STRUMENTI FORMALI PER LA BIOINFORMATICA

La gerarchia di Chomsky - Parte II

Ringraziamenti

Le figure (e il testo) sono prese dai libri (o dispense):

P. Degano, *Fondamenti di Informatica: Calcolabilità e Complessità*, Dispense, 2019.

J. Hopcroft, R. Motwani, J. Ullman , *Automi, Linguaggi e Calcolabilità*, Addison Wesley Pearson Education Italia s.r.l, Terza Edizione, 2009.

Michael Sipser, *Introduzione alla teoria della Computazione*, Apogeo Education, Maggioli Editore, 2016 (traduzione italiana di Introduction to the Theory of Computation, 3rd Edition).

Abbiamo definito una gerarchia di grammatiche e una corrispondente gerarchia di linguaggi.

Definizione

Una grammatica (di tipo 0 o a struttura di frase) G, è una quadrupla $G = (V, T, P, S)$ dove:

Definizione

Una grammatica (di tipo 0 o a struttura di frase) G, è una quadrupla $G = (V, T, P, S)$ dove:

- V*:* **Insieme finito di variabili** *(dette anche non terminali o categorie sintattiche).*

Definizione

Una grammatica (di tipo 0 o a struttura di frase) G, è una quadrupla $G = (V, T, P, S)$ dove:

- V: **Insieme finito di variabili** *(dette anche non terminali o categorie sintattiche).*
- T: **Insieme finito di simboli terminali** *(o alfabeto dei terminali).* $T \cap V = \emptyset$.

Definizione

Una grammatica (di tipo 0 o a struttura di frase) G, è una quadrupla $G = (V, T, P, S)$ dove:

- V*:* **Insieme finito di variabili** *(dette anche non terminali o categorie sintattiche).*
- T*:* **Insieme finito di simboli terminali** *(o alfabeto dei terminali).* $T \cap V = \emptyset$.
- P*:* **Insieme finito delle produzioni**. *Ogni produzione ha la forma* $\alpha \to \beta$, *dove* $\alpha \in (V \cup T)^+$ *e* $\beta \in (V \cup T)^*$.

Definizione

Una grammatica (di tipo 0 o a struttura di frase) G, è una quadrupla $G = (V, T, P, S)$ dove:

- V: **Insieme finito di variabili** *(dette anche non terminali o categorie sintattiche).*
- T: **Insieme finito di simboli terminali** *(o alfabeto dei terminali).* $T \cap V = \emptyset$.
- P: **Insieme finito delle produzioni**. *Ogni produzione ha la forma $\alpha \to \beta$, dove $\alpha \in (V \cup T)^+$ e $\beta \in (V \cup T)^*$.*
- S: *Una variabile, detta* **simbolo iniziale** *(o start symbol).*

Definizione

Una grammatica $G = (V, T, P, S)$ è una grammatica di tipo 1 o dipendente dal contesto se ogni $\alpha \to \beta \in P$ è tale che $\alpha \in (V \cup T)^+$, $\beta \in (V \cup T)^+$ e il lato destro di ogni produzione ha lunghezza almeno pari al lato sinistro

$$\forall \alpha \to \beta \in P \quad |\beta| \geq |\alpha|$$

In più permettiamo che $S \to \epsilon$ sia in P a condizione che S non compaia alla destra di nessuna produzione.

Definizione

Una grammatica $G = (V, T, P, S)$ *è una grammatica di tipo 2 se ogni* $\alpha \to \beta \in P$ *è tale che* $\alpha \in V$ *e* $\beta \in (V \cup T)^+$ *cioè ogni produzione ha la forma* $A \to \beta$*, dove* A *è una variabile e* β *è una stringa non vuota di variabili e terminali.*
In più permettiamo che $S \to \epsilon$ *sia in* P *a condizione che* S *non compaia alla destra di nessuna produzione.*

Definizione

Una grammatica $G = (V, T, P, S)$ è una grammatica di tipo 2 se ogni $\alpha \to \beta \in P$ è tale che $\alpha \in V$ e $\beta \in (V \cup T)^+$ cioè ogni produzione ha la forma $A \to \beta$, dove A è una variabile e β è una stringa non vuota di variabili e terminali.
In più permettiamo che $S \to \epsilon$ sia in P a condizione che S non compaia alla destra di nessuna produzione.

