# Gerarchia di Chomsky

# Gerarchia di Chomsky

| Tipo | Linguaggio | Grammatica | Automa |
| --- | --- | --- | --- |
| 0 | Ricorsivamente Enumerabile<br>SEMIDECIDIBILE | Senza restrizioni | Turing Machine |
| 1 | Dipendente dal contesto<br>Ricorsivo, DECIDIBILE | Dipendente dal contesto | Linearly Bounded Automaton |
| 2 | Libero dal contest<br>Ricorsivo, DECIDIBILE | Libera dal contesto | Non Det. PushDown Automaton |
| 3 | Regolare<br>Ricorsivo, DECIDIBILE | Regolare | DFA, NFA |

Potere espressivo

Tipo 3 $\subset$ Tipo 2 $\subset$ Tipo 1 $\subset$ Tipo 0

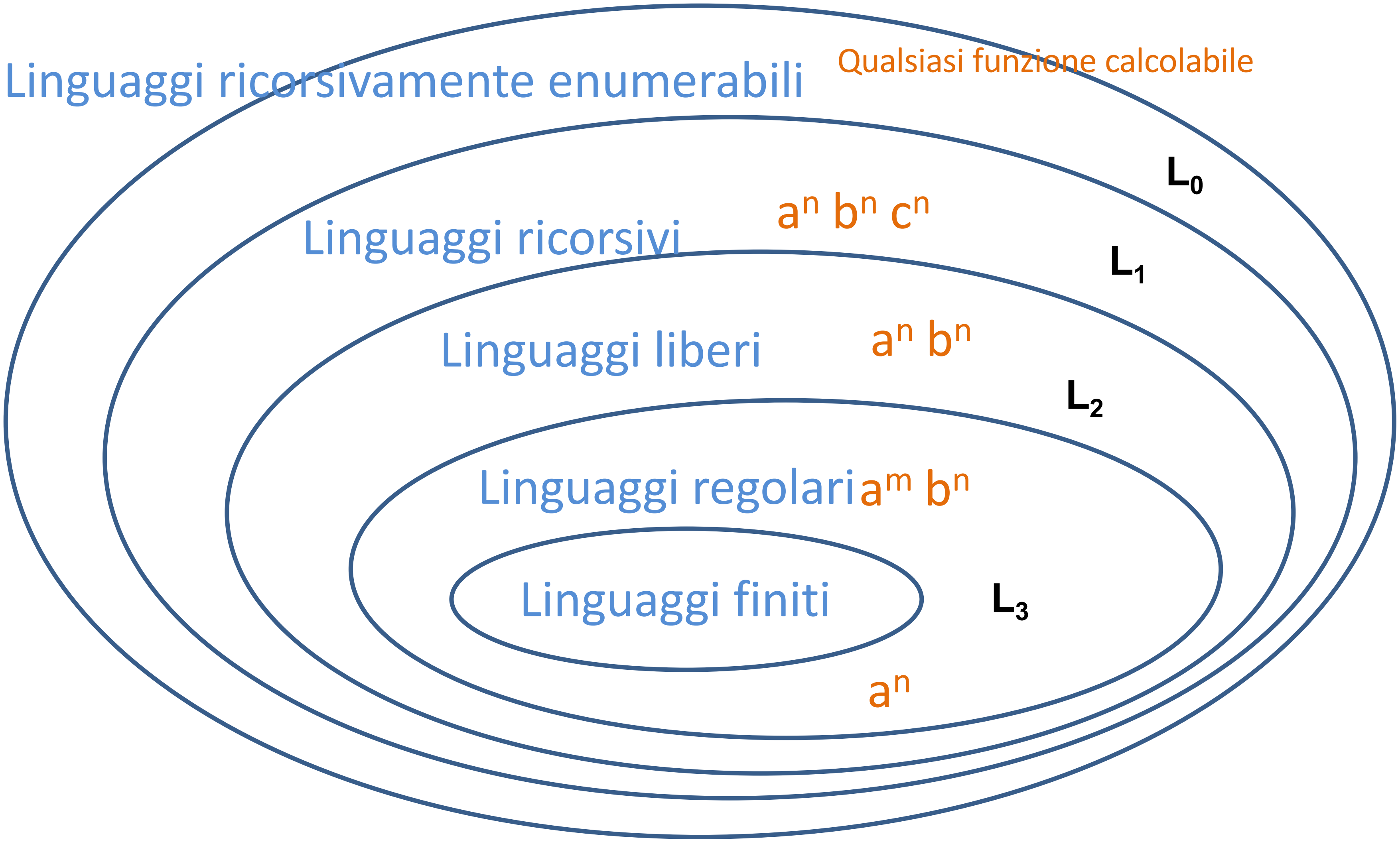
# Note sulla calcolabilità

- Un linguaggio L(G) è **decidibile** se è decidibile se una stringa w appartiene a L(G)
