Riassunto delle cose utili - Primo parziale

Gabriel Rovesti

Indice

1 Introduzione

Questo documento presenta una raccolta sistematizzata delle principali tecniche di dimostrazione utilizzate per i linguaggi formali, con particolare attenzione a:

- Dimostrare se un linguaggio è regolare
- Dimostrare se un linguaggio non è regolare
- Dimostrare se un linguaggio è context-free
- Dimostrare la chiusura delle classi di linguaggi rispetto a varie operazioni

Per ogni categoria, verranno presentati i principi teorici, le tecniche specifiche con esempi di applicazione, e schemi risolutivi riutilizzabili.

2 Dimostrare se un linguaggio è regolare

Per dimostrare che un linguaggio è regolare, esistono diverse tecniche:

2.1 Costruzione diretta di un automa a stati finiti

La tecnica più diretta consiste nel costruire esplicitamente un DFA (Deterministic Finite Automaton) o un NFA (Non-deterministic Finite Automaton) che riconosca il linguaggio.

Schema generale:

1. Identificare l'alfabeto Σ
2. Definire l'insieme degli stati Q (includendo uno stato iniziale q_0 e stati finali F)
3. Definire la funzione di transizione δ
4. Verificare che l'automa accetti esattamente le stringhe del linguaggio

Esempio: Dimostriamo che il linguaggio $L = \{w \in \{0,1\}^* \mid w \text{ ha un numero pari di } 1\}$ è regolare.

Soluzione: Costruiamo un DFA $A = (Q, \Sigma, \delta, q_0, F)$ dove:

- $Q = \{q_0, q_1\}$ dove q_0 rappresenta "numero pari di 1" e q_1 "numero dispari di 1"
- $\Sigma = \{0, 1\}$
- $\delta(q_0, 0) = q_0$ (lo 0 non cambia la parità)
- $\delta(q_0, 1) = q_1$ (un 1 cambia da pari a dispari)
- $\delta(q_1, 0) = q_1$ (lo 0 non cambia la parità)
- $\delta(q_1, 1) = q_0$ (un 1 cambia da dispari a pari)
- q_0 è lo stato iniziale
- $F = \{q_0\}$ sono gli stati finali (accettiamo stringhe con numero pari di 1)

2.2 Utilizzare operazioni di chiusura dei linguaggi regolari

I linguaggi regolari sono chiusi rispetto a diverse operazioni, tra cui unione, intersezione, concatenazione, stella di Kleene, complemento e differenza.

Schema generale:

1. Identificare operazioni che, applicate a linguaggi regolari, producono il linguaggio desiderato
2. Dimostrare che i linguaggi componenti sono regolari
3. Concludere che il linguaggio risultante è regolare per le proprietà di chiusura

Esempio: Dimostriamo che il linguaggio $L = \{w \in \{0,1\}^* \mid w \text{ non contiene la sottostringa } 101\}$ è regolare.

Soluzione: Osserviamo che $L = \{0,1\}^* \setminus (\{0,1\}^* \cdot \{101\} \cdot \{0,1\}^*)$. Poiché $\{0,1\}^*$ è regolare, $\{101\}$ è finito (quindi regolare), la concatenazione preserva la regolarità, e il complemento di un linguaggio regolare è regolare, ne consegue che L è regolare.

2.3 Esprimere il linguaggio attraverso un'espressione regolare

Se riusciamo a descrivere il linguaggio tramite un'espressione regolare, allora è regolare per definizione.

2.4 Costruzione di automi per operazioni specifiche

Per alcune operazioni comuni su linguaggi, possiamo costruire automi specifici.

Esempio: Riconoscere suffissi di un linguaggio Consideriamo l'operazione $suffixes(L) = \{y \mid xy \in L \text{ per qualche } x \in \Sigma^*\}$. Se L è regolare, anche $suffixes(L)$ è regolare.

Soluzione: Sia $A = (Q, \Sigma, \delta, q_0, F)$ un DFA che riconosce L. Costruiamo un NFA $A' = (Q', \Sigma, \delta', q_0', F')$ che riconosce $suffixes(L)$:

- $Q' = Q \cup \{q_0'\}$ dove q_0' è un nuovo stato
- $\delta'(q_0', a) = \{\delta(q_0, a), q_0'\}$ per ogni $a \in \Sigma$
- $\delta'(q, a) = \{\delta(q, a)\}$ per ogni $q \in Q$, $a \in \Sigma$
- $F' = F$

Questo NFA può iniziare a simulare A da qualsiasi punto della stringa di input, riconoscendo così tutti i suffissi di L.

