

Linguaggi Formali e Traduttori

3.1 Grammatiche libere dal contesto

- Sommario
- Grammatiche libere dal contesto
- Derivazioni
- Esempio
- Linguaggio generato da una grammatica
- Grammatiche e linguaggi regolari
- Esempio: stringhe della forma $a^n b^n$
- Esempio: stringhe della forma $a^m b^n$
- Esempio: espressioni aritmetiche
- Esempio: comando di assegnamento in Java
- Esercizi

È proibito condividere e divulgare in qualsiasi forma i materiali didattici caricati sulla piattaforma e le lezioni svolte in videoconferenza: ogni azione che viola questa norma sarà denunciata agli organi di Ateneo e perseguita a termini di legge.

Sommario

Motivazione

- Il linguaggio $L = \{a^n b^n \mid n \in \mathbb{N}\}$ non è regolare.
- Ponendo $a = ($ e $b =)$, notiamo che L è il linguaggio delle parentesi bilanciate.
- Linguaggi come L sono di fondamentale importanza per descrivere la struttura di espressioni e blocchi nei linguaggi di programmazione.

In questa lezione

- Studiamo un approccio generativo – le **grammatiche libere** – per la descrizione di **linguaggi liberi**.
- Definiamo grammatiche per generare alcuni linguaggi liberi, tra cui L .
- Mostriamo che i linguaggi liberi includono tutti quelli regolari.

Grammatiche libere dal contesto

Definizione

Una **grammatica libera dal contesto** (o semplicemente **grammatica libera**) è una quadrupla $G = (V, T, P, S)$ dove:

- V è un insieme finito di **variabili** (o **simboli non terminali**, o **categorie sintattiche**).
- T è un alfabeto di simboli **terminali**.
- P è un insieme finito di **produzioni** della forma $A \rightarrow \alpha$, in cui:
 - $A \in V$ è detta **testa** della produzione;
 - $\alpha \in (V \cup T)^*$ è detta **corpo** della produzione.
- $S \in V$ è il **simbolo iniziale** della grammatica.

Convenzioni

- Le lettere a, b, c, \dots denotano simboli terminali (elementi di T).
- Le lettere A, B, C, \dots denotano variabili (elementi di V).
- Le lettere X, Y, Z, \dots denotano simboli (elementi di $V \cup T$).
- Le lettere u, v, w, \dots denotano stringhe di simboli terminali (elementi di T^*).
- Le lettere $\alpha, \beta, \gamma, \dots$ denotano stringhe di simboli (elementi di $(V \cup T)^*$).
- Abbreviamo $A \rightarrow \alpha_1, \dots, A \rightarrow \alpha_n$ con $A \rightarrow \alpha_1 \mid \dots \mid \alpha_n$

Derivazioni

Definizione

Fissata una grammatica $G = (V, T, P, S)$, definiamo le derivazioni in un passo e in zero o più passi come segue:

- scriviamo $\alpha A \beta \Rightarrow_G \alpha \gamma \beta$ se $A \rightarrow \gamma \in P$.

In tal caso diciamo che $\alpha \gamma \beta$ deriva in un passo da $\alpha A \beta$ in G .

- scriviamo \Rightarrow_G^* per la chiusura riflessiva e transitiva di \Rightarrow_G , ovvero:
 - $\alpha \Rightarrow_G^* \alpha$
 - se $\alpha \Rightarrow_G \beta$ e $\beta \Rightarrow_G^* \gamma$, allora $\alpha \Rightarrow_G^* \gamma$

Quando $\alpha \Rightarrow_G^* \beta$ diciamo che β deriva in zero o più passi da α in G .

Convenzione

Omettiamo G da \Rightarrow_G e \Rightarrow_G^* quando è chiaro a quale grammatica ci si riferisce.

Esempio

Data la grammatica

$$G = (\{A\}, \{0, 1\}, \{A \rightarrow \epsilon \mid 0 \mid 1 \mid 0A0 \mid 1A1\}, A)$$

abbiamo le seguenti derivazioni:

- $A \Rightarrow \epsilon$
- $A \Rightarrow 0$
- $A \Rightarrow 1$
- $A \Rightarrow 0A0 \Rightarrow 00$
- $A \Rightarrow 1A1 \Rightarrow 101$
- $A \Rightarrow 0A0 \Rightarrow 01A10 \Rightarrow 011A110 \Rightarrow 0110110$

Note

- La **variabile A** indica una stringa palindroma arbitraria
- Le produzioni ci permettono di **riscrivere A** in una stringa palindroma più specifica
- Riscrivendo A ottengo tutte e sole le stringhe w palindrome ($w = w^R$)

Linguaggio generato da una grammatica

Definizione

Data una grammatica $G = (V, T, P, S)$, il linguaggio generato da G , denotato da $L(G)$ è definito come

$$L(G) \stackrel{\text{def}}{=} \{w \in T^* \mid S \Rightarrow_G^* w\}$$

Esempio

Per la grammatica G della slide precedente abbiamo

$$L(G) = \{w \in \{0, 1\}^* \mid w = w^R\}$$

Definizione

Diciamo che L è un **linguaggio libero** se esiste una grammatica libera che lo genera.

Grammatiche e linguaggi regolari

Teorema

Per ogni linguaggio regolare L esiste una grammatica G tale che $L(G) = L$.

Dimostrazione

Sia $A = (Q, \Sigma, \delta, q_0, F)$ un DFA che riconosce L .

Definiamo la grammatica $G = (Q, \Sigma, P, q_0)$ dove P è così definito:

- se $q \in Q$ e $a \in \Sigma$, allora $q \rightarrow a\delta(q, a) \in P$
- se $q \in F$, allora $q \rightarrow \epsilon \in P$

È facile vedere che $q_0 \Rightarrow^* w \Leftrightarrow \hat{\delta}(q_0, w) \in F$ da cui segue il risultato.

Osservazioni

- Il teorema mostra che le grammatiche possono generare tutti i linguaggi regolari.
- Sappiamo che le grammatiche possono generare linguaggi non regolari (come quello delle stringhe palindrome).
- I linguaggi liberi includono propriamente i linguaggi regolari.

Esempio: stringhe della forma $a^n b^n$

Si consideri la grammatica

$$(\{S\}, \{a, b\}, P, S)$$

in cui P è l'insieme di produzioni

- $S \rightarrow \epsilon$
- $S \rightarrow aSb$

Alcune derivazioni

- $S \Rightarrow \epsilon$
- $S \Rightarrow aSb \Rightarrow ab$
- $S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aabb$
- $S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb \Rightarrow aaabbb$

In generale

- $S \Rightarrow^* a^n b^n$ per ogni $n \in \mathbb{N}$

Esempio: stringhe della forma $a^m b^n$

Si consideri la grammatica

$$(\{S, A, B\}, \{a, b\}, P, S)$$

in cui P è l'insieme di produzioni:

- $S \rightarrow AB$
- $A \rightarrow \epsilon \mid aA$
- $B \rightarrow \epsilon \mid bB$

Alcune derivazioni

- $S \Rightarrow AB \Rightarrow A \Rightarrow \epsilon$
- $S \Rightarrow AB \Rightarrow aAB \Rightarrow aAbB \Rightarrow abB \Rightarrow ab$
- $S \Rightarrow AB \Rightarrow aAB \Rightarrow aaAB \Rightarrow aaaAB \Rightarrow aaaB \Rightarrow aaabB \Rightarrow aaab$

In generale

- $S \Rightarrow^* a^m b^n$ per ogni $m, n \in \mathbb{N}$

Esempio: espressioni aritmetiche

Si consideri la grammatica

$$(\{E\}, \{x, y, +, *, (,)\}, P, E)$$

in cui P è l'insieme di produzioni

- $E \rightarrow x$
- $E \rightarrow y$
- $E \rightarrow E + E$
- $E \rightarrow E * E$
- $E \rightarrow (E)$

Alcune derivazioni

- $E \Rightarrow E + E \Rightarrow x + E \Rightarrow x + y$
- $E \Rightarrow E + E \Rightarrow x + E \Rightarrow x + E * E \Rightarrow x + y * E \Rightarrow x + y * y$
- $E \Rightarrow E * E \Rightarrow (E) * E \Rightarrow (E + E) * E$
 $\Rightarrow (x + E) * E \Rightarrow (x + y) * E \Rightarrow (x + y) * y$

Esempio: comando di assegnamento in Java

Si consideri la grammatica

$$(\{S, R, E\}, \{=, [,], c, x\}, P, S)$$

in cui P è l'insieme di produzioni

- $S \rightarrow R = E$
- $R \rightarrow x \mid R[E]$
- $E \rightarrow c \mid R$

Alcune derivazioni

- $S \Rightarrow R = E \Rightarrow x = E \Rightarrow x = c$
- $S \Rightarrow R = E \Rightarrow R = R \Rightarrow x = R \Rightarrow x = x$
- $S \Rightarrow R = E \Rightarrow R[E] = E \Rightarrow x[E] = E \Rightarrow x[c] = E \Rightarrow x[c] = c$

Alcune stringhe non derivabili

- $S \not\Rightarrow^* x$
- $S \not\Rightarrow^* c = x$

Esercizi

Sulla definizione di grammatiche

Definire grammatiche per generare i seguenti linguaggi:

1. Il linguaggio generato da $(ab)^*$.
2. Le stringhe di parentesi quadre bilanciate (es. $[[[]]]$).
3. Le stringhe di 0 e 1 della forma ww^R .
4. Stringhe di 0 e 1 in cui c'è lo stesso numero di 0 e 1 (es. 00100111).
5. Espressioni booleane composte dalle costanti t (true), f (false) e i connettivi \wedge (congiunzione), \vee (disgiunzione) e \neg (negazione). Ammettere la possibilità di usare parentesi (es. $t \wedge (f \vee \neg t)$ e $\neg(t \vee f)$).