- **Nota**. Le grammatiche di tipo 2 e le grammatiche context-free sono computazionalmente equivalenti: ogni grammatica di tipo 2 è una grammatica context-free e se G è una grammatica context-free esiste una grammatica G' di tipo 2 tale che $L(G) = L(G')$.

Definizione

Una grammatica $G = (V, T, P, S)$ *è una grammatica di tipo 3 o regolare se ogni* $\alpha \to \beta \in P$ *è tale che* $\alpha \in V$ *e* $\beta \in T \cup (T \cdot V)$, *cioè se ogni produzione è della forma* $A \to aB$ *o* $A \to a$, *dove* A, B *sono variabili e* a *è un terminale.*

In più permettiamo che $S \to \epsilon$ *sia in* P *a condizione che* S *non compaia alla destra di nessuna produzione.*

Riassumendo

- Una grammatica $G = (V, T, P, S)$, con V, T insiemi finiti e disgiunti, P insieme finito, $S \in V$ è

Riassumendo

- Una grammatica $G = (V, T, P, S)$, con V, T insiemi finiti e disgiunti, P insieme finito, $S \in V$ è
  - **di tipo 1** se ogni produzione in P ha la forma $\alpha \to \beta$, $\alpha, \beta \in (V \cup T)^+$ e $|\beta| \geq |\alpha|$.

- Una grammatica $G = (V, T, P, S)$, con V, T insiemi finiti e disgiunti, P insieme finito, $S \in V$ è
  - **di tipo 1** se ogni produzione in P ha la forma $\alpha \to \beta$, $\alpha, \beta \in (V \cup T)^+$ e $|\beta| \geq |\alpha|$.
  - **di tipo 2** se ogni produzione in P ha la forma $A \to \beta$, con $A \in V$ e $\beta \in (V \cup T)^+$

Riassumendo

- Una grammatica $G = (V, T, P, S)$, con V, T insiemi finiti e disgiunti, P insieme finito, $S \in V$ è
  - **di tipo 1** se ogni produzione in P ha la forma $\alpha \to \beta$, $\alpha, \beta \in (V \cup T)^+$ e $|\beta| \geq |\alpha|$.
  - **di tipo 2** se ogni produzione in P ha la forma $A \to \beta$, con $A \in V$ e $\beta \in (V \cup T)^+$
  - **di tipo 3** se ogni produzione in P ha la forma $A \to aB$ o $A \to a$, dove A, B sono variabili e a è un terminale.

- Una grammatica $G = (V, T, P, S)$, con V, T insiemi finiti e disgiunti, P insieme finito, $S \in V$ è
  - **di tipo 1** se ogni produzione in P ha la forma $\alpha \to \beta$, $\alpha, \beta \in (V \cup T)^+$ e $|\beta| \geq |\alpha|$.
  - **di tipo 2** se ogni produzione in P ha la forma $A \to \beta$, con $A \in V$ e $\beta \in (V \cup T)^+$
  - **di tipo 3** se ogni produzione in P ha la forma $A \to aB$ o $A \to a$, dove A, B sono variabili e a è un terminale.
- In più permettiamo che $S \to \epsilon$ sia in P a condizione che S non compaia alla destra di nessuna produzione.

Teorema (Gerarchia di grammatiche)

La classe delle grammatiche di tipo i include strettamente quella delle grammatiche di tipo $i+1$, $0 \leq i \leq 2$.

Alla gerarchia di grammatiche introdotta corrisponde una gerarchia sui linguaggi generati.

Alla gerarchia di grammatiche introdotta corrisponde una gerarchia sui linguaggi generati.

Definizione

Un linguaggio L è di tipo i se esiste una grammatica G di tipo i tale che $L = L(G)$.

Alla gerarchia di grammatiche introdotta corrisponde una gerarchia sui linguaggi generati.

Definizione

Un linguaggio L è di tipo i se esiste una grammatica G di tipo i tale che $L = L(G)$.

Otteniamo quattro classi di linguaggi, i linguaggi di tipo 0, 1, 2, 3. Le quattro classi formano la

Gerarchia di Chomsky

Abbiamo definito una gerarchia di grammatiche e una corrispondente gerarchia di linguaggi.