3 Dimostrare se un linguaggio non è regolare

3.1 Pumping Lemma per linguaggi regolari

Il metodo principale per dimostrare che un linguaggio non è regolare è il Pumping Lemma.

Teorema (Pumping Lemma): Per ogni linguaggio regolare L, esiste un intero $p > 0$ (la "lunghezza di pumping") tale che, per ogni stringa $s \in L$ con $|s| \geq p$, s può essere suddivisa in tre parti $s = xyz$ con le seguenti proprietà:

1. $|xy| \leq p$
2. $|y| > 0$
3. Per ogni $i \geq 0$, $xy^iz \in L$

Schema generale per la dimostrazione per contraddizione:

1. Assumere per assurdo che il linguaggio L sia regolare
2. Applicare il Pumping Lemma: esiste un $p > 0$ con le proprietà indicate
3. Scegliere una stringa $s \in L$ con $|s| \geq p$ opportunamente costruita

4. Mostrare che per ogni divisione $s = xyz$ che soddisfa le condizioni 1 e 2 del lemma, esiste un $i \geq 0$ tale che $xy^iz \notin L$, contraddicendo la condizione 3

5. Concludere che L non è regolare

Esempio: Dimostriamo che il linguaggio $L = \{0^n1^n \mid n \geq 0\}$ non è regolare.

Soluzione: Supponiamo per assurdo che L sia regolare. Allora esiste un intero $p > 0$ come nel Pumping Lemma. Consideriamo la stringa $s = 0^p1^p \in L$. Per il Pumping Lemma, s può essere scritta come $s = xyz$ con $|xy| \leq p$, $|y| > 0$ e $xy^iz \in L$ per ogni $i \geq 0$.

Dato che $|xy| \leq p$, entrambe x e y sono composte solo da 0. Sia $y = 0^k$ con $k > 0$. Consideriamo $xy^0z = xz$. Questa stringa contiene $p - k$ zeri e p uni. Poiché $k > 0$, il numero di zeri è minore del numero di uni, quindi $xz \notin L$. Questo contraddice il Pumping Lemma, quindi L non è regolare.

3.2 Dimostrazioni strutturali

In alcuni casi, possiamo dimostrare che un linguaggio non è regolare utilizzando proprietà strutturali dei linguaggi regolari.

Esempio: Dimostriamo che il linguaggio $L = \{ww \mid w \in \{a, b\}^*\}$ non è regolare.

Soluzione: Supponiamo per assurdo che L sia regolare. Per il Pumping Lemma, esiste un intero $p > 0$ tale che ogni stringa $s \in L$ con $|s| \geq p$ può essere decomposta come $s = xyz$ con $|xy| \leq p$, $|y| > 0$ e $xy^iz \in L$ per ogni $i \geq 0$.

Consideriamo la stringa $s = a^pb^pa^pb^p \in L$. Per il Pumping Lemma, possiamo scrivere $s = xyz$ con le proprietà indicate. Dato che $|xy| \leq p$, xy è contenuta interamente nella prima parte a^p di s. Sia $y = a^k$ con $0 < k \leq p$.

Consideriamo $xy^2z = xa^{2k}z$. Questa stringa avrà $p + k$ lettere a nella prima metà, ma solo p lettere a nella seconda metà. Quindi xy^2z non può essere della forma ww e non appartiene a L, contraddicendo il Pumping Lemma. Pertanto, L non è regolare.

3.3 Proprietà di chiusura e intersezione con linguaggi regolari

I linguaggi regolari sono chiusi rispetto all'intersezione con altri linguaggi regolari. Possiamo sfruttare questa proprietà per dimostrare che un linguaggio non è regolare.

Schema generale:

1. Assumere per assurdo che il linguaggio L sia regolare
2. Trovare un linguaggio regolare R tale che $L \cap R$ è un linguaggio noto non regolare
3. Poiché i linguaggi regolari sono chiusi rispetto all'intersezione, se L fosse regolare, anche $L \cap R$ sarebbe regolare
4. Concludere che L non può essere regolare

Esempio: Consideriamo $L = \{a^i b^j c^k \mid i, j, k \geq 0 \text{ e se } j > 0 \text{ allora } i = k\}$. Dimostriamo che L non è regolare.

Soluzione: Supponiamo per assurdo che L sia regolare. Intersechiamo L con il linguaggio regolare $R = a^* b^+ c^*$. Otteniamo $L \cap R = \{a^i b^j c^i \mid i, j > 0\}$ che è un linguaggio non regolare (facilmente dimostrabile con il Pumping Lemma). Questo contraddice la chiusura dei linguaggi regolari rispetto all'intersezione. Quindi L non è regolare.