Sul linguaggio generato da una grammatica

Descrivere il linguaggio generato dalle seguenti grammatiche:

1. $(\{S\}, \{a, b\}, \{S \rightarrow \epsilon \mid aaSb\}, S)$
2. $(\{S, A\}, \{a, b\}, \{S \rightarrow \epsilon \mid ASb, A \rightarrow \epsilon \mid a\}, S)$
3. $(\{S, X, C\}, \{a, b, c\}, \{S \rightarrow XC, X \rightarrow \epsilon \mid aXb, C \rightarrow \epsilon \mid cC\}, S)$
4. $(\{S\}, \{a, b\}, \{S \rightarrow \epsilon \mid aSb \mid bSa\}, S)$

Linguaggi Formali e Traduttori

3.2 Alberi sintattici, derivazioni canoniche e ambiguità

- Sommario
- Alberi sintattici
- Esempio
- Grammatiche ambigue
- Ambiguità e derivazioni
- Derivazioni canoniche
- Esempio
- Esercizi

È proibito condividere e divulgare in qualsiasi forma i materiali didattici caricati sulla piattaforma e le lezioni svolte in videoconferenza: ogni azione che viola questa norma sarà denunciata agli organi di Ateneo e perseguita a termini di legge.

Sommario

Problema

- Per tradurre un programma è importante riconoscerne la **struttura**, che è codificata nella derivazione che genera quel programma.
- Tuttavia, possono esserci derivazioni differenti che generano lo stesso programma, anche imponendo certe discipline sull'ordine delle riscritture nelle derivazioni.

In questa lezione

- Definiamo gli **alberi sintattici** come alternativa alle derivazioni per ragionare sulla struttura delle stringhe generate da una grammatica astraendo dall'**ordine** delle riscritture.
- Mostriamo che certe grammatiche sono **ambigue**, in quanto ammettono alberi sintattici diversi.
- Definiamo due forme **canoniche** di derivazione, a **sinistra** e a **destra**, per disciplinare l'ordine delle riscritture e caratterizzare l'ambiguità di una grammatica (anche) in termini di derivazioni canoniche.

Alberi sintattici

Definizione

Data una grammatica $G = (V, T, P, S)$, gli **alberi sintattici** di G sono alberi con le seguenti caratteristiche:

- Ogni nodo interno (diverso da una foglia) è etichettato con una variabile in V .
- Ogni foglia è etichettata con una variabile in V , o da un terminale in T , o da ϵ .
- Se una foglia è etichettata con ϵ , è anche l'unico figlio del suo genitore.
- Se un nodo interno è etichettato con A e i suoi figli sono etichettati (da sinistra a destra) con X_1, X_2, \dots, X_n , allora $A \rightarrow X_1 X_2 \cdots X_n$ è una produzione in P .

Definizione

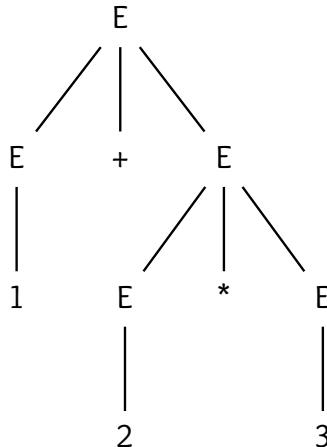
Il **prodotto** di un albero sintattico è la stringa ottenuta concatenando, da sinistra verso destra, le etichette di tutte le foglie dell'albero.

Teorema

$A \Rightarrow_G^* \alpha$ se e solo se esiste un albero sintattico di G con radice A e prodotto α .

Esempio

$$G = (\{E\}, \{0, 1, \dots, 9, +, *, (,)\}, \{E \rightarrow 0 \mid \dots \mid 9 \mid E + E \mid E * E \mid (E)\}, E)$$



$$\begin{aligned} E &\Rightarrow E + E \\ &\Rightarrow 1 + E \\ &\Rightarrow 1 + E * E \\ &\Rightarrow 1 + 2 * E \\ &\Rightarrow 1 + 2 * 3 \end{aligned}$$

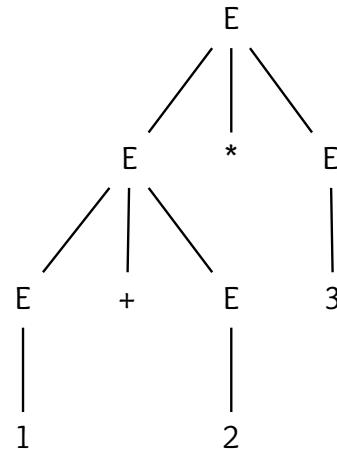
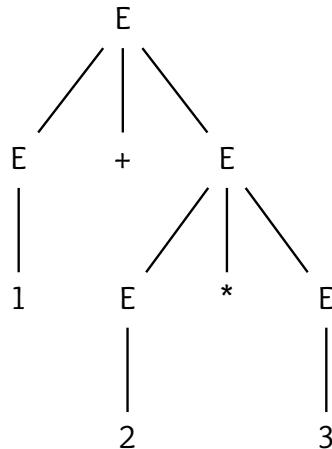
Grammatiche ambigue

Definizione

Una grammatica è **ambigua** se ammette più alberi sintattici distinti con lo stesso prodotto.

Esempio

La grammatica delle **espressioni in forma infissa** è ambigua.



- Questi alberi sottintendono due significati molto diversi per l'espressione $1 + 2 * 3$ che certamente vorremmo tradurre in modi diversi

Ambiguità e derivazioni

Attenzione

- Basta trovare **due** alberi sintattici **distinti** con lo stesso prodotto per dire che una grammatica ambigua
- **Non** basta trovare due derivazioni distinte che generano la stessa stringa per dire che una grammatica ambigua

Esempio

- $E \Rightarrow E + E \Rightarrow \underline{1 + E} \Rightarrow 1 + 2$
- $E \Rightarrow E + E \Rightarrow \underline{E + 2} \Rightarrow 1 + 2$

Queste due derivazioni sono evidentemente distinte (si noti la parte sottolineata), hanno lo stesso prodotto (**1 + 2**), ma corrispondono allo stesso albero sintattico!

Osservazioni

- L'ordine in cui vengono riscritte variabili diverse è irrilevante (esempio in questa slide)
- Quello che conta è se la stessa variabile viene riscritta in modi diversi ([slide precedente](#))
- Imponendo un particolare ordine di riscrittura delle variabili possiamo individuare l'ambiguità guardando le derivazioni

Derivazioni canoniche

Definizione

Una derivazione $X \Rightarrow^* \alpha$ si dice **derivazione a sinistra** se (a ogni passo) viene riscritta la variabile che si trova più a sinistra. Useremo \Rightarrow_{lm} e \Rightarrow_{lm}^* come simboli per derivazioni a sinistra, dove lm abbrevia leftmost.

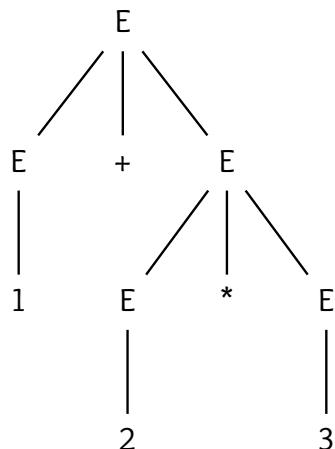
In maniera speculare si definiscono le **derivazioni a destra** e la notazione associata \Rightarrow_{rm} e \Rightarrow_{rm}^* , dove rm abbrevia rightmost.