Abbiamo anche una corrispondente gerarchia di automi.

Linguaggi di tipo 2 o liberi dal contesto

- Le grammatiche di tipo 2 sono le grammatiche in cui il lato sinistro di ogni produzione $A \rightarrow \alpha$ è una variabile A e il lato destro α è una stringa non vuota di variabili e terminali. In più permettiamo che $S \rightarrow \epsilon$ sia in P a condizione che S non compaia alla destra di nessuna produzione.

Linguaggi di tipo 2 o liberi dal contesto

- Le grammatiche di tipo 2 sono le grammatiche in cui il lato sinistro di ogni produzione $A \rightarrow \alpha$ è una variabile A e il lato destro α è una stringa non vuota di variabili e terminali. In più permettiamo che $S \rightarrow \epsilon$ sia in P a condizione che S non compaia alla destra di nessuna produzione.
- Le grammatiche di tipo 2 e le grammatiche context-free sono computazionalmente equivalenti.

Linguaggi di tipo 2 o liberi dal contesto

- Le grammatiche di tipo 2 sono le grammatiche in cui il lato sinistro di ogni produzione $A \to \alpha$ è una variabile A e il lato destro α è una stringa non vuota di variabili e terminali. In più permettiamo che $S \to \epsilon$ sia in P a condizione che S non compaia alla destra di nessuna produzione.
- Le grammatiche di tipo 2 e le grammatiche context-free sono computazionalmente equivalenti.
- Un linguaggio L è di tipo 2 se esiste una grammatica G di tipo 2 tale che $L(G) = L$.

Linguaggi di tipo 2 o liberi dal contesto

- Le grammatiche di tipo 2 sono le grammatiche in cui il lato sinistro di ogni produzione $A \to \alpha$ è una variabile A e il lato destro α è una stringa non vuota di variabili e terminali. In più permettiamo che $S \to \epsilon$ sia in P a condizione che S non compaia alla destra di nessuna produzione.
- Le grammatiche di tipo 2 e le grammatiche context-free sono computazionalmente equivalenti.
- Un linguaggio L è di tipo 2 se esiste una grammatica G di tipo 2 tale che $L(G) = L$.
- Per ogni grammatica G di tipo 2 esiste un automa a pila P tale che $L(G) = L(P)$.

Linguaggi di tipo 2 o liberi dal contesto

- Le grammatiche di tipo 2 sono le grammatiche in cui il lato sinistro di ogni produzione $A \to \alpha$ è una variabile A e il lato destro α è una stringa non vuota di variabili e terminali. In più permettiamo che $S \to \epsilon$ sia in P a condizione che S non compaia alla destra di nessuna produzione.
- Le grammatiche di tipo 2 e le grammatiche context-free sono computazionalmente equivalenti.
- Un linguaggio L è di tipo 2 se esiste una grammatica G di tipo 2 tale che $L(G) = L$.
- Per ogni grammatica G di tipo 2 esiste un automa a pila P tale che $L(G) = L(P)$.
- Per ogni automa a pila P esiste una grammatica G di tipo 2 tale che $L(P) = L(G)$.

- Le grammatiche di tipo 3 (o regolari) sono le grammatiche in cui ogni produzione è della forma $A \to aB$ o $A \to a$, dove A, B sono variabili e a è un terminale. In più permettiamo che $S \to \epsilon$ sia in P a condizione che S non compaia alla destra di nessuna produzione.

Linguaggi di tipo 3

- Le grammatiche di tipo 3 (o regolari) sono le grammatiche in cui ogni produzione è della forma $A \rightarrow aB$ o $A \rightarrow a$, dove A, B sono variabili e a è un terminale. In più permettiamo che $S \rightarrow \epsilon$ sia in P a condizione che S non compaia alla destra di nessuna produzione.
- Un linguaggio L è di tipo 3 se esiste una grammatica G di tipo 3 tale che $L(G) = L$.

- Le grammatiche di tipo 3 (o regolari) sono le grammatiche in cui ogni produzione è della forma $A \to aB$ o $A \to a$, dove A, B sono variabili e a è un terminale. In più permettiamo che $S \to \epsilon$ sia in P a condizione che S non compaia alla destra di nessuna produzione.
- Un linguaggio L è di tipo 3 se esiste una grammatica G di tipo 3 tale che $L(G) = L$.
- Abbiamo dimostrato che per ogni grammatica G di tipo 3 esiste un automa finito non deterministico $\mathcal{A}$ tale che $L(G) = L(\mathcal{A})$.