3.4 Varianti del Pumping Lemma nell'applicazione

Quando si applica il Pumping Lemma, la scelta della stringa e l'analisi delle possibili decomposizioni sono cruciali. Ecco alcune varianti comuni:

1. Linguaggi della forma $L = \{a^n b^n \mid n \geq 0\}$: Come visto nell'esempio precedente, si sceglie $s = a^p b^p$ e si dimostra che il "pumping" nella parte degli a porta a stringhe non nel linguaggio.

2. Linguaggi della forma $L = \{a^{n^2} \mid n \geq 0\}$: Si sceglie $s = a^{p^2}$ e si mostra che aggiungere o rimuovere un numero di a non trasforma il numero totale in un quadrato perfetto.

3. Linguaggi con vincoli sulla parità o sull'uguaglianza di conteggi: Come $L = \{w \in \{0,1\}^* \mid w \text{ ha lo stesso numero di 0 e di 1}\}$, dove si sceglie $s = 0^p 1^p$ e si dimostra che modificare il conteggio di un simbolo rompe l'equità.

4. Linguaggi con relazioni o condizioni strutturali: Come $L = \{w\#w \mid w \in \{a,b\}^*\}$, dove si sceglie $s = a^p \# a^p$ e si mostra che alterando una parte si rompe la relazione di uguaglianza.

4 Dimostrare se un linguaggio è context-free

4.1 Costruzione di una grammatica context-free

Il metodo più diretto è costruire una grammatica context-free che generi esattamente il linguaggio desiderato.

Schema generale:

1. Identificare i componenti strutturali del linguaggio
2. Definire le variabili non terminali che rappresentano questi componenti
3. Definire le regole di produzione
4. Verificare che la grammatica generi esattamente il linguaggio desiderato

Esempio: Dimostriamo che il linguaggio $L = \{a^n b^n \mid n \geq 0\}$ è context-free.

Soluzione: Costruiamo una grammatica $G = (V, \Sigma, R, S)$ dove:

- $V = \{S\}$
- $\Sigma = \{a, b\}$
- $R = \{S \to aSb \mid \varepsilon\}$

Questa grammatica genera esattamente le stringhe della forma $a^n b^n$ per $n \geq 0$:

- La regola $S \to \varepsilon$ genera la stringa vuota (caso $n = 0$)
- La regola $S \to aSb$ permette di aggiungere una coppia a, b attorno a una stringa già generata, garantendo che il numero di a sia uguale al numero di b

4.2 Costruzione di un automa a pila (PDA)

Un altro approccio consiste nel costruire un PDA (Pushdown Automaton) che riconosca il linguaggio.

Schema generale:

1. Identificare come utilizzare la pila per monitorare le proprietà necessarie
2. Definire gli stati e le transizioni del PDA
3. Verificare che il PDA accetti esattamente le stringhe del linguaggio

Esempio: Dimostriamo che il linguaggio $L = \{a^n b^n \mid n \geq 0\}$ è context-free usando un PDA.

Soluzione: Costruiamo un PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ dove:

- $Q = \{q_0, q_1, q_2\}$
- $\Sigma = \{a, b\}$
- $\Gamma = \{Z_0, A\}$ (Z_0 è il simbolo iniziale della pila, A è usato per contare le a)
- Transizioni δ:
  - $\delta(q_0, a, Z_0) = \{(q_0, AZ_0)\}$ (se vedo a, spingo A sulla pila)
  - $\delta(q_0, a, A) = \{(q_0, AA)\}$ (se vedo a, spingo A sulla pila)
  - $\delta(q_0, b, A) = \{(q_1, \varepsilon)\}$ (se vedo b, prelevo A dalla pila)
  - $\delta(q_1, b, A) = \{(q_1, \varepsilon)\}$ (se vedo b, prelevo A dalla pila)
  - $\delta(q_1, \varepsilon, Z_0) = \{(q_2, Z_0)\}$ (transizione ε allo stato finale)
- q_0 è lo stato iniziale
- Z_0 è il simbolo iniziale della pila
- $F = \{q_2\}$ è l'insieme degli stati finali

Il PDA funziona così: per ogni a in input, spinge un simbolo A sulla pila. Quando inizia a leggere b, passa allo stato q_1 e per ogni b rimuove un simbolo A. Se alla fine la pila contiene solo Z_0 (cioè tutti gli A sono stati rimossi), la stringa viene accettata.

4.3 Operazioni di chiusura sui linguaggi context-free

I linguaggi context-free sono chiusi rispetto a varie operazioni, come unione, concatenazione, stella di Kleene, sostituzione e omomorfismo.

