Sommario

1. **Gerarchia di Chomsky**
2. **Proprietà di chiusura dei CFL**
3. **Forma normale di Chomsky per CFG**
4. **Problemi di decisione per CFG**

Gerarchia di Chomsky

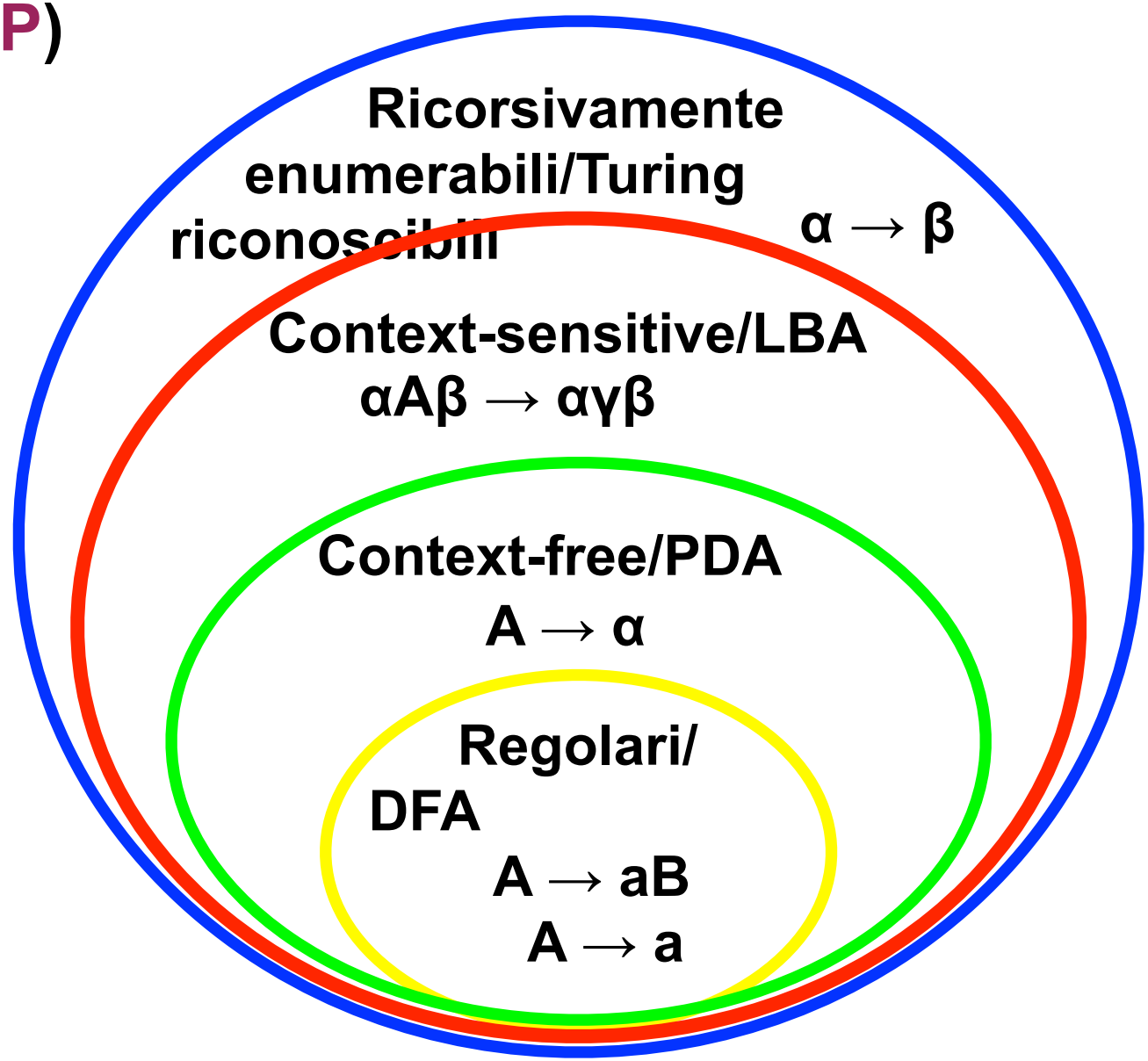
G = (T,V,S,P)

a in T

A,B in V,

α, β in $(T \cup V)^*$,

γ in $(T \cup V)^+$



Chiusura rispetto a unione CFL

Sia $G_1 = (V_1, T, S_1, P_1)$ e $G_2 = (V_2, T, S_2, P_2)$, con $V_1 \cap V_2 \neq \varnothing$, allora la grammatica per l'unione è

$G = (V, T, S, P)$ dove:

$S \notin V_1 \cup V_2$

$V = V_1 \cup V_2 \cup \{S\}$

$P = P_1 \cup P_2 \cup \{S \rightarrow S_1 | S \rightarrow S_2\}$

$S_1 \Rightarrow^* x$ e $S_2 \Rightarrow^* y \Leftrightarrow S \Rightarrow S_1 \Rightarrow^* x$ o $S \Rightarrow S_2 \Rightarrow^* y$,

quindi $L(G_1) \cup L(G_2) = L(G)$,

Chiusura rispetto a intersezione CFL

La classe NON è chiusa rispetto all'intersezione.