- Le grammatiche di tipo 3 (o regolari) sono le grammatiche in cui ogni produzione è della forma $A \to aB$ o $A \to a$, dove A, B sono variabili e a è un terminale. In più permettiamo che $S \to \epsilon$ sia in P a condizione che S non compaia alla destra di nessuna produzione.
- Un linguaggio L è di tipo 3 se esiste una grammatica G di tipo 3 tale che $L(G) = L$.
- Abbiamo dimostrato che per ogni grammatica G di tipo 3 esiste un automa finito non deterministico $\mathcal{A}$ tale che $L(G) = L(\mathcal{A})$.
- Abbiamo dimostrato che per ogni automa finito deterministico $\mathcal{A}$ esiste una grammatica G di tipo 3 tale che $L(\mathcal{A}) = L(G)$.

- Le grammatiche di tipo 0 sono le grammatiche che non hanno senza alcuna restrizione sulla forma delle produzioni.

- Le grammatiche di tipo 0 sono le grammatiche che non hanno senza alcuna restrizione sulla forma delle produzioni.
- Un linguaggio L è di tipo 0 se esiste una grammatica G di tipo 0 tale che $L(G) = L$.

- Le grammatiche di tipo 0 sono le grammatiche che non hanno senza alcuna restrizione sulla forma delle produzioni.
- Un linguaggio L è di tipo 0 se esiste una grammatica G di tipo 0 tale che $L(G) = L$.
- Sia L di tipo 0. È possibile dimostrare che esiste una macchina di Turing non deterministica M che lo riconosce, cioè tale che $L = L(M)$.

- Le grammatiche di tipo 0 sono le grammatiche che non hanno senza alcuna restrizione sulla forma delle produzioni.
- Un linguaggio L è di tipo 0 se esiste una grammatica G di tipo 0 tale che $L(G) = L$.
- Sia L di tipo 0. È possibile dimostrare che esiste una macchina di Turing non deterministica M che lo riconosce, cioè tale che $L = L(M)$.
- Viceversa, data una macchina di Turing M è possibile dimostrare che esiste una grammatica G di tipo 0 tale che $L(G) = L(M)$.

- Le grammatiche di tipo 0 sono le grammatiche che non hanno senza alcuna restrizione sulla forma delle produzioni.
- Un linguaggio L è di tipo 0 se esiste una grammatica G di tipo 0 tale che $L(G) = L$.
- Sia L di tipo 0. È possibile dimostrare che esiste una macchina di Turing non deterministica M che lo riconosce, cioè tale che $L = L(M)$.
- Viceversa, data una macchina di Turing M è possibile dimostrare che esiste una grammatica G di tipo 0 tale che $L(G) = L(M)$.
- Quindi la classe dei linguaggi di tipo 0 coincide con la classe dei linguaggi Turing riconoscibili.

- Poiché ogni linguaggio di tipo $i+1$ è anche di tipo i, $0 \leq i \leq 2$, i linguaggi context-sensitive sono Turing-riconoscibili. Anche i linguaggi context-free e quelli regolari sono Turing-riconoscibili.

- Poiché ogni linguaggio di tipo $i+1$ è anche di tipo i, $0 \leq i \leq 2$, i linguaggi context-sensitive sono Turing-riconoscibili. Anche i linguaggi context-free e quelli regolari sono Turing-riconoscibili.
- Vogliamo però confrontare queste classi di linguaggi con la classe più ristretta dei linguaggi decidibili.

- Poiché ogni linguaggio di tipo $i+1$ è anche di tipo i, $0 \leq i \leq 2$, i linguaggi context-sensitive sono Turing-riconoscibili. Anche i linguaggi context-free e quelli regolari sono Turing-riconoscibili.
- Vogliamo però confrontare queste classi di linguaggi con la classe più ristretta dei linguaggi decidibili.
- Nota. Quando parliamo di "linguaggio context-sensitive" (o "context-free" o "regolare") intenderemo sempre "una rappresentazione finita di un linguaggio context-sensitive (o context-free o regolare)".