Schema generale:

1. Identificare operazioni che, applicate a linguaggi context-free, producono il linguaggio desiderato
2. Dimostrare che i linguaggi componenti sono context-free
3. Concludere che il linguaggio risultante è context-free per le proprietà di chiusura

Esempio: Dimostriamo che il linguaggio $L = \{a^i b^j c^k \mid i, j, k \geq 0 \text{ e } i = j \text{ o } j = k\}$ è context-free.

Soluzione: Osserviamo che $L = L_1 \cup L_2$ dove $L_1 = \{a^i b^i c^k \mid i, k \geq 0\}$ e $L_2 = \{a^i b^j c^j \mid i, j \geq 0\}$.

L_1 è context-free perché può essere generato dalla grammatica $S \to aAb \mid A, A \to aAb \mid C, C \to cC \mid \varepsilon$.

L_2 è context-free perché può essere generato dalla grammatica $S \to aS \mid B, B \to bBc \mid \varepsilon$.

Poiché l'unione di linguaggi context-free è context-free, $L = L_1 \cup L_2$ è context-free.

5 Tecniche per dimostrare operazioni di chiusura

5.1 Chiusura dei linguaggi regolari

I linguaggi regolari sono chiusi rispetto alle seguenti operazioni:

- Unione, intersezione, complemento, differenza
- Concatenazione, stella di Kleene
- Inversione (reversal)
- Omomorfismo, omomorfismo inverso
- Sostituzione

Schema generale per dimostrare la chiusura:

1. Assumere che i linguaggi di partenza siano riconosciuti da DFA
2. Costruire un nuovo automa che riconosca il linguaggio risultante dall'operazione
3. Dimostrare che l'automa costruito è un DFA o NFA
4. Concludere che il linguaggio risultante è regolare

Esempio: Chiusura rispetto all'operazione "a/L" Definiamo $a/L = \{w \mid aw \in L\}$. Dimostriamo che se L è regolare, anche a/L è regolare.

Soluzione: Sia $A = (Q, \Sigma, \delta, q_0, F)$ un DFA che riconosce L. Costruiamo un DFA $A' = (Q, \Sigma, \delta, q_0', F)$ dove $q_0' = \delta(q_0, a)$.

A' simula cosa farebbe A dopo aver letto il simbolo a, quindi accetta esattamente le stringhe w tali che $aw \in L$.

Esempio: Chiusura rispetto a "ROL(L)" Definiamo $ROL(L) = \{wa \mid aw \in L, w \in \Sigma^*, a \in \Sigma\}$. Dimostriamo che se L è regolare, anche $ROL(L)$ è regolare.

Soluzione: Sia $A = (Q, \Sigma, \delta, q_0, F)$ un DFA che riconosce L. Costruiamo un NFA $A' = (Q', \Sigma, \delta', q_0', F')$ dove:

- $Q' = Q \times \Sigma \cup \{q_0'\}$
- $\delta'(q_0', a) = \{(q_0, a)\}$ per ogni $a \in \Sigma$
- $\delta'((q, b), a) = \{(\delta(q, a), b)\}$ per ogni $q \in Q$, $a, b \in \Sigma$
- $F' = \{(q, a) \mid \delta(q, a) \in F\}$

A' tiene traccia del primo simbolo letto nella seconda componente dello stato, e verifica che dopo aver letto l'intera stringa, aggiungendo questo primo simbolo alla fine si otterrebbe una stringa in L.

5.2 Chiusura dei linguaggi context-free

I linguaggi context-free sono chiusi rispetto alle seguenti operazioni:

- Unione, concatenazione, stella di Kleene
- Omomorfismo, sostituzione
- Intersezione con linguaggi regolari

Ma non sono chiusi rispetto a:

- Intersezione (in generale)
- Complemento

Schema generale per dimostrare la chiusura:

1. Assumere che i linguaggi di partenza siano generati da grammatiche context-free
2. Costruire una nuova grammatica che generi il linguaggio risultante dall'operazione
3. Dimostrare che la grammatica costruita è context-free
4. Concludere che il linguaggio risultante è context-free

Esempio: Chiusura rispetto all'operazione "suffix(L)" Definiamo $suffix(L) = \{y \mid xy \in L \text{ per qualche } x \in \Sigma^*\}$. Dimostriamo che se L è context-free, anche $suffix(L)$ è context-free.