Il linguaggio

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

non è content-free, ma può essere ottenuto come intersezione di due linguaggi contesto-free:

$L_1 = \{a^n b^n c^m \mid n,m \geq 0\}$ e $L_2 = \{a^m b^n c^n \mid n,m \geq 0\}$.

Chiusura rispetto a complemento?

NO! data la chiusura rispetto all'unione.

Chiusura rispetto a prodotto CFL

Sia $G_1 = (V_1, T, S_1, P_1)$ e $G_2 = (V_2, T, S_2, P_2)$, con $V_1 \cap V_2 \neq \varnothing$, allora la grammatica per il prodotto è

$G = (V, T, S, P)$ dove:

$S \notin V_1 \cup V_2$

$V = V_1 \cup V_2 \cup \{S\}$

$P = P_1 \cup P_2 \cup \{S \rightarrow S_1 S_2\}$

$S_1 \Rightarrow^* x$ e $S_2 \Rightarrow^* y \Leftrightarrow S \Rightarrow S_1 S_2 \Rightarrow^* xy$,

quindi $L(G_1)L(G_2) = L(G)$,

Chiusura rispetto a stella di Kleene CFL

Sia $G_1 = (V_1, T, S_1, P_1)$, allora la grammatica per la stella di Kleene è

$G = (V, T, S, P)$ dove:

$S \notin V_1$

$V = V_1 \cup \{S\}$

$P = P_1 \cup \{S \rightarrow S_1 S,\ S \rightarrow \varepsilon\}$

$S_1 \Rightarrow^* x_1$, $S_1 \Rightarrow^* x_2$,…, $S_1 \Rightarrow^* x_n$, $\Leftrightarrow$

$S \Rightarrow S_1 S \Rightarrow \ldots \Rightarrow S_1 S_1 \ldots S_1 S \Rightarrow^* x_1 x_2 \ldots x_n$
quindi $L(G_1)^* = L(G)$,

Decidere se L = ∅, quando L è CFL

Algoritmo per decidere se il linguaggio generato da una CFG è vuoto.

Input: una CFG =(V,T,S,P)

Marca i terminali

1. **Ripeti fino a quando non ci sono variabili che si possono marcare**
   marca ogni variabile parte sinistra di una regola in cui la parte destra è marcata
3. **Se S non è marcato rispondi sì, altrimenti no.**

Esempio:

G: S → AB | AC
C → SB
A → a
B → b

Alla prima iterazione si marcano A e B, alla seconda S, alla terza C e fine, perchè sono tutte marcate.

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.

Supponiamo che le grammatiche non abbiano simboli inutili, cioè simboli da cui non si deriva un terminale o che non occorrono in parole derivate dal simbolo iniziale .

Forma normale di Chomsky

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

G : S → aSb | ε

Passo 1. Eliminazione S → ε ed eventuali A → B

G1 : S → aSb | ab L(G1) - { ε } = L(G)

Passo 2. Aggiunta nuove variabili per i terminali

G2 : S → ASB | AB
A → a
B → b

Passo 3. riduzione lunghezza parte destra

G3 : S → AD | AB
D → SB
A → a
B → b

Forma normale di Chomsky

G : S → aXbX

X→ aY | bY| ε

Y → X | c

Passo 1. Eliminazione X → ε, ma Y → X quindi anche Y è annullabile

G1 : S → aXbX |aXb | abX | ab

X→ aY | bY| a | b

Y → X | c

Eliminazione di Y → X

Forma normale di Chomsky

G1 : S → aXbX |aXb | abX | ab
X→ aY | bY| a | b
Y → X | c

Eliminazione di Y → X

G2 : S → aXbX |aXb | abX | ab
X→ aY | bY| a | b
Y → aY | bY| a | b| c

Forma normale di Chomsky

G2 : S → aXbX |aXb | abX | ab

X→ aY | bY| a | b

Y → aY | bY| a | b| c

Passo 3. riduzione lunghezza parte destra

G3 : S → AXBX |AXB | ABX | AB

X→ AY | BY| a | b

Y → AY | BY| a | b| c

A→ a

B → b

Forma normale di Chomsky

Passo 3. riduzione lunghezza parte destra

G3 : S → AXBX |AXB | ABX | AB

X→ AY | BY| a | b

Y → AY | BY| a | b| c

A→ a

B → b

G4 : S → AD |AF | AG | AB

D → XE

E → BX

F → XB

G → BX

X→ AY | BY| a | b

Y → AY | BY| a | b| c

A→ a

B → b

Forma normale di Chomsky

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

Passo 1. Eliminazione regole di cancellazione $A \rightarrow \varepsilon$
Passo 2 Eliminazione eventuali regole del tipo $D \rightarrow B$
Passo 3 introdurre una variabile per ogni terminale e sostituire la variabile al terminale in ogni parte destra di una regola in cui occorre
Passo 4 ridurre la lunghezza delle parti destre delle regole introducendo nuove variabili

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.