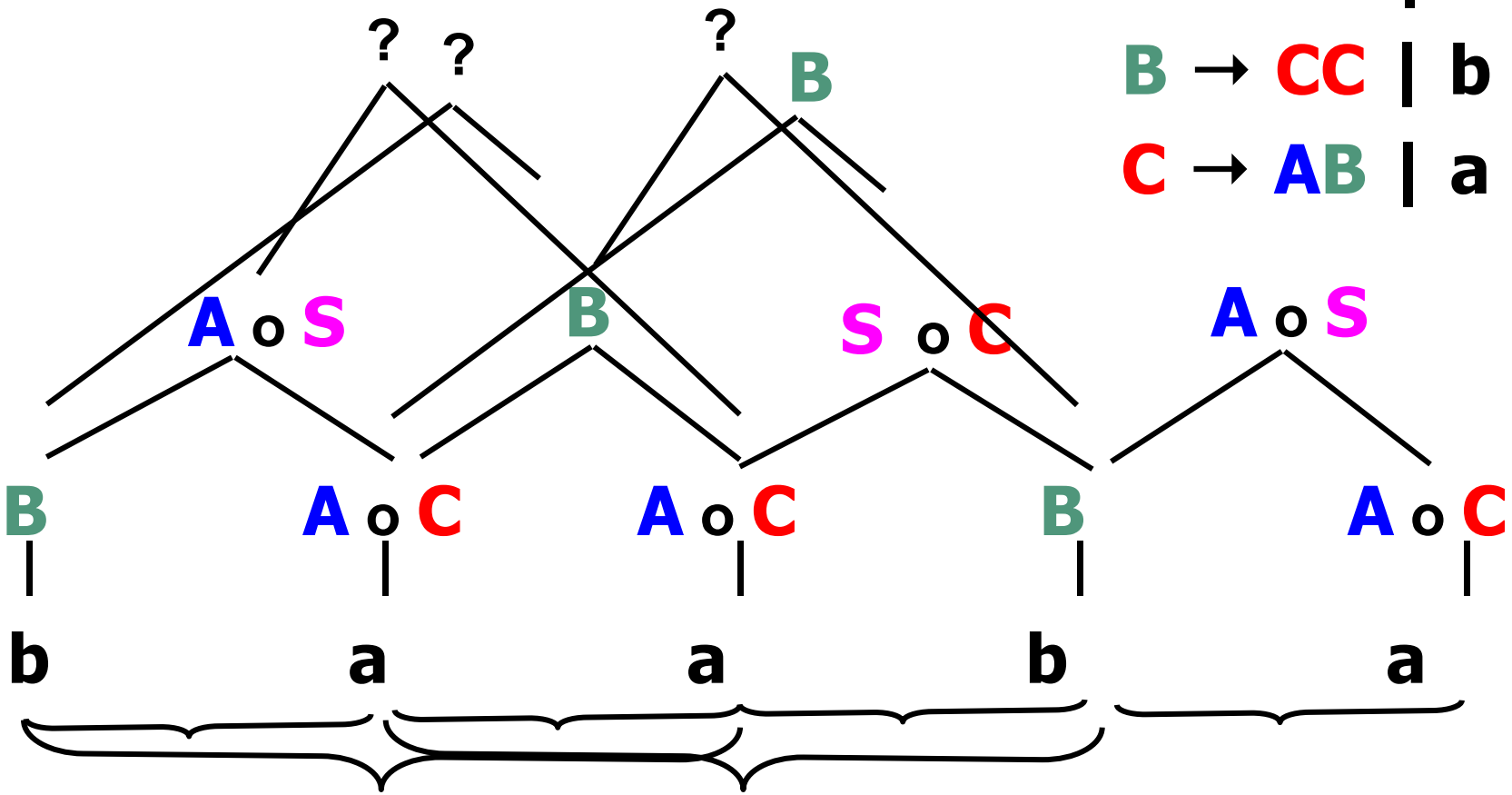
2 - Algoritmo CYK: l'idea

$S \rightarrow AB \mid BC$

$A \rightarrow BA \mid a$

$B \rightarrow CC \mid b$

$C \rightarrow AB \mid a$



Algoritmo CYK: definizioni preliminari

Sia G = (T,V,S,P) una CFG.
Data una parola w= $a_1 \ldots a_n$, denotiamo con x_{ij} la sottostringa di w determinata dai simboli tra l'i-simo e il j-simo: $x_{ij} = a_i \ldots a_j$

e chiamiamo V_{ij} l'insieme delle variabili che possono generare x_{ij} , formalmente

$$V_{ij} = \{A \mid A \Rightarrow^* x_{ij}\}$$

Se $S \in V_{1n}$ allora $x \in L(G)$

La matrice dei sottoinsiemi

	1	2	3	4	5
1	V_{11}	V_{12}	V_{13}	V_{14}	V_{15}
2		V_{22}	V_{23}	V_{24}	V_{25}
3			V_{33}	V_{34}	V_{35}
4				V_{44}	V_{45}
5					V_{55}