Soluzione: Sia $G = (V, \Sigma, R, S)$ una grammatica context-free che genera L. Costruiamo una nuova grammatica $G' = (V', \Sigma, R', S')$ dove:

- $V' = V \cup \{S'\}$ dove S' è un nuovo simbolo non terminale
- R' contiene tutte le regole di R, più le seguenti nuove regole:
  - $S' \to S$
  - $S' \to aS'$ per ogni $a \in \Sigma$

Questa grammatica genera $suffix(L)$ perché:

- Può generare qualsiasi stringa in L (usando $S' \to S$ e poi le regole originali)
- Può aggiungere qualsiasi prefisso arbitrario a una stringa di L (usando ripetutamente $S' \to aS'$)

Esempio: Chiusura rispetto all'operazione "superstring(L)" Definiamo $superstring(L) = \{xyz \mid y \in L \text{ e } x, z \in \Sigma^*\}$. Dimostriamo che se L è context-free, anche $superstring(L)$ è context-free.

Soluzione: Sia $G = (V, \Sigma, R, S)$ una grammatica context-free che genera L. Costruiamo una nuova grammatica $G' = (V', \Sigma, R', S')$ dove:

- $V' = V \cup \{S', A, B\}$ dove S', A, B sono nuovi simboli non terminali
- R' contiene tutte le regole di R, più le seguenti nuove regole:
  - $S' \to ASB$
  - $A \to aA \mid \varepsilon$ per ogni $a \in \Sigma$
  - $B \to bB \mid \varepsilon$ per ogni $b \in \Sigma$

Questa grammatica genera $superstring(L)$ perché:

- Può generare qualsiasi prefisso $x \in \Sigma^*$ usando le regole di A
- Può generare qualsiasi stringa $y \in L$ usando le regole originali di G
- Può generare qualsiasi suffisso $z \in \Sigma^*$ usando le regole di B

6 Esempi di dimostrazioni complete

6.1 Esempio 1: $L = \{0^n 1^n \mid n \geq 1\}$ non è regolare

Dimostrazione: Supponiamo per assurdo che L sia regolare. Per il Pumping Lemma, esiste un intero $p > 0$ tale che ogni stringa $s \in L$ con $|s| \geq p$ può essere decomposta come $s = xyz$ con $|xy| \leq p$, $|y| > 0$ e $xy^i z \in L$ per ogni $i \geq 0$.

Consideriamo la stringa $s = 0^p 1^p \in L$. Per la condizione $|xy| \leq p$, la sottostringa xy è composta solo da 0. Quindi $y = 0^k$ per qualche $k > 0$.

Consideriamo ora $xy^0 z = xz$. Questa stringa contiene $p - k$ zeri e p uni, quindi non è della forma $0^n 1^n$ e non appartiene a L. Questo contraddice la condizione che $xy^i z \in L$ per ogni $i \geq 0$.

Pertanto, L non è regolare.

6.2 Esempio 2: $L = \{a^i b^j c^k \mid i, j, k \geq 0 \text{ e } i + j = k\}$ è context-free

Dimostrazione: Costruiamo una grammatica context-free $G = (V, \Sigma, R, S)$ che genera L.

- $V = \{S, A\}$
- $\Sigma = \{a, b, c\}$
- R contiene le seguenti regole:
  - $S \to Ac$
  - $A \to aAc \mid bAc \mid \varepsilon$

Per comprendere come questa grammatica genera L, osserviamo che ogni derivazione ha la forma:

$$\begin{aligned} S &\Rightarrow Ac \\ &\Rightarrow aAcc \\ &\Rightarrow abAccc \\ &\Rightarrow \dots \\ &\Rightarrow a^i b^j \varepsilon c^{i+j+1} \\ &= a^i b^j c^{i+j+1} \end{aligned}$$

La derivazione produce una stringa in cui il numero di c è $i + j + 1$, dove i è il numero di a e j è il numero di b. Sia $k = i + j + 1$, quindi $i + j = k - 1$. Possiamo modificare leggermente la grammatica per ottenere $i + j = k$:

- $V = \{S, A\}$

- $\Sigma = \{a, b, c\}$
- R contiene le seguenti regole:
  - $S \to A$
  - $A \to aAc \mid bAc \mid \varepsilon$

Con questa grammatica, ogni derivazione produrrà stringhe della forma $a^i b^j c^{i+j}$, ovvero stringhe in cui il numero di c è uguale alla somma del numero di a e b. Quindi, questa grammatica genera esattamente il linguaggio $L = \{a^i b^j c^k \mid i, j, k \geq 0 \text{ e } i + j = k\}$.

6.3 Esempio 3: dimostrazione che flip(L) è regolare se L è regolare

Definiamo flip$(L) = \{w \in \{0,1\}^* \mid$ il flip di w appartiene a $L\}$, dove il flip di una stringa si ottiene cambiando tutti gli 0 in 1 e tutti gli 1 in 0.