Proposizione

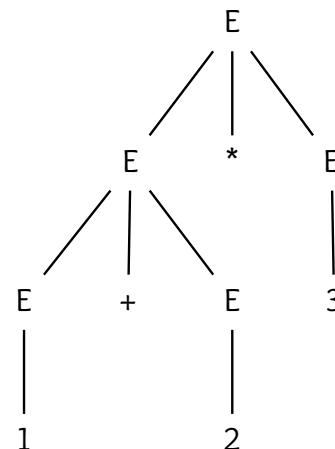
Se esistono due derivazioni canoniche distinte di G (entrambe leftmost o entrambe rightmost) per derivare la stessa stringa allora G è ambigua

Esempio

$$\begin{array}{ll} E & \Rightarrow_{lm} E + E \\ & \Rightarrow_{lm} 1 + E \\ & \Rightarrow_{lm} 1 + E * E \\ & \Rightarrow_{lm} 1 + 2 * E \\ & \Rightarrow_{lm} 1 + 2 * 3 \end{array}$$



$$\begin{array}{ll} E & \Rightarrow_{lm} E * E \\ & \Rightarrow_{lm} E + E * E \\ & \Rightarrow_{lm} 1 + E * E \\ & \Rightarrow_{lm} 1 + 2 * E \\ & \Rightarrow_{lm} 1 + 2 * 3 \end{array}$$



Esercizi

Mostrare che le seguenti grammatiche sono ambigue

1. $S \rightarrow aS \mid aSbS \mid \epsilon$
2. $B \rightarrow t \mid f \mid B \wedge B \mid B \vee B \mid \neg B \mid (B)$

trovando per ciascuna:

- due alberi sintattici distinti con lo stesso prodotto;
- due derivazioni canoniche a sinistra distinte per la stessa stringa;
- due derivazioni canoniche a destra distinte per la stessa stringa.

Linguaggi Formali e Traduttori

3.3 Eliminazione dell'ambiguità

- Sommario
- Ambiguità delle espressioni in forma infissa
- Eliminazione dell'ambiguità delle espressioni
- Esempio di derivazione (1/2)
- Esempio di derivazione (2/2)
- Un linguaggio inerentemente ambiguo
- Esercizi

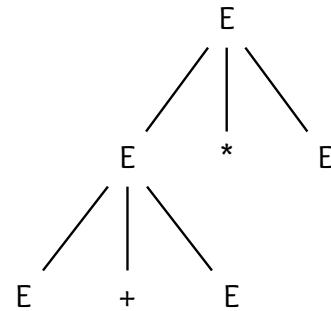
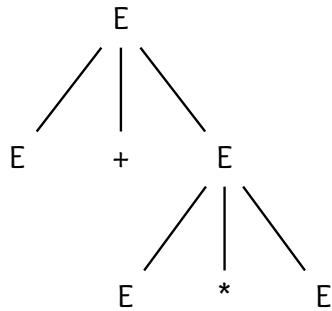
È proibito condividere e divulgare in qualsiasi forma i materiali didattici caricati sulla piattaforma e le lezioni svolte in videoconferenza: ogni azione che viola questa norma sarà denunciata agli organi di Ateneo e perseguita a termini di legge.

Sommario

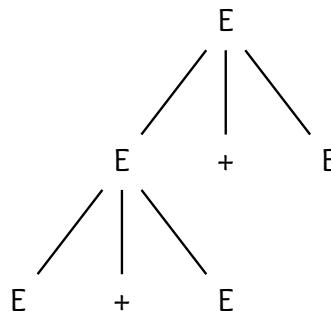
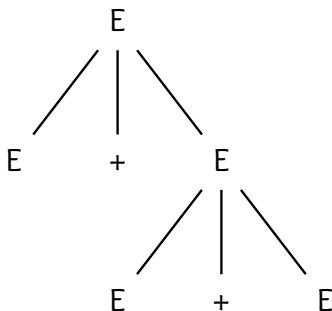
- Mostriamo tecniche per eliminare l'ambiguità da una grammatica in alcuni casi
- Mostriamo un esempio di linguaggio inerentemente ambiguo

Ambiguità delle espressioni in forma infissa

Precedenza degli operatori



Associatività degli operatori



Eliminazione dell'ambiguità delle espressioni

Strategia

- Si “stratificano” e “sbilanciano” le espressioni
- Espressione = somma (associativa a sinistra) di termini
- Termine = prodotto (associativo a sinistra) di fattori
- Fattore = costante o espressione tra parentesi

La grammatica delle espressioni modificate

$$(\{E, T, F\}, \{0, 1, \dots, 9, +, *, (,)\}, P, E)$$

dove P è l'insieme delle seguenti produzioni:

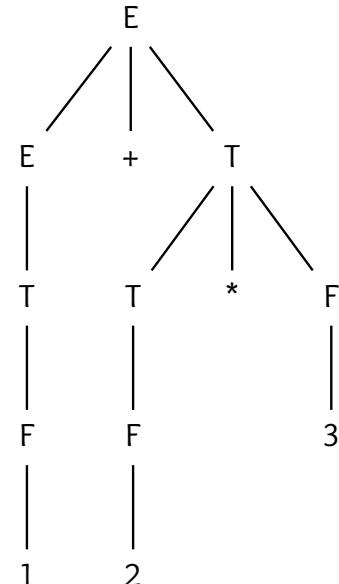
- $E \rightarrow T \mid E + T$
- $T \rightarrow F \mid T * F$
- $F \rightarrow 0 \mid 1 \mid \dots \mid 9 \mid (E)$

Note

- Quella proposta è una modifica ad hoc per la grammatica.
- L'eliminazione dell'ambiguità (laddove possibile) va pianificata per ogni grammatica.

Esempio di derivazione (1/2)

$$\begin{array}{lcl} E & \Rightarrow & E + T \\ & \Rightarrow & T + T \\ & \Rightarrow & F + T \\ & \Rightarrow & 1 + T \\ & \Rightarrow & 1 + T * F \\ & \Rightarrow & 1 + F * F \\ & \Rightarrow & 1 + 2 * F \\ & \Rightarrow & 1 + 2 * 3 \end{array}$$

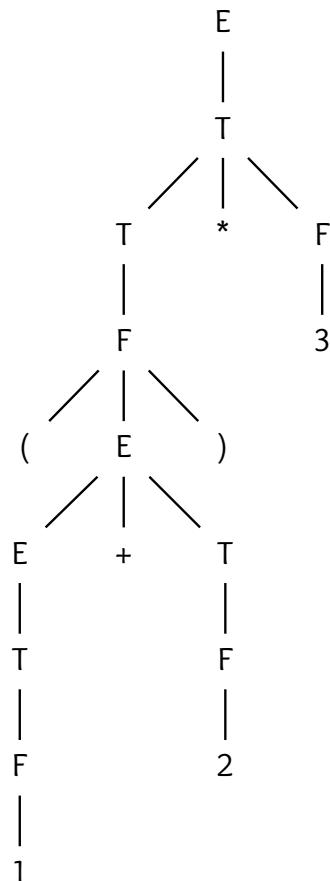


Nota

Anche nella grammatica modificata, l'espressione la cui struttura comporta il calcolo della moltiplicazione prima della somma è generabile senza l'uso di parentesi, in quanto la convenzione abituale dà precedenza alla moltiplicazione rispetto alla somma.

Esempio di derivazione (2/2)

$$\begin{aligned} E &\Rightarrow T \\ &\Rightarrow T * F \\ &\Rightarrow F * F \\ &\Rightarrow (E) * F \\ &\Rightarrow (E + T) * F \\ &\Rightarrow (T + T) * F \\ &\Rightarrow (F + T) * F \\ &\Rightarrow (1 + F) * F \\ &\Rightarrow (1 + 2) * F \\ &\Rightarrow (1 + 2) * 3 \end{aligned}$$



Nota

Nella grammatica modificata, l'espressione la cui struttura comporta il calcolo della somma prima della moltiplicazione è generabile solo con parentesi.

Un linguaggio inherentemente ambiguo

Non esiste alcuna grammatica non ambigua che generi il linguaggio

$$L = \{a^n b^n c^m d^m \mid n \geq 1, m \geq 1\} \cup \{a^n b^m c^m d^n \mid n \geq 1, m \geq 1\}$$

pertanto L si definisce **inherentemente ambiguo**.

Intuizione

In ogni grammatica che genera L ci sono sempre almeno due derivazioni canoniche distinte che generano una stringa della forma $a^n b^n c^n d^n$.

Esempio

- $S \rightarrow AB \mid C$
- $A \rightarrow aAb \mid ab$
- $B \rightarrow cBd \mid cd$
- $C \rightarrow aCd \mid aDd$
- $D \rightarrow bDc \mid bc$

$$\begin{array}{lll} S & \Rightarrow_{lm} & AB \\ & & \Rightarrow_{lm} aAbB \\ & & \Rightarrow_{lm} aabbB \\ & & \Rightarrow_{lm} aabbcBd \\ & & \Rightarrow_{lm} aabbccdd \end{array}$$

$$\begin{array}{lll} S & \Rightarrow_{lm} & C \\ & & \Rightarrow_{lm} aCd \\ & & \Rightarrow_{lm} aaDdd \\ & & \Rightarrow_{lm} aabDcdd \\ & & \Rightarrow_{lm} aabbccdd \end{array}$$

Esercizi

Data la grammatica

$$B \rightarrow t \mid f \mid B \wedge B \mid B \vee B \mid \neg B \mid (B)$$

delle espressioni booleane, risolvere i seguenti esercizi:

1. Modificare la grammatica per eliminarne l'ambiguità, dando ai connettivi la precedenza $\vee < \wedge < \neg$ e l'associatività a destra per \vee e \wedge .
2. Usando la grammatica non ambigua ottenuta nell'esercizio precedente, mostrare le derivazioni a sinistra, a destra e gli alberi sintattici relativi alle espressioni $t \wedge f \vee t$ e $\neg t \wedge f$.