- Poiché ogni linguaggio di tipo $i + 1$ è anche di tipo i, $0 \leq i \leq 2$, i linguaggi context-sensitive sono Turing-riconoscibili. Anche i linguaggi context-free e quelli regolari sono Turing-riconoscibili.
- Vogliamo però confrontare queste classi di linguaggi con la classe più ristretta dei linguaggi decidibili.
- Nota. Quando parliamo di "linguaggio context-sensitive" (o "context-free" o "regolare") intenderemo sempre "una rappresentazione finita di un linguaggio context-sensitive (o context-free o regolare)".

  Quindi grammatiche o automi o, per i linguaggi regolari, anche espressioni regolari.

Le grammatiche di tipo 1 sono le grammatiche in cui il lato destro di ogni produzione ha lunghezza almeno pari al lato sinistro (a parte eventualmente la produzione $S \to \epsilon$).

Linguaggi di tipo 1 o dipendenti dal contesto

Le grammatiche di tipo 1 sono le grammatiche in cui il lato destro di ogni produzione ha lunghezza almeno pari al lato sinistro (a parte eventualmente la produzione $S \to \epsilon$).

Un linguaggio L è di tipo 1 se esiste una grammatica G di tipo 1 tale che $L(G) = L$.

Teorema

I linguaggi di tipo 1 sono decidibili.

Nota. La MT che decide il linguaggio usa implicitamente come sottoprogramma una TM che decide

$$A_{CSG} = \{\langle G, w\rangle \mid G \text{ è di tipo 1 e } w \in L(G)\}$$

(Idea della prova). Sia $G = (V, \Sigma, P, S)$ una grammatica di tipo 1 e sia $w \in L(G)$, $w \neq \epsilon$. Quindi $S \underset{G}{\overset{*}{\Rightarrow}} w$.

(Idea della prova). Sia $G = (V, \Sigma, P, S)$ una grammatica di tipo 1 e sia $w \in L(G)$, $w \neq \epsilon$. Quindi $S \overset{*}{\underset{G}{\Rightarrow}} w$.

Ogni derivazione diretta in $S \overset{*}{\underset{G}{\Rightarrow}} w$ sostituisce una stringa α con una stringa β di lunghezza maggiore o uguale alla lunghezza di α.

(Idea della prova). Sia $G = (V, \Sigma, P, S)$ una grammatica di tipo 1 e sia $w \in L(G)$, $w \neq \epsilon$. Quindi $S \overset{*}{\underset{G}{\Rightarrow}} w$.

Ogni derivazione diretta in $S \overset{*}{\underset{G}{\Rightarrow}} w$ sostituisce una stringa α con una stringa β di lunghezza maggiore o uguale alla lunghezza di α.

Quindi in ognuna di tali derivazioni dirette compaiono solo stringhe di lunghezza minore o uguale alla lunghezza di w.

(Idea della prova). Ad esempio sia $G = (V, \Sigma, P, S)$, dove $V = \{S, B\}$, $T = \{a, b, c\}$ e P consiste nelle seguenti produzioni

$$S \to aSBc \mid abc, \quad cB \to Bc, \quad bB \to bb$$

(Idea della prova). Ad esempio sia $G = (V, \Sigma, P, S)$, dove $V = \{S, B\}$, $T = \{a, b, c\}$ e P consiste nelle seguenti produzioni

$$S \rightarrow aSBc \mid abc, \quad cB \rightarrow Bc, \quad bB \rightarrow bb$$

Sappiamo che $L(G) = \{a^n b^n c^n \mid n \geq 1\}$

(Idea della prova). Ad esempio sia $G = (V, \Sigma, P, S)$, dove $V = \{S, B\}$, $T = \{a, b, c\}$ e P consiste nelle seguenti produzioni

$$S \to aSBc \mid abc, \quad cB \to Bc, \quad bB \to bb$$

Sappiamo che $L(G) = \{a^n b^n c^n \mid n \geq 1\}$

$$S \Rightarrow aSBc \Rightarrow aabcBc \Rightarrow aabBcc \Rightarrow aabbcc$$

(Idea della prova). Sia $G = (V, \Sigma, P, S)$ una grammatica di tipo 1 e sia $w \in \Sigma^+$ con $|w| = n$.