Dimostrazione: Sia $A = (Q, \Sigma, \delta, q_0, F)$ un DFA che riconosce L. Costruiamo un DFA $A' = (Q', \Sigma, \delta', q_0', F')$ che riconosce flip(L) come segue:

- $Q' = Q$ (l'insieme degli stati rimane lo stesso)
- $\Sigma = \{0, 1\}$ (l'alfabeto rimane lo stesso)
- $\delta'(q, 0) = \delta(q, 1)$ e $\delta'(q, 1) = \delta(q, 0)$ per ogni $q \in Q$ (la funzione di transizione scambia gli 0 con gli 1)
- $q_0' = q_0$ (lo stato iniziale non cambia)
- $F' = F$ (gli stati finali non cambiano)

Per dimostrare che A' riconosce flip(L), dobbiamo mostrare che per ogni stringa $w \in \{0,1\}^*$, $w \in$ flip(L) se e solo se w è accettata da A'.

Sia $\overline{w}$ il flip di w.

($\Rightarrow$) Se $w \in$ flip(L), allora $\overline{w} \in L$. Quindi, $\overline{w}$ è accettata da A. Ciò significa che esiste una sequenza di stati $q_0, q_1, \ldots, q_n$ in A tale che $\delta(q_{i-1}, \overline{w}_i) = q_i$ per $i = 1, \ldots, n$, e $q_n \in F$.

Per costruzione di A', abbiamo $\delta'(q_{i-1}, w_i) = \delta(q_{i-1}, \overline{w}_i) = q_i$ per $i = 1, \ldots, n$. Pertanto, w è accettata da A'.

($\Leftarrow$) Se w è accettata da A', allora esiste una sequenza di stati $q_0, q_1, \ldots, q_n$ in A' tale che $\delta'(q_{i-1}, w_i) = q_i$ per $i = 1, \ldots, n$, e $q_n \in F'$.

Per costruzione di A', abbiamo $\delta(q_{i-1}, \overline{w}_i) = \delta'(q_{i-1}, w_i) = q_i$ per $i = 1, \ldots, n$. Poiché $F' = F$, $q_n \in F$. Quindi, $\overline{w}$ è accettata da A, il che significa che $\overline{w} \in L$ e $w \in$ flip(L).

Poiché A' è un DFA, flip(L) è regolare.

6.4 Esempio 4: $L = \{a^n b^m c^m \mid n, m \geq 0\} \cup \{a^n b^n c^m \mid n, m \geq 0\}$ è context-free

Dimostrazione: Osserviamo che $L = L_1 \cup L_2$, dove $L_1 = \{a^n b^m c^m \mid n, m \geq 0\}$ e $L_2 = \{a^n b^n c^m \mid n, m \geq 0\}$.

Dimostriamo che L_1 e L_2 sono entrambi context-free, e quindi la loro unione L è context-free per la proprietà di chiusura.

Per L_1, costruiamo la grammatica $G_1 = (V_1, \Sigma, R_1, S_1)$ dove:

- $V_1 = \{S_1, A, B\}$
- $\Sigma = \{a, b, c\}$
- R_1 contiene le regole:
  - $S_1 \to aS_1 \mid A$
  - $A \to bBc \mid \varepsilon$
  - $B \to bBc \mid \varepsilon$

G_1 genera L_1 perché:

- $S_1 \to aS_1 \ldots \to a^n A$ genera la parte a^n
- $A \to bBc$ inizia la generazione della parte $b^m c^m$
- $B \to bBc \ldots \to b^{m-1} B c^{m-1} \to b^m c^m$ completa la generazione

Per L_2, costruiamo la grammatica $G_2 = (V_2, \Sigma, R_2, S_2)$ dove:

- $V_2 = \{S_2, C, D\}$
- $\Sigma = \{a, b, c\}$
- R_2 contiene le regole:
  - $S_2 \to aC_2b \mid D$
  - $C_2 \to aC_2b \mid \varepsilon$
  - $D \to cD \mid \varepsilon$

G_2 genera L_2 perché:

- $S_2 \to aC_2b \ldots \to a^n C_2 b^n \to a^n b^n$ genera la parte $a^n b^n$
- $S_2 \to D \to cD \ldots \to c^m D \to c^m$ genera la parte c^m

Ora, costruiamo una grammatica $G = (V, \Sigma, R, S)$ per $L = L_1 \cup L_2$:

- $V = V_1 \cup V_2 \cup \{S\}$
- $\Sigma = \{a, b, c\}$
- R contiene tutte le regole di R_1 e R_2, più le regole:
  - $S \to S_1 \mid S_2$

Questa grammatica genera $L = L_1 \cup L_2$, che è quindi context-free.