Linguaggi Formali e Traduttori

3.4 Automi a pila

- Sommario
- Esempio informale
- Automi a pila
- Esempio: riconoscitore di stringhe $a^n b^n$
- Descrizioni istantanee
- Mosse di un automa a pila
- Esempio
- Linguaggio accettato da un automa a pila
- Esempio: riconoscitore di stringhe $w w^R$
- Esercizi

È proibito condividere e divulgare in qualsiasi forma i materiali didattici caricati sulla piattaforma e le lezioni svolte in videoconferenza: ogni azione che viola questa norma sarà denunciata agli organi di Ateneo e perseguita a termini di legge.

Sommario

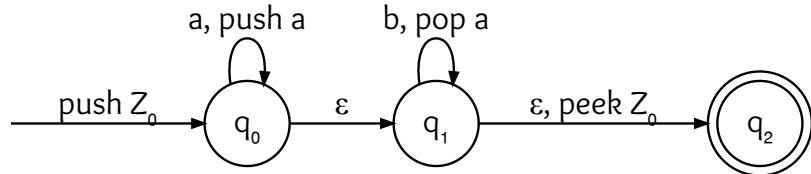
Motivazione

- Le grammatiche libere forniscono un approccio generativo per la descrizione di linguaggi liberi.

In questa lezione

- Studiamo un approccio riconoscitivo – gli **automi a pila** – per la descrizione di linguaggi liberi.
- Definiamo due nozioni di linguaggio riconosciuto da un automa a pila, il linguaggio riconosciuto **per stato finale** ed il linguaggio riconosciuto **per pila vuota**.
- Mostriamo che le due nozioni sono equivalenti.

Esempio informale



Inizializzazione

- La pila contiene un unico simbolo Z_0 usato come “sentinella” (Z_0 = la pila finisce qui).

Stato q_0 : conteggio delle a .

- L'automa accumula sulla pila le a .
- L'automa può “scommettere” di aver letto tutte le a e passare a q_1 .

Stato q_1 : conteggio delle b .

- L'automa controlla che, per ogni b della stringa, vi sia una a sulla pila e la rimuove.
- Se l'automa vede la sentinella Z_0 sulla pila deve aver raggiunto la fine della stringa e passa a q_2 .

Stato q_2 : accettazione.

Automi a pila

Definizione

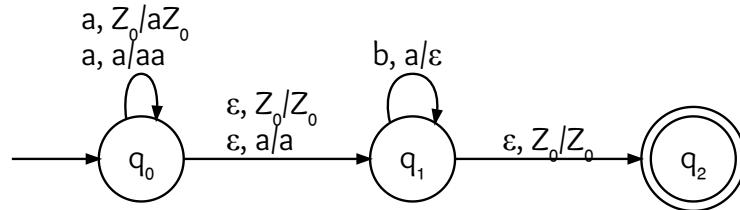
Un **automa a pila** (detto anche **PDA**, da PushDown Automaton) è una settupla $A = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ dove:

- Q è un insieme finito di **stati**
- Σ è l'**alfabeto di input** (simboli che possono comparire nella stringa da riconoscere)
- Γ è l'**alfabeto della pila** (simboli che possono comparire sulla pila)
- $\delta : Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \rightarrow \wp(Q \times \Gamma^*)$ è la **funzione di transizione**
- $q_0 \in Q$ è lo **stato iniziale**
- $Z_0 \in \Gamma$ è il **simbolo iniziale** presente sulla pila all'inizio del riconoscimento
- $F \subseteq Q$ è l'insieme di **stati finali**

Interpretazione di $(p, \gamma) \in \delta(q, \alpha, Z)$

- Quando l'automa si trova nello stato q e il simbolo in cima alla pila è Z ...
- ... l'automa può leggere il simbolo α dalla stringa (o nulla se $\alpha = \epsilon$) ...
- ... spostandosi nello stato p ...
- ... rimuovendo (pop) Z dalla cima della pila ...
- ... e inserendo (push) tutti i simboli γ sulla pila.

Esempio: riconoscitore di stringhe $a^n b^n$



Definiamo il PDA $(\{q_0, q_1, q_2\}, \{a, b\}, \{a, Z_0\}, \delta, q_0, Z_0, \{q_2\})$ dove

| Transizione | = | Etichetta | Azione sulla pila |
|------------------------------|---|-----------------------|-------------------------|
| $\delta(q_0, a, Z_0)$ | = | $\{(q_0, aZ_0)\}$ | $a, Z_0/aZ_0$ push a |
| $\delta(q_0, a, a)$ | = | $\{(q_0, aa)\}$ | $a, a/aa$ push a |
| $\delta(q_0, \epsilon, Z_0)$ | = | $\{(q_1, Z_0)\}$ | $\epsilon, Z_0/Z_0$ – |
| $\delta(q_0, \epsilon, a)$ | = | $\{(q_1, a)\}$ | $\epsilon, a/a$ – |
| $\delta(q_1, b, a)$ | = | $\{(q_1, \epsilon)\}$ | $b, a/\epsilon$ pop a |
| $\delta(q_1, \epsilon, Z_0)$ | = | $\{(q_2, Z_0)\}$ | $\epsilon, Z_0/Z_0$ – |

Descrizioni istantanee

Definizione

Dato un automa a pila $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, una **descrizione istantanea** di P (talvolta abbreviata con D.I.) è una tripla (q, w, α) in cui:

- $q \in Q$ è lo stato in cui si trova l'automa
- $w \in \Sigma^*$ è ciò che rimane da riconoscere della stringa di input
- $\alpha \in \Gamma^*$ è il contenuto della pila dalla cima (sinistra di α) al fondo (destra di α)

Intuizione

La descrizione istantanea è intesa a specificare completamente la configurazione di un automa a pila in un momento durante il processo di riconoscimento di una stringa.

Mosse di un automa a pila

Definizione

Dato un automa a pila $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, definiamo la relazione \vdash_P come segue

$$\begin{aligned}(q, aw, X\beta) &\vdash_P (p, w, \alpha\beta) \quad \text{se } (p, \alpha) \in \delta(q, a, X) \\ (q, w, X\beta) &\vdash_P (p, w, \alpha\beta) \quad \text{se } (p, \alpha) \in \delta(q, \varepsilon, X)\end{aligned}$$

e diciamo che P fa una **mossa** da I a J (dove I e J sono descrizioni istantanee) se $I \vdash_P J$.

Definizione

Scriviamo \vdash_P^* per la chiusura riflessiva e transitiva di \vdash_P . Ovvero, \vdash_P^* è la relazione tale che

- $I \vdash_P^* I$
- se $I \vdash_P K$ e $K \vdash_P^* J$, allora $I \vdash_P^* J$

Convenzione

Scriviamo semplicemente \vdash e \vdash^* laddove l'automa P di riferimento è chiaro dal contesto.

Esempio

Tutte le mosse dell'automa in [slide 5](#) partendo dalla descrizione istantanea $(q_0, aabb, Z_0)$:

$$\begin{array}{l} (q_0, aabb, Z_0) \leftarrow (q_1, aabb, Z_0) \leftarrow (q_2, aabb, Z_0) \\ \quad \top \\ (q_0, abb, aZ_0) \leftarrow (q_1, abb, aZ_0) \\ \quad \top \\ (q_0, bb, aaZ_0) \leftarrow (q_1, bb, aaZ_0) \\ \quad \top \\ (q_1, b, aZ_0) \\ \quad \top \\ (q_1, \varepsilon, Z_0) \leftarrow (q_2, \varepsilon, Z_0) \end{array}$$

Note

- Le mosse “verticali” (\top) corrispondono alla lettura di un simbolo dalla stringa di input.
- Le mosse “orizzontali” (\leftarrow) corrispondono a transizioni spontanee.
- C’è una sequenza di mosse che porta alla consumazione completa dell’input **aabb**.