- Un linguaggio L(G) è **semidecidibile** se è possibile stabilire se una stringa w appartiene a L(G) , ma non è possibile stabilire la **non** appartenenza
- Esistono linguaggi ricorsivi che non sono dipendenti dal contesto

# Note sulla calcolabilità (cont.)

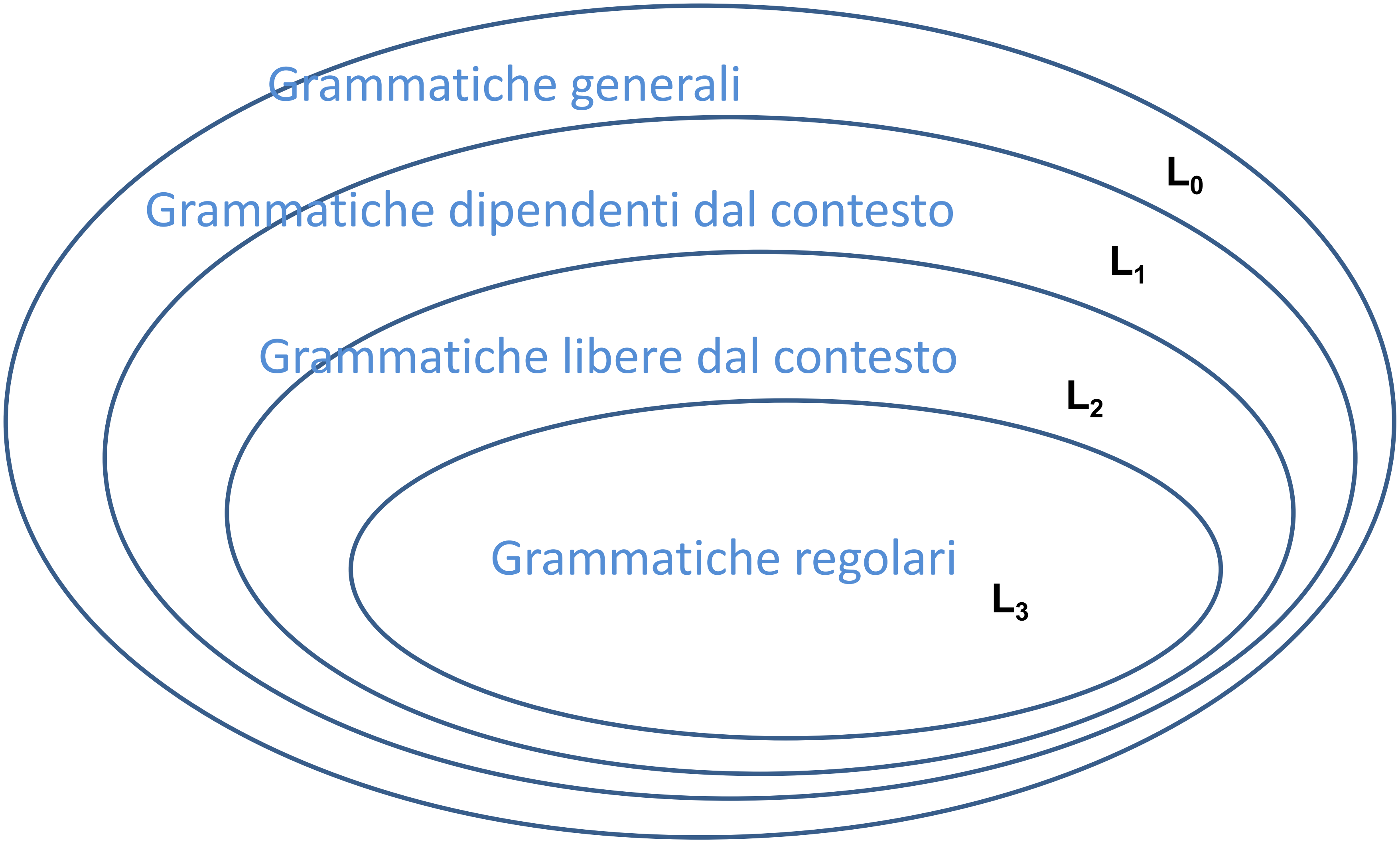
- Un algoritmo (che termina) A definisce una funzione $f_A$ che ha come dominio l'insieme dei possibili dati iniziali e come codominio l'insieme dei possibili dati finali.
- Definiamo l'insieme delle **funzioni calcolabili** $F_S = \{f_A \mid A \in \mathbf{A}$ (insieme di tutti gli algoritmi)$\}$
- Si può dimostrare che le funzioni calcolabili sono un piccolissimo sottoinsieme di tutte le funzioni