7 Tecniche specifiche per le grammatiche context-free generalizzate

Le grammatiche context-free generalizzate consentono di avere espressioni regolari sul lato destro delle regole di produzione. Vediamo come dimostrare che queste grammatiche non aumentano il potere espressivo delle grammatiche context-free normali.

7.1 Teorema di equivalenza

Ogni grammatica context-free generalizzata può essere convertita in una grammatica context-free normale che genera lo stesso linguaggio.

Dimostrazione strutturale: Mostreremo come rimpiazzare le regole con espressioni regolari sul lato destro con regole equivalenti in forma standard.

Consideriamo i seguenti casi:

1. Rimpiazza ogni regola $A \to R + S$ con le due regole $A \to R$ e $A \to S$
2. Per ogni regola $A \to R \cdot S$, aggiungi due nuove variabili A_R e A_S e rimpiazza la regola con le regole $A \to A_R A_S$, $A_R \to R$ e $A_S \to S$
3. Per ogni regola $A \to S^*$, aggiungi una nuova variabile A_S e rimpiazza la regola con le regole $A \to AA_S \mid \varepsilon$ e $A_S \to S$
4. Rimpiazza ogni regola $A \to \emptyset$ con nessuna regola (rimuovi la regola)
5. Rimpiazza ogni regola $A \to \varepsilon$ con la regola standard $A \to \varepsilon$

Ripeti questo processo finché non rimangono solamente regole nella forma standard $A \to u$ dove u è una stringa di variabili e terminali, o $u = \varepsilon$.

7.2 Esempi di applicazione

Esempio 1: Consideriamo la grammatica generalizzata con regole:

- $S \to aS \mid aSbS \mid \varepsilon$

Esempio 2: Consideriamo la grammatica generalizzata con regole:

- $S \to (S) \mid SS \mid \varepsilon$

Entrambe queste grammatiche sono già in forma standard perché non contengono espressioni regolari sul lato destro.

Esempio 3: Consideriamo la grammatica generalizzata con regole:

- $S \to a(b + c)^*$

Trasformiamola in una grammatica in forma standard:

- $S \to aA$
- $A \to BA \mid \varepsilon$
- $B \to b \mid c$

8 Il Pumping Lemma per linguaggi context-free

Analogamente al caso dei linguaggi regolari, esiste un Pumping Lemma anche per i linguaggi context-free, che può essere utilizzato per dimostrare che un linguaggio non è context-free.

8.1 Enunciato del Pumping Lemma per linguaggi context-free

Teorema (Pumping Lemma per linguaggi context-free): Per ogni linguaggio context-free L, esiste un intero $p > 0$ (la "lunghezza di pumping") tale che, per ogni stringa $s \in L$ con $|s| \geq p$, s può essere suddivisa in cinque parti $s = uvxyz$ con le seguenti proprietà:

1. $|vxy| \leq p$
2. $|vy| > 0$
3. Per ogni $i \geq 0$, $uv^ixy^iz \in L$

8.2 Schema generale per la dimostrazione per contraddizione

1. Assumere per assurdo che il linguaggio L sia context-free
2. Applicare il Pumping Lemma: esiste un $p > 0$ con le proprietà indicate
3. Scegliere una stringa $s \in L$ con $|s| \geq p$ opportunamente costruita
4. Mostrare che per ogni divisione $s = uvxyz$ che soddisfa le condizioni 1 e 2 del lemma, esiste un $i \geq 0$ tale che $uv^ixy^iz \notin L$, contraddicendo la condizione 3
5. Concludere che L non è context-free

8.3 Esempio: $L = \{a^n b^n c^n \mid n \geq 0\}$ non è context-free

Dimostrazione: Supponiamo per assurdo che L sia context-free. Allora esiste un intero $p > 0$ come nel Pumping Lemma. Consideriamo la stringa $s = a^p b^p c^p \in L$. Per il Pumping Lemma, s può essere scritta come $s = uvxyz$ con $|vxy| \leq p$, $|vy| > 0$ e $uv^i xy^i z \in L$ per ogni $i \geq 0$.

Dato che $|vxy| \leq p$, la sottostringa vxy può contenere al massimo p caratteri, quindi può contenere caratteri di al massimo due tipi diversi (ad esempio, a e b, o b e c), ma non tutti e tre.

Consideriamo i vari casi:

1. Se vxy contiene solo a, allora v e y contengono solo a. Pertanto, uv^2xy^2z conterrà più di p caratteri a, ma esattamente p caratteri b e p caratteri c. Quindi, $uv^2xy^2z \notin L$.