Linguaggio accettato da un automa a pila

Definizione

Dato $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, il linguaggio accettato da P per stato finale è

$$L(P) \stackrel{\text{def}}{=} \{w \in \Sigma^* \mid (q_0, w, Z_0) \vdash_P^* (q, \varepsilon, \alpha), q \in F\}$$

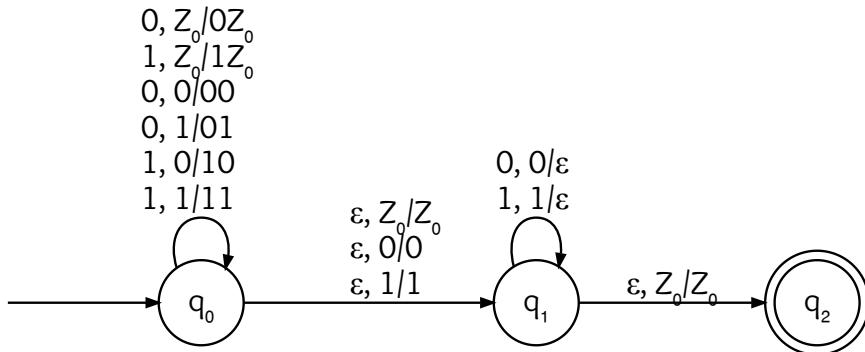
mentre il linguaggio accettato da P per pila vuota è

$$N(P) \stackrel{\text{def}}{=} \{w \in \Sigma^* \mid (q_0, w, Z_0) \vdash_P^* (q, \varepsilon, \varepsilon)\}$$

Note

- Nell'accettazione per stato finale, il contenuto della pila nella D.I. finale è irrilevante.
- Nell'accettazione per pila vuota, lo stato nella D.I. finale può non essere finale.
- In entrambi i casi, la stringa di input deve essere consumata completamente.

Esempio: riconoscitore di stringhe ww^R



Esercizi

1. Definire PDA per riconoscere i seguenti linguaggi. Usare l'accettazione per stato finale o per pila vuota, a seconda di cosa è più conveniente.
 1. $\{a^n b^{2n} \mid n \geq 0\}$
 2. $\{a^{2n} b^n \mid n \geq 0\}$
 3. $\{a^m b^n \mid 0 \leq m \leq n\}$
 4. $\{a^m b^n \mid 0 \leq n \leq m\}$
 5. $\{w2w^R \mid w \in \{0,1\}^*\}$
2. Determinare tutte le mosse possibili dell'automa mostrato in [slide 10](#) a partire dalla D.I. $(q_0, 0110, Z_0)$. Usare uno schema analogo a quello della [slide 8](#).

Linguaggi Formali e Traduttori

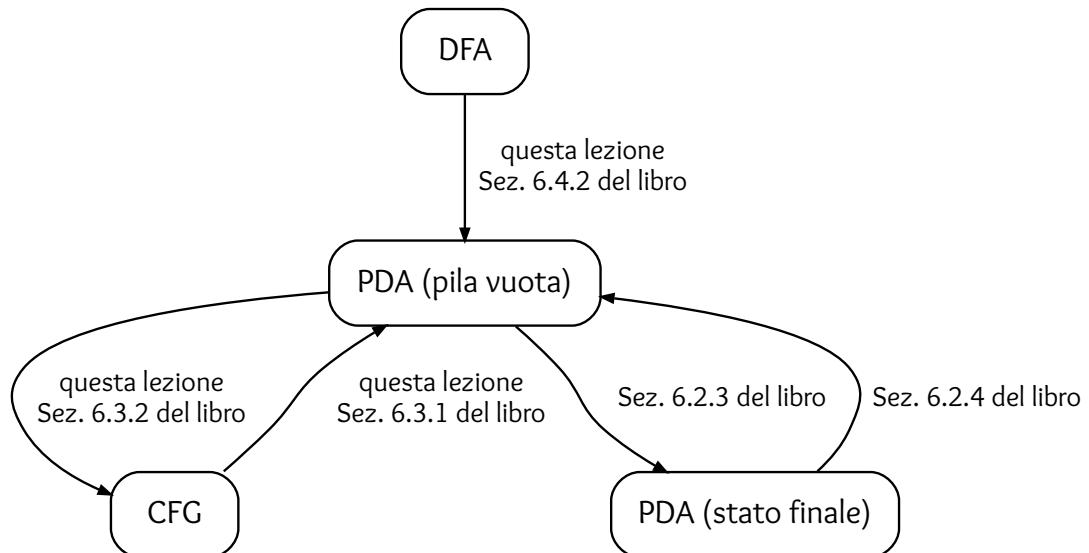
3.5 Relazione tra automi a pila e grammatiche libere

- Sommario
- Relazione tra CFG e PDA
- Osservazione
- Automi a pila deterministici
- Esempio: riconoscitore di stringhe wcw^R
- DPDA e linguaggi regolari
- DPDA e grammatiche ambigue

È proibito condividere e divulgare in qualsiasi forma i materiali didattici caricati sulla piattaforma e le lezioni svolte in videoconferenza: ogni azione che viola questa norma sarà denunciata agli organi di Ateneo e perseguita a termini di legge.

Sommario

- Studiamo la relazione tra CFG e PDA.
- Definiamo una variante **deterministica** dei PDA.
- Mostriamo che i PDA deterministici sono in grado di riconoscere tutti i linguaggi regolari e un sottoinsieme dei linguaggi liberi non inerentemente ambigui.



Relazione tra CFG e PDA

Teorema

1. Per ogni CFG G , esiste un PDA P tale che $N(P) = L(G)$.
2. Per ogni PDA P , esiste una CFG G tale che $L(G) = N(P)$.

Intuizione per 1

Data $G = (V, T, Q, S)$, definiamo un PDA che simuli le derivazioni a sinistra di G . Sia P il PDA $(\{q\}, T, V \cup T, \delta, q, S, \emptyset)$ dove δ è definita come segue:

$$\begin{aligned}\delta(q, \varepsilon, A) &= \{(q, \beta) \mid A \rightarrow \beta \in Q\} && \text{per ogni } A \in V \\ \delta(q, a, a) &= \{(q, \varepsilon)\} && \text{per ogni } a \in T\end{aligned}$$

Per concludere la dimostrazione è sufficiente mostrare che

$$\alpha \Rightarrow_{lm}^* w \Leftrightarrow (q, w, \alpha) \vdash^* (q, \varepsilon, \varepsilon)$$

in quanto, come caso particolare, avremo

$$G \text{ genera } w \Leftrightarrow S \Rightarrow^* w \Leftrightarrow (q, w, S) \vdash^* (q, \varepsilon, \varepsilon) \Leftrightarrow P \text{ accetta } w \text{ per pila vuota}$$

I dettagli della dimostrazione si trovano nel libro di testo.

Osservazione

Richiamando la funzione di transizione definita nella slide precedente:

$$\begin{aligned}\delta(q, \varepsilon, A) &= \{(q, \beta) \mid A \rightarrow \beta \in Q\} \quad \text{per ogni } A \in V \\ \delta(q, a, a) &= \{(q, \varepsilon)\} \quad \text{per ogni } a \in T\end{aligned}$$

- Il PDA che riconosce il linguaggio generato da una CFG fa uso chiave del **non determinismo** per “indovinare” la produzione giusta da usare per riscrivere una variabile A
- I risultati di equivalenza tra PDA e CFG hanno una valenza principalmente teorica, ma non aiutano molto a ottenere riconoscitori efficienti per linguaggi liberi
- Consideriamo PDA **deterministici**

Automi a pila deterministici

Intuizione

Non devono esserci “scelte” possibili a partire dalla stessa D.I.

Definizione

Un **automa a pila deterministico (DPDA)**, da Deterministic PushDown Automaton) è un PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ in cui, per ogni $q \in Q, a \in \Sigma$ e $X \in \Gamma$, l’insieme $\delta(q, a, X) \cup \delta(q, \varepsilon, X)$ contiene al massimo un elemento.

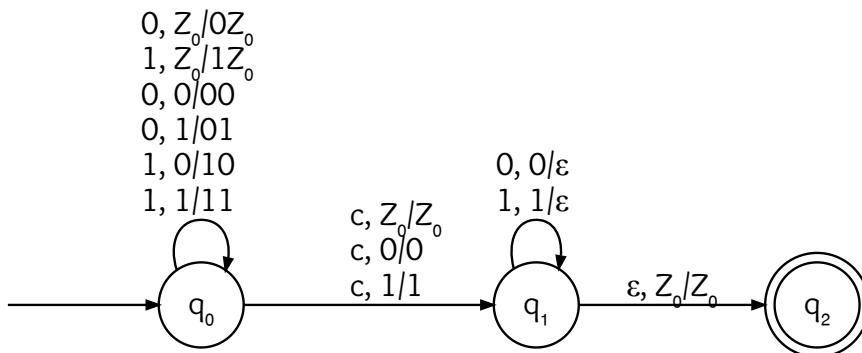
Note

- A parità di stato, simbolo letto e simbolo sulla pila, un DPDA può fare al massimo una mossa.
- L’insieme $\delta(q, a, X) \cup \delta(q, \varepsilon, X)$ può essere vuoto, nel qual caso il DPDA rifiuta definitivamente la stringa, senza che vi siano altre possibilità di riconoscerla.