(Idea della prova). Sia $G = (V, \Sigma, P, S)$ una grammatica di tipo 1 e sia $w \in \Sigma^+$ con $|w| = n$.

Risulta $w \in L(G)$ se e solo se esiste una sequenza $\alpha_1, \alpha_2, \ldots, \alpha_k$ di stringhe in $\bigcup_{t=1}^{n}(\Sigma \cup V)^t$ tali che

(Idea della prova). Sia $G = (V, \Sigma, P, S)$ una grammatica di tipo 1 e sia $w \in \Sigma^+$ con $|w| = n$.

Risulta $w \in L(G)$ se e solo se esiste una sequenza $\alpha_1, \alpha_2, \ldots, \alpha_k$ di stringhe in $\bigcup_{t=1}^{n} (\Sigma \cup V)^t$ tali che

$\alpha_1 = S$, $\alpha_k = w$ e $\alpha_i \underset{G}{\Rightarrow} \alpha_{i+1}$, $i = 1, \ldots, k-1$

(Idea della prova). Sia $G = (V, \Sigma, P, S)$ una grammatica di tipo 1 e sia $w \in \Sigma^+$ con $|w| = n$.

Risulta $w \in L(G)$ se e solo se esiste una sequenza $\alpha_1, \alpha_2, \ldots, \alpha_k$ di stringhe in $\bigcup_{t=1}^{n} (\Sigma \cup V)^t$ tali che

$\alpha_1 = S$, $\alpha_k = w$ e $\alpha_i \underset{G}{\Rightarrow} \alpha_{i+1}$, $i = 1, \ldots, k-1$

ovvero

$$S = \alpha_1 \underset{G}{\Rightarrow} \alpha_2 \ldots \underset{G}{\Rightarrow} \alpha_{k-1} \underset{G}{\Rightarrow} \alpha_k = w$$

Prova (cenni). Sia $G = (V, \Sigma, P, S)$ una grammatica di tipo 1 per L. Sia M_G una MT definita come segue (M_G ha una copia di G memorizzata).

$M_G =$ "Su input $w \in \Sigma^*$:

Prova (cenni). Sia $G = (V, \Sigma, P, S)$ una grammatica di tipo 1 per L. Sia M_G una MT definita come segue (M_G ha una copia di G memorizzata).

$M_G =$ "Su input $w \in \Sigma^*$:

1. Verifica se $w = \epsilon$. In tal caso accetta se $S \to \epsilon \in P$; altrimenti rifiuta.

Prova (cenni). Sia $G = (V, \Sigma, P, S)$ una grammatica di tipo 1 per L. Sia M_G una MT definita come segue (M_G ha una copia di G memorizzata).

$M_G =$ "Su input $w \in \Sigma^*$:

1. Verifica se $w = \epsilon$. In tal caso accetta se $S \to \epsilon \in P$; altrimenti rifiuta.
2. Se $w \neq \epsilon$, conta i caratteri di w. Sia $n = |w|$.

Prova (cenni). Sia $G = (V, \Sigma, P, S)$ una grammatica di tipo 1 per L. Sia M_G una MT definita come segue (M_G ha una copia di G memorizzata).

M_G = "Su input $w \in \Sigma^*$:

1. Verifica se $w = \epsilon$. In tal caso accetta se $S \to \epsilon \in P$; altrimenti rifiuta.
2. Se $w \neq \epsilon$, conta i caratteri di w. Sia $n = |w|$.
3. Costruisce un grafo orientato i cui vertici sono le stringhe di $(V \cup \Sigma)^+$ di lunghezza minore di o uguale a n. Pone un arco dalla stringa α alla stringa β, dove $|\alpha| \leq n$ e $|\beta| \leq n$, se $\alpha \underset{G}{\Rightarrow} \beta$.

Prova (cenni). Sia $G = (V, \Sigma, P, S)$ una grammatica di tipo 1 per L. Sia M_G una MT definita come segue (M_G ha una copia di G memorizzata).