Se una parola ha lunghezza 5, l'algoritmo calcola V_{15} partendo dal calcolo di V_{11},..., V_{55}

Per esempio per calcolare V_{24} si devono considerare le coppie di insiemi V_{22} ,V_{34} e V_{23} ,V_{44}

Per produrre V_{ij} si considerano j-i coppie di insiemi tra quelli già prodotti, V_{ii} e V_{i+1j}, V_{ii+1} e V_{i+2j}, …, V_{ij-1} e V_{jj}, che corrispondono alle scomposizioni di $x_{ij} = a_i \ldots a_j$

a_i e $a_{i+1} \ldots a_j$, $a_i a_{i+1}$ e $a_{i+2} \ldots a_j$, $a_i \ldots a_{j-1}$ e a_j

seconda diagonale

$S \rightarrow AB \mid BC$

$A \rightarrow BA \mid a$

$B \rightarrow CC \mid b$

$C \rightarrow AB \mid a$

	1	2	3	4	5
1	$V_{11}=B$	$V_{12}=A, S$			
2		$V_{22}= A,C$			
3			$V_{33}= A,C$		
4				$V_{44}=B$	
5					$V_{55}= A,C$

B in V_{11} perchè $B \Rightarrow b$

A,C in V_{22} perchè $A \Rightarrow a$ e $C \Rightarrow a$

$S \rightarrow BC$ e $A \rightarrow BA$ sono produzioni, infatti

$S \Rightarrow BC \Rightarrow bC \Rightarrow ba$ e

$A \Rightarrow BA \Rightarrow bA \Rightarrow ba$

$w = \underbrace{baaba}$

terza diagonale

$S \rightarrow AB \mid BC$

$A \rightarrow BA \mid a$

$B \rightarrow CC \mid b$

$C \rightarrow AB \mid a$

	1	2	3	4	5
1	$v_{11}=B$	$v_{12}=A, S$	$v_{13}=\varnothing$		
2		$v_{22}=A,C$	$v_{23}=B$		
3			$v_{33}=A,C$	$v_{34}=S,C$	
4				$v_{44}=B$	$v_{45}=S,A$
5					$v_{55}=A,C$

$w = \underbrace{baa}ba$

v_{11} e v_{23} per la scomposizione b e aa e

v_{12} e v_{33} per la scomposizione ba e a.

terza diagonale 2

$S \rightarrow AB \mid BC$

$A \rightarrow BA \mid a$

$B \rightarrow CC \mid b$

$C \rightarrow AB \mid a$

	1	2	3	4	5
1	$V_{11}=B$	$V_{12}=A, S$	$V_{13}=\varnothing$		
2		$V_{22}=A,C$	$V_{23}=B$	$V_{24}=B$	
3			$V_{33}=A,C$	$V_{34}=S,C$	
4				$V_{44}=B$	$V_{45}=S,A$
5					$V_{55}=A,C$

$V_{22} = A,C$

$V_{34} = S,C$

$B \rightarrow CC$ è una prod.

allora B va in V_{24}

infatti $B \Rightarrow CC \Rightarrow Cab \Rightarrow aab$

$w = \underbrace{baa}_{}ba$

V_{22} e V_{34} per la scomposizione **a** e **ab** e

V_{12} e V_{33} per la scomposizione **ba** e **a**.

Esempio di esecuzione

$S \rightarrow AB \mid BC$

$A \rightarrow BA \mid a$

$B \rightarrow CC \mid b$

$C \rightarrow AB \mid a$

	1	2	3	4	5
1	$V_{11}=B$	$V_{12}=A, S$	$V_{13}=\varnothing$	$V_{14}=\varnothing$	$V_{15}=A,S,C$
2		$V_{22}=A,C$	$V_{23}=B$	$V_{24}=B$	$V_{25}=S, A,C$
3			$V_{33}=A,C$	$V_{34}=S,C$	$V_{35}=B$
4				$V_{44}=B$	$V_{45}=S,A$
5					$V_{55}=A,C$

$S \in V_{15}$ quindi **baaba** $\in L(G)$

$w =$ **baaba**

V_{11} e V_{25} per la scomposizione **b** e **aaba** e
V_{12} e V_{35} per la scomposizione **ba** e **aba**.
V_{13} e V_{45} per la scomposizione **baa** e **ba**.
V_{14} e V_{45} per la scomposizione **baab** e **a**.

Algoritmo CYK: pseudocodice

Input $w = w_1 \ldots w_n$:
for i = 1 to n si inizializza la diagonale
  for ogni variabile A:
    if $A \rightarrow w_i$ è una regola della grammatica
    then metti A in V_{ii}.
for r = 2 to n r è la lunghezza della sottostringa
  for i = 1 to n - r + 1 i è la posizione di partenza della sottostringa
    j = i + r - 1, j è la posizione finale della sottostringa
    for k = i to j-1 k è il punto di separazione
      for each rule $A \rightarrow BC$:
        if B è in V_{ik} and C è in V_{k+1j}
        then metti A in V_{ij}.
if S è in V_{1n}, return 1 else return 0

Complessità di tempo $O(n^3)$.