Esempio: riconoscitore di stringhe wcw^R

- È possibile dimostrare che non esiste un DPDA in grado di riconoscere il linguaggio delle stringhe della forma ww^R . Intuizione: il PDA deve “scommettere” sul punto in cui finisce il prefisso w e inizia il suffisso w^R .
- Separando con una “sentinella” c il prefisso w dal suffisso w^R , il linguaggio delle stringhe della forma wcw^R diventa riconoscibile dal DPDA seguente

$$(\{q_0, q_1, q_2\}, \{0, 1, c\}, \{0, 1, Z_0\}, \delta, q_0, Z_0, \{q_2\})$$



- Notare che il comportamento è deterministico anche nello stato q_1 , in quanto la transizione ϵ è possibile solo se in cima alla pila c’è Z_0 .

DPDA e linguaggi regolari

Teorema

Ogni linguaggio regolare è riconosciuto da un DPDA.

Dimostrazione

Sia $A = (Q, \Sigma, \delta_A, q_0, F)$ un DFA che riconosce un certo linguaggio regolare L . Definiamo un PDA strutturalmente identico ad A che non usa la pila. Sia

$$P \stackrel{\text{def}}{=} (Q, \Sigma, \{Z_0\}, \delta_P, q_0, Z_0, F)$$

dove

$$\begin{aligned}\delta_P(q, a, Z_0) &= \{(\delta_A(q, a), Z_0)\} \quad \text{per ogni } q \in Q \text{ e } a \in \Sigma \\ \delta_P(q, \varepsilon, Z_0) &= \emptyset \quad \text{per ogni } q \in Q\end{aligned}$$

È facile mostrare che $(q_0, w, Z_0) \vdash^* (\hat{\delta}_A(q_0, w), \varepsilon, Z_0)$ da cui segue il risultato.

DPDA e grammatiche ambigue

Teorema

Per ogni DPDA P , esiste una grammatica libera non ambigua G tale che $L(G) = N(P)$.

Osservazione

Il viceversa del risultato qui sopra **non vale**. In particolare, esistono grammatiche non ambigue il cui linguaggio non è riconosciuto da alcun DPDA. Ad esempio

$$S \rightarrow \epsilon \mid 0S0 \mid 1S1$$

è non ambigua e genera il linguaggio $\{ww^R \mid w \in \{0,1\}^*\}$ che **non è riconoscibile da un DPDA**.

Conclusione

La famiglia dei linguaggi riconoscibili da DPDA è inclusa in – ma non coincide con – quella dei linguaggi generabili da grammatiche libere non ambigue.

Potere espressivo dei formalismi visti fino ad ora

- DFA = NFA = ϵ -NFA = RE
- DFA \subsetneq DPDA \subsetneq CFG non ambigue \subsetneq CFG = PDA
- Non abbiamo ancora un algoritmo per ottenere il DPDA quando esiste

Linguaggi Formali e Traduttori

3.6 Pumping lemma per i linguaggi liberi

- Sommario
- Linguaggi **non** liberi
- Pumping lemma per linguaggi liberi
- Il linguaggio $a^kb^kc^k$ non è libero
- Programma per dimostrare il pumping lemma
- Forma normale di Chomsky
- Alberi sintattici di grammatiche CNF
- Pumping lemma: dimostrazione (1/5)
- Pumping lemma: dimostrazione (2/5)
- Pumping lemma: dimostrazione (3/5)
- Pumping lemma: dimostrazione (4/5)
- Pumping lemma: dimostrazione (5/5)
- Esercizi

È proibito condividere e divulgare in qualsiasi forma i materiali didattici caricati sulla piattaforma e le lezioni svolte in videoconferenza: ogni azione che viola questa norma sarà denunciata agli organi di Ateneo e perseguita a termini di legge.

Sommario

Per dimostrare che un linguaggio **è libero**, basta esibire un automa a pila (PDA) che lo riconosce, oppure una grammatica libera che lo genera. L'incapacità di trovare siffatto automa o siffatta grammatica non è una dimostrazione del fatto che il linguaggio non è libero.

In questa lezione rispondiamo alle seguenti domande:

1. Esistono linguaggi **non** liberi?
2. Se sì, come dimostro che un linguaggio **non** è libero?

Linguaggi non liberi

- Cerchiamo una proprietà P soddisfatta da tutti i linguaggi liberi:

$$L \text{ libero} \Rightarrow L \text{ soddisfa } P$$

- Se troviamo un linguaggio L che **non soddisfa P** , allora per **contrapposizione** possiamo concludere che **L non è libero**:

$$L \text{ non soddisfa } P \Rightarrow L \text{ non è libero}$$

Pumping lemma per linguaggi liberi

Teorema

Per ogni linguaggio libero L esiste $n \in \mathbb{N}$ tale che, per ogni $z \in L$ con $|z| \geq n$, esistono u, v, w, x e y tali che $z = uvwxy$ e inoltre:

- (1) $vx \neq \varepsilon$
- (2) $|vwx| \leq n$
- (3) $uv^kwx^ky \in L$ per ogni $k \geq 0$.

In prosa

- Ogni stringa z “sufficientemente lunga” ($|z| \geq n$) di un linguaggio libero L ...
- ... contiene due sottostinghe non entrambe vuote (1) ...
- ... e “non troppo distanti” una dall’altra (2) ...
- ... che possono essere entrambe eliminate ($k = 0$) ...
- ... o replicate a piacere lo stesso numero di volte ($k > 0$) ...
- ... consentendoci di trovare altre stringhe di L (3).

Il linguaggio $a^k b^k c^k$ non è libero

Dimostriamo che $L = \{a^k b^k c^k \mid k \geq 0\}$ non è libero facendo vedere che per L il pumping lemma non vale.

Il linguaggio $a^k b^k c^k$ non è libero

Dimostriamo che $L = \{a^k b^k c^k \mid k \geq 0\}$ non è libero facendo vedere che per L il pumping lemma non vale.

Supponiamo, per assurdo, che esista n con le proprietà enunciate nella [slide 4](#).

Il linguaggio $a^k b^k c^k$ non è libero

Dimostriamo che $L = \{a^k b^k c^k \mid k \geq 0\}$ non è libero facendo vedere che per L il pumping lemma non vale.

Supponiamo, per assurdo, che esista n con le proprietà enunciate nella [slide 4](#).

Consideriamo la stringa $z = a^n b^n c^n$, che è in L e ha la proprietà $|z| = 3n \geq n$.

Il linguaggio $a^k b^k c^k$ non è libero

Dimostriamo che $L = \{a^k b^k c^k \mid k \geq 0\}$ non è libero facendo vedere che per L il pumping lemma non vale.

Supponiamo, per assurdo, che esista n con le proprietà enunciate nella [slide 4](#).

Consideriamo la stringa $z = a^n b^n c^n$, che è in L e ha la proprietà $|z| = 3n \geq n$.

Devono esistere u, v, w, x e y tali che $z = uvwxy$ e che soddisfano le condizioni 1–3 della [slide 4](#).

Il linguaggio $a^k b^k c^k$ non è libero

Dimostriamo che $L = \{a^k b^k c^k \mid k \geq 0\}$ non è libero facendo vedere che per L il pumping lemma non vale.

Supponiamo, per assurdo, che esista n con le proprietà enunciate nella [slide 4](#).

Consideriamo la stringa $z = a^n b^n c^n$, che è in L e ha la proprietà $|z| = 3n \geq n$.

Devono esistere u, v, w, x e y tali che $z = uvwxy$ e che soddisfano le condizioni 1–3 della [slide 4](#).

Dalla condizione 1 sappiamo che $vx \neq \epsilon$.

Il linguaggio $a^k b^k c^k$ non è libero

Dimostriamo che $L = \{a^k b^k c^k \mid k \geq 0\}$ non è libero facendo vedere che per L il pumping lemma non vale.

Supponiamo, per assurdo, che esista n con le proprietà enunciate nella [slide 4](#).

Consideriamo la stringa $z = a^n b^n c^n$, che è in L e ha la proprietà $|z| = 3n \geq n$.

Devono esistere u, v, w, x e y tali che $z = uvwxy$ e che soddisfano le condizioni 1–3 della [slide 4](#).

Dalla condizione 1 sappiamo che $vx \neq \epsilon$.

Dalla condizione 2 sappiamo che vwx non può contenere sia a che c , in quanto in z la a più a destra è separata dalla c più a sinistra da $n b$. Dunque i casi (non esclusivi, ma che coprono tutte le possibilità) sono due: o vwx non contiene a oppure vwx non contiene c .

Il linguaggio $a^k b^k c^k$ non è libero

Dimostriamo che $L = \{a^k b^k c^k \mid k \geq 0\}$ non è libero facendo vedere che per L il pumping lemma non vale.

Supponiamo, per assurdo, che esista n con le proprietà enunciate nella [slide 4](#).

Consideriamo la stringa $z = a^n b^n c^n$, che è in L e ha la proprietà $|z| = 3n \geq n$.

Devono esistere u, v, w, x e y tali che $z = uvwxy$ e che soddisfano le condizioni 1–3 della [slide 4](#).

Dalla condizione 1 sappiamo che $vx \neq \epsilon$.