# Gerarchia di Chomsky




$L0 \supset L1 \supset L2 \supset L3 \supset L$ finiti

# Gerarchia di grammatiche



# Grammatiche Generali*

## Grammatiche di *tipo 0*

Produzioni: $\alpha$ --> $\beta$, dove $\alpha$, $\beta$ sono stringhe di terminali e non terminali

Ex:

- S --> aBc
- aB --> cA
- Ac --> d

Un linguaggio generato da una grammatica di questo genere è detto ricorsivamente enumerabile

* O *non ristrette* o *a struttura di frase*

# Grammatiche dipendenti dal contesto

## Grammatiche di *tipo 1*

Produzioni: $\alpha \dashrightarrow \beta$, dove $\alpha$, $\beta$ sono stringhe di terminali e non terminali e $|\alpha| \leq |\beta|$

Ex:

- S --> abc | aAbc
- aB --> bA
- Ac --> Bbcc
- bB --> Bb
- aB --> aa | aaA

Genera $a^n b^n c^n$

# Grammatiche libere dal contesto

**Grammatiche di *tipo 2***

Produzioni: S --> β, dove S ∈ V, è un simbolo non terminale e β ∈ (V ∪ T)* è una stringa di terminali e non terminali

Ex:

- S --> ab | aSb

Genera il linguaggio $a^n b^n$

# Grammatiche regolari

## **Grammatiche di *tipo 3***

Produzioni: S --> v, dove S $\in$ V è un simbolo non terminale e v è una stringa composta da un terminale o da un terminale e un non terminale

Ex:

- S --> a | aS

Genera $a^n$

# Stringa vuota

Dato una grammatica G che genera il linguaggio L(G) senza ε, si può ottenere L(G) ∪ {ε}, con un nuovo simbolo iniziale S'

- S' --> ε
- S' --> α [per ogni produzione S --> α in G]

Si noti che in questo modo S' non appare mai sulla parte destra di una produzione.

Si possono allora estendere le definizioni delle grammatiche viste, così da permettere una produzione S' --> ε, senza mai avere S' sulla parte destra di una produzione