2. Se vxy contiene solo b, allora v e y contengono solo b. Pertanto, uv^2xy^2z conterrà più di p caratteri b, ma esattamente p caratteri a e p caratteri c. Quindi, $uv^2xy^2z \notin L$.

3. Se vxy contiene solo c, allora v e y contengono solo c. Pertanto, uv^2xy^2z conterrà più di p caratteri c, ma esattamente p caratteri a e p caratteri b. Quindi, $uv^2xy^2z \notin L$.

4. Se vxy contiene a e b, allora almeno uno tra v e y contiene a o b. In $uv^0xy^0z = uxz$, il numero di a sarà diverso dal numero di b, che saranno diversi dal numero di c. Quindi, $uxz \notin L$.

5. Se vxy contiene b e c, il ragionamento è analogo al caso precedente.

In tutti i casi, troviamo un i (0 o 2) tale che $uv^i xy^i z \notin L$, contraddicendo il Pumping Lemma. Pertanto, L non è context-free.

9 Tecniche di dimostrazione per casi particolari

9.1 Linguaggi con operatori aritmetici

Molti linguaggi includono relazioni aritmetiche tra i conteggi dei simboli. Vediamo alcune tecniche specifiche per questi casi.

Esempio: $L = \{a^n b^m \mid n \leq m \leq 2n\}$ Dimostriamo che L è context-free.

Soluzione: Osserviamo che $L = \{a^n b^n (b?)^n \mid n \geq 0\}$, dove $(b?)^n$ significa che possiamo avere da 0 a n simboli b aggiuntivi. Costruiamo una grammatica context-free:

- $S \to AB$
- $A \to aAb \mid \varepsilon$

- $B \to bB \mid \varepsilon$

Questa grammatica genera $a^n b^n$ usando A, e poi può aggiungere fino a n simboli b aggiuntivi usando B.

9.2 Linguaggi con palindromi e altri pattern

I linguaggi che coinvolgono palindromi o pattern simili spesso richiedono tecniche specifiche.

Esempio: $L = \{ww^R \mid w \in \{a, b\}^*\}$**, dove** w^R **è l'inverso di** w Dimostriamo che L è context-free.

Soluzione: Costruiamo una grammatica context-free:

- $S \to \varepsilon \mid aSa \mid bSb$

Questa grammatica genera palindromi, che hanno la forma ww^R solo quando $w = w^R$ (cioè quando w stesso è un palindromo). Per generare ww^R per qualsiasi w, abbiamo bisogno di una grammatica diversa. Ad esempio:

- $S \to aSa \mid bSb \mid C$
- $C \to aCa \mid bCb \mid \varepsilon$

Questa grammatica non è corretta. La grammatica corretta per $L = \{ww^R \mid w \in \{a, b\}^*\}$ è:

- $S \to aSa \mid bSb \mid \varepsilon$

Per $L = \{ww \mid w \in \{a, b\}^*\}$ (che è diverso da $\{ww^R \mid w \in \{a, b\}^*\}$), possiamo dimostrare che non è context-free usando il Pumping Lemma.

9.3 Linguaggi che codificano problemi di decisione

Alcuni linguaggi codificano problemi di decisione complessi. Ad esempio, il linguaggio delle stringhe che rappresentano numeri primi, o il linguaggio delle stringhe che rappresentano grammatiche ambigue.

Esempio: $L = \{a^n \mid n$ **è un numero primo**$\}$ Questo linguaggio è regolare? Dimostriamo che non lo è.

Soluzione: Supponiamo per assurdo che L sia regolare. Per il Pumping Lemma, esiste un intero $p > 0$ tale che ogni stringa $s \in L$ con $|s| \geq p$ può essere decomposta come $s = xyz$ con $|xy| \leq p$, $|y| > 0$ e $xy^i z \in L$ per ogni $i \geq 0$.

Sia q un numero primo maggiore di p. Allora $s = a^q \in L$. Per il Pumping Lemma, s può essere scritta come $s = xyz$ con $|xy| \leq p$, $|y| > 0$ e $xy^i z \in L$ per ogni $i \geq 0$.

Poiché $|xy| \leq p < q$, abbiamo $|y| = k > 0$ e $|z| = q - |xy| > 0$. Consideriamo xy^2z. Questa stringa ha lunghezza $|x|+2|y|+|z| = |x|+|y|+|y|+|z| = q+|y| = q+k$.

Ma $q + k$ non è necessariamente un numero primo, quindi xy^2z non è necessariamente in L, contraddicendo il Pumping Lemma. Pertanto, L non è regolare.