Dalla condizione 2 sappiamo che vwx non può contenere sia a che c , in quanto in z la a più a destra è separata dalla c più a sinistra da n b . Dunque i casi (non esclusivi, ma che coprono tutte le possibilità) sono due: o vwx non contiene a oppure vwx non contiene c .

Dalla condizione 3 sappiamo che $uwxy \in L$.

Il linguaggio $a^k b^k c^k$ non è libero

Dimostriamo che $L = \{a^k b^k c^k \mid k \geq 0\}$ non è libero facendo vedere che per L il pumping lemma non vale.

Supponiamo, per assurdo, che esista n con le proprietà enunciate nella [slide 4](#).

Consideriamo la stringa $z = a^n b^n c^n$, che è in L e ha la proprietà $|z| = 3n \geq n$.

Devono esistere u, v, w, x e y tali che $z = uvwxy$ e che soddisfano le condizioni 1–3 della [slide 4](#).

Dalla condizione 1 sappiamo che $vx \neq \epsilon$.

Dalla condizione 2 sappiamo che vwx non può contenere sia a che c , in quanto in z la a più a destra è separata dalla c più a sinistra da n b . Dunque i casi (non esclusivi, ma che coprono tutte le possibilità) sono due: o vx non contiene a oppure vx non contiene c .

Dalla condizione 3 sappiamo che $uw y \in L$.

Ora, se vx non contiene a , in $uw y$ il numero di a è rimasto n , mentre il numero di b e/o c è diminuito. Se vx non contiene c , in $uw y$ il numero di c è rimasto n , mentre il numero di a e/o di b è diminuito. In entrambi i casi abbiamo raggiunto una contraddizione.

Programma per dimostrare il pumping lemma

1. Argomentiamo che ogni grammatica libera G può essere trasformata in una forma – detta in **forma normale di Chomsky** – che è “quasi equivalente” a G e in cui le produzioni sono particolarmente semplici.
2. L'esistenza della forma normale di Chomsky di una grammatica è conseguenza di una serie di trasformazioni (non difficili, ma complessivamente tediose) della grammatica dettagliate nel libro di testo (Sezioni 7.1.1 – 7.1.4, lettura facoltativa con caffè).
3. Per ogni grammatica in forma normale di Chomsky, dimostriamo una relazione forte tra la profondità di un albero sintattico della grammatica e la lunghezza del suo prodotto.
4. Dimostriamo il pumping lemma per i linguaggi liberi.

Forma normale di Chomsky

Definizione

Diciamo che una grammatica è in **forma normale di Chomsky (CNF)**, da Chomsky Normal Form) se ogni sua produzione è della forma

- $A \rightarrow BC$ dove A , B e C sono variabili, oppure
- $A \rightarrow a$ dove A è una variabile e a un terminale.

Osservazione

È evidente che nessuna variabile (inclusa quella iniziale) è annullabile in una grammatica CNF. Infatti, ogni derivazione $A \Rightarrow \dots$ aumenta o lascia invariata la lunghezza della stringa derivata da A , la quale è una stringa lunga 1.

Teorema

Se G è una grammatica che genera almeno una stringa non vuota, allora esiste una grammatica G' in forma normale di Chomsky tale che $L(G') = L(G) - \{\epsilon\}$.

Dimostrazione (facoltativa)

Si veda la Sezione 7.1 del libro.

Alberi sintattici di grammatiche CNF

Teorema

Sia G una grammatica in forma normale di Chomsky e w il prodotto di un albero sintattico di G avente profondità $n \geq 1$. Allora $|w| \leq 2^{n-1}$.

Dimostrazione

Si procede per induzione sulla profondità n dell'albero, ricordando che ogni foglia dell'albero deve essere etichettata con un terminale.

(Caso base $n = 1$) Allora l'albero ha una radice A e un'unica foglia a che coincide con w . Concludiamo $1 = |w| \leq 2^{n-1} = 1$.

(Caso induttivo $n > 1$) Allora l'albero ha una radice A con esattamente due figli etichettati B e C alla radice di due sottoalberi la cui profondità è non superiore a $n - 1$.

Detti w_1 e w_2 i prodotti di questi due sottoalberi,abbiamo che $w = w_1 w_2$.

Usando l'ipotesi induttiva, deduciamo che $|w_1| \leq 2^{n-2}$ e $|w_2| \leq 2^{n-2}$.

Concludiamo $|w| = |w_1| + |w_2| \leq 2^{n-2} + 2^{n-2} = 2^{n-1}$.

Pumping lemma: dimostrazione (1/5)

Sia L un linguaggio libero.

Se $L = \emptyset$ oppure $L = \{\epsilon\}$, allora è sufficiente prendere $n = 1$ e l'enunciato del pumping lemma vale banalmente, dal momento che non ci sono stringhe di L di lunghezza maggiore o uguale a 1.

Se L contiene almeno una stringa diversa da ϵ , sia $G = (V, T, P, S)$ una grammatica in forma normale di Chomsky che genera $L - \{\epsilon\}$.

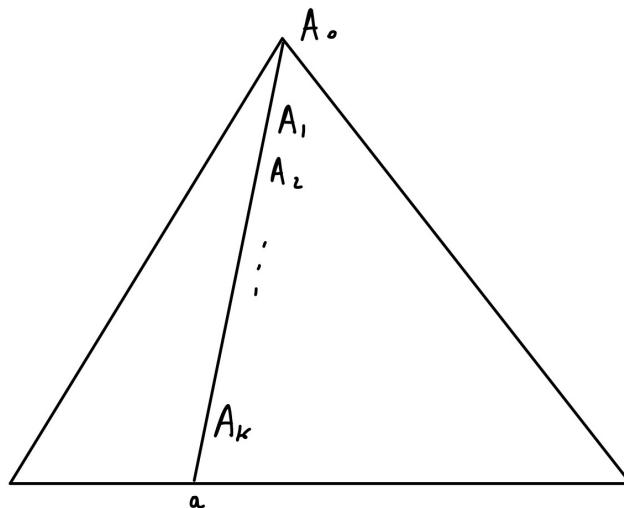
Poniamo $n = 2^m$ dove $m = |V|$.

Prendiamo $z \in L$ tale che $|z| \geq n$.

Per il [teorema della slide precedente](#), ogni albero sintattico di G di profondità m ha come prodotto stringhe lunghe al massimo $2^{m-1} = n/2$. Deduciamo che ogni albero sintattico che ha come radice S e come prodotto z deve avere una profondità maggiore o uguale a $m + 1$, in quanto z è lunga almeno il doppio di 2^{m-1} .

Pumping lemma: dimostrazione (2/5)

Deduciamo che questo albero sintattico avrà la forma



in cui è presente un cammino lungo $k \geq m + 1$ che tocca almeno $m + 2$ nodi, dei quali almeno $m + 1$ sono nodi interni e dunque etichettati con variabili A_i , mentre uno (l'ultimo) è una foglia etichettata con un terminale a .

In particolare, le almeno $m + 1$ variabili toccate dal cammino non possono essere tutte distinte, poiché la grammatica ne ha solo m .

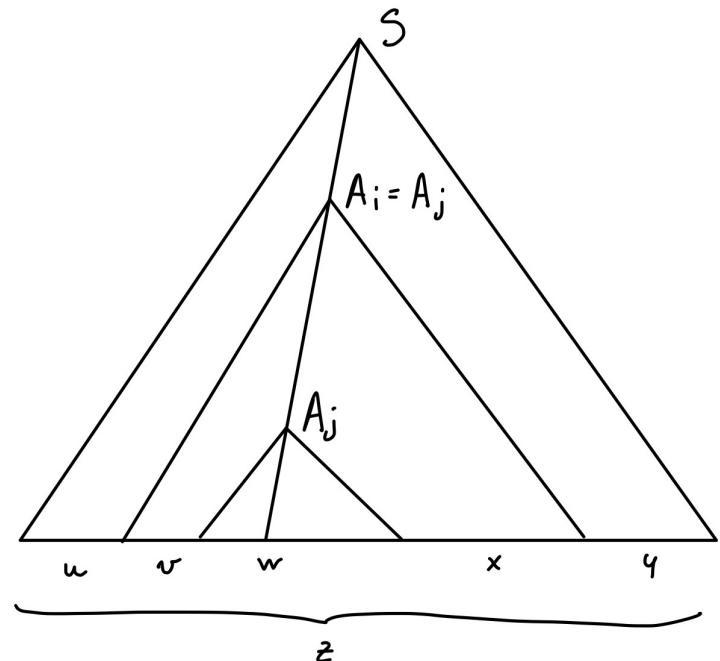
Pumping lemma: dimostrazione (3/5)

Deduciamo che almeno due delle ultime $m + 1$ variabili del cammino (da A_{k-m} a A_k incluse) devono essere uguali.

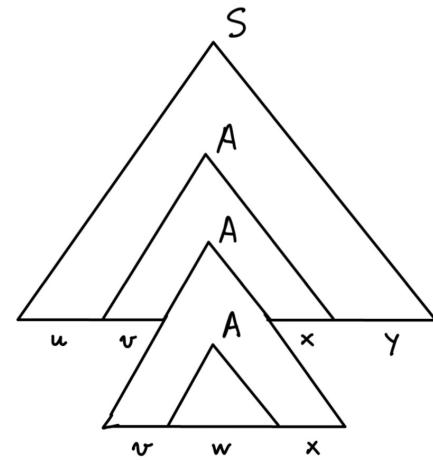
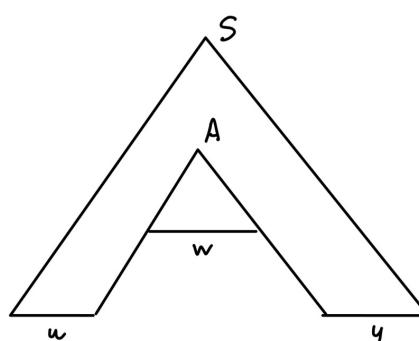
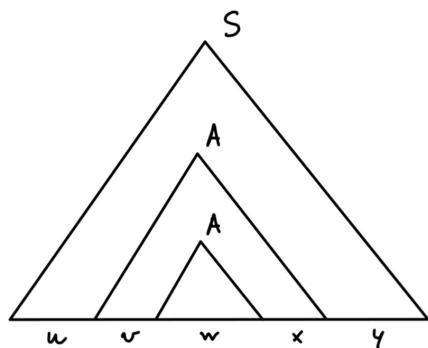
Supponiamo dunque $A_i = A_j = A$ con $k - m \leq i < j \leq k$. Il cammino fatto da $A_0 = S$ ad a può essere rappresentato come nella figura a destra, in cui il prodotto dell'albero è stato così scomposto:

- w è il prodotto del sottoalbero radicato in A_j ;
- v e x sono le stringhe rispettivamente a sinistra e a destra di w nel prodotto del sottoalbero radicato in A_i ;
- u e y sono le stringhe rispettivamente a sinistra e a destra del prodotto del sottoalbero radicato in A_i .

Evidentemente $z = uvwxy$.



Pumping lemma: dimostrazione (4/5)



Dall'albero sintattico iniziale (riprodotto a sinistra) è ora possibile costruirne altri:

- rimpiazzando il sottoalbero radicato in A_i con quello radicato in A_j si ottiene un albero sintattico con prodotto $uw\gamma$;
- rimpiazzando il sottoalbero radicato in A_j con (una copia di) quello radicato in A_i si ottiene un albero sintattico con prodotto $uvvwxxyy = uv^2wx^2y$;
- in generale, iterando il rimpiazzamento del punto precedente, è possibile ottenere alberi sintattici con prodotto uv^kwx^ky per ogni $k \geq 1$.

Pumping lemma: dimostrazione (5/5)

Per concludere la dimostrazione osserviamo che:

- $vx \neq \varepsilon$

Infatti, il cammino da A_i ad A_j deve contenere almeno una diramazione che produce almeno un simbolo in quanto la grammatica è in forma normale di Chomsky e non contiene produzioni ε ;

- $|vwx| \leq n$

Infatti il sottoalbero radicato in A_i ha profondità non superiore a $m + 1$ (vi è una sola variabile che si ripete nel suo cammino più lungo), dunque per il teorema in [slide 8](#) il suo prodotto vwx ha una lunghezza non superiore a $2^m = n$.

Esercizi

Dimostrare che i seguenti linguaggi non sono liberi:

1. $\{ww \mid w \in \{0,1\}^*\}$
2. $\{0^i 1^j 2^i 3^j \mid i, j \geq 1\}$
3. $\{0^i 1^j 2^k \mid i < j < k\}$

Linguaggi Formali e Traduttori

3.7 Proprietà di chiusura dei linguaggi liberi

- Sommario
- Unione e concatenazione
- Intersezione
- Intersezione con un linguaggio regolare
- Complemento e differenza
- Inversione
- Esercizi e quesiti

È proibito condividere e divulgare in qualsiasi forma i materiali didattici caricati sulla piattaforma e le lezioni svolte in videoconferenza: ogni azione che viola questa norma sarà denunciata agli organi di Ateneo e perseguita a termini di legge.

Sommario

Proprietà di chiusura

Dati due linguaggi liberi L ed L' , i seguenti linguaggi sono liberi?

- $L \cup L'$
- $L \cap L'$
- LL'
- \overline{L}
- $L - L'$
- L^R

Unione e concatenazione

Teorema

I linguaggi liberi sono chiusi per unione e concatenazione.

Dimostrazione

Siano L_1 ed L_2 linguaggi liberi. Dunque esistono due grammatiche libere $G_1 = (V_1, T_1, P_1, S_1)$ e $G_2 = (V_2, T_2, P_2, S_2)$ tali che $L_1 = L(G_1)$ e $L_2 = L(G_2)$.

Senza perdere in generalità, possiamo assumere che $V_1 \cap V_2 = \emptyset$ e che $S \notin V_1 \cup V_2$. Infatti, è sempre possibile scegliere nuovi nomi e rinominare (in maniera consistente) le variabili di una grammatica senza modificarne il linguaggio generato.

Ora, la grammatica

$$(V_1 \cup V_2, T_1 \cup T_2, P_1 \cup P_2 \cup \{S \rightarrow S_1 | S_2\}, S)$$

genera $L_1 \cup L_2$ mentre la grammatica

$$(V_1 \cup V_2, T_1 \cup T_2, P_1 \cup P_2 \cup \{S \rightarrow S_1 S_2\}, S)$$

genera $L_1 L_2$. Concludiamo che $L_1 \cup L_2$ e $L_1 L_2$ sono liberi.

Intersezione

Osservazione

I linguaggi liberi **non** sono chiusi per intersezione.

Dimostrazione

I linguaggi

- $L_1 \stackrel{\text{def}}{=} \{a^n b^n c^m \mid m, n \geq 0\}$
- $L_2 = \{a^m b^n c^n \mid m, n \geq 0\}$

sono libri. Se i linguaggi libri fossero chiusi per intersezione, allora anche il linguaggio

$$L_1 \cap L_2 = \{a^n b^n c^n \mid n \geq 0\}$$

sarebbe libero, mentre **abbiamo dimostrato che non lo è.**

Intersezione con un linguaggio regolare

Teorema

Se L è un linguaggio libero ed R è un linguaggio regolare, allora $L \cap R$ è un linguaggio libero.

Dimostrazione (intuizione)

Se L è un linguaggio libero allora esiste un PDA P che accetta L per stato finale.

Se R è un linguaggio regolare allora esiste un DFA M che accetta R .

Si può costruire un PDA che accetta $L \cap R$ per stato finale costruendo il “prodotto” di P ed M , in maniera analoga a quanto già visto nella [costruzione diretta del DFA che riconosce l'intersezione di due linguaggi regolari](#).

Dettagli nella Sezione 7.3.4 del libro (facoltativa).

Osservazione

L'intersezione di un linguaggio libero e uno regolare non è un linguaggio regolare in generale. Si prendano ad esempio $L = \{a^n b^n \mid n \geq 0\}$ ed $R = L(a^* b^*)$. Siccome $L \subseteq R$ abbiamo $L \cap R = L$, il quale [non è regolare](#).

Complemento e differenza

Osservazione

I linguaggi liberi **non** sono chiusi per complemento e differenza.

Dimostrazione

Sappiamo che i linguaggi liberi **sono chiusi per unione**. Se fossero chiusi anche per complemento, allora avremmo che

$$L_1 \cap L_2 = \overline{\overline{L_1} \cap \overline{L_2}} = \overline{\overline{L_1} \cup \overline{L_2}}$$

sarebbe sempre un linguaggio libero, contrariamente a quanto **dimostrato in precedenza**.

Dato un linguaggio libero L su un alfabeto Σ , il linguaggio Σ^* è a sua volta libero (dimostrare per esercizio). Se i linguaggi liberi fossero chiusi per differenza, allora $\Sigma^* - L$ sarebbe sempre libero, ma questo è il complemento di L che, come visto sopra, non è libero in generale.

Inversione

Teorema

Se L è un linguaggio libero, allora anche L^R è un linguaggio libero.

Dimostrazione (parziale)

Sia $G = (V, T, P, S)$ una CFG che genera L .

Definiamo $G^R = (V, T, P^R, S)$ dove

$$P^R = \{A \rightarrow \alpha^R \mid A \rightarrow \alpha \in P\}$$

Si può dimostrare che $L(G^R) = L(G)^R$ (Sezione 7.3.3 del libro).

Esercizi e quesiti

1. Se L è un linguaggio libero, cosa si può dire di L^* e di L^+ ? Sono liberi?
2. Se L_i con $i \in \mathbb{N}$ è una famiglia **infinita** di linguaggi liberi, cosa si può dire di $\bigcup_{i \in \mathbb{N}} L_i$? È sempre un linguaggio libero?
3. Se L è un linguaggio libero ed R è un linguaggio regolare, cosa si può dire di $L - R$? È libero?