$M_G =$ "Su input $w \in \Sigma^*$:

1. Verifica se $w = \epsilon$. In tal caso accetta se $S \to \epsilon \in P$; altrimenti rifiuta.
2. Se $w \neq \epsilon$, conta i caratteri di w. Sia $n = |w|$.
3. Costruisce un grafo orientato i cui vertici sono le stringhe di $(V \cup \Sigma)^+$ di lunghezza minore di o uguale a n. Pone un arco dalla stringa α alla stringa β, dove $|\alpha| \leq n$ e $|\beta| \leq n$, se $\alpha \underset{G}{\Rightarrow} \beta$.
4. Verifica se c'è un cammino dal vertice S al vertice w. In caso affermativo accetta; altrimenti rifiuta."

- M_G è un decisore.

- M_G è un decisore.

  Dopo aver eseguito i primi tre passi, M_G simula uno degli algoritmi esistenti per decidere se c'è un cammino dal vertice S al vertice w. In base all'esistenza o meno di tale cammino, M_G accetta o rifiuta w.

- M_G è un decisore.

  Dopo aver eseguito i primi tre passi, M_G simula uno degli algoritmi esistenti per decidere se c'è un cammino dal vertice S al vertice w. In base all'esistenza o meno di tale cammino, M_G accetta o rifiuta w.

  Quindi M_G si ferma su ogni input w.

- M_G è un decisore.

  Dopo aver eseguito i primi tre passi, M_G simula uno degli algoritmi esistenti per decidere se c'è un cammino dal vertice S al vertice w. In base all'esistenza o meno di tale cammino, M_G accetta o rifiuta w.

  Quindi M_G si ferma su ogni input w.

- M_G decide L.

- M_G è un decisore.

  Dopo aver eseguito i primi tre passi, M_G simula uno degli algoritmi esistenti per decidere se c'è un cammino dal vertice S al vertice w. In base all'esistenza o meno di tale cammino, M_G accetta o rifiuta w.

  Quindi M_G si ferma su ogni input w.

- M_G decide L.

  Infatti $w \in L(G)$ se e solo se c'è un cammino dal vertice S al vertice w.

- M_G è un decisore.

  Dopo aver eseguito i primi tre passi, M_G simula uno degli algoritmi esistenti per decidere se c'è un cammino dal vertice S al vertice w. In base all'esistenza o meno di tale cammino, M_G accetta o rifiuta w.

  Quindi M_G si ferma su ogni input w.

- M_G decide L.

  Infatti $w \in L(G)$ se e solo se c'è un cammino dal vertice S al vertice w.

  Quindi M_G accetta w se e solo se $w \in L(G)$.

Ricordiamo che il nome di “grammatica dipendente dal contesto” deriva dal fatto che per ogni G di tipo 1, esiste G' tale che $L(G) = L(G')$ e in cui le produzioni hanno la forma

$$\alpha_1 A \alpha_2 \to \alpha_1 \beta \alpha_2$$

dove A è una variabile, $\alpha_1, \alpha_2, \beta$ sono stringhe di variabili e terminali e $\beta \neq \epsilon$. (Quindi anche G' è di tipo 1).

Ricordiamo che il nome di “grammatica dipendente dal contesto” deriva dal fatto che per ogni G di tipo 1, esiste G' tale che $L(G) = L(G')$ e in cui le produzioni hanno la forma

$$\alpha_1 A \alpha_2 \rightarrow \alpha_1 \beta \alpha_2$$

dove A è una variabile, $\alpha_1, \alpha_2, \beta$ sono stringhe di variabili e terminali e $\beta \neq \epsilon$. (Quindi anche G' è di tipo 1).

Quindi è possibile sostituire A con β ogni volta che A appare nella stringa in un contesto particolare, cioè tra α_1 e α_2.

Ricordiamo che il nome di “grammatica dipendente dal contesto” deriva dal fatto che per ogni G di tipo 1, esiste G' tale che $L(G) = L(G')$ e in cui le produzioni hanno la forma

$$\alpha_1 A \alpha_2 \to \alpha_1 \beta \alpha_2$$

dove A è una variabile, $\alpha_1, \alpha_2, \beta$ sono stringhe di variabili e terminali e $\beta \neq \epsilon$. (Quindi anche G' è di tipo 1).

Quindi è possibile sostituire A con β ogni volta che A appare nella stringa in un contesto particolare, cioè tra α_1 e α_2.

Alcuni autori adottano come definizione quella in cui le produzioni hanno questo secondo vincolo. Ma la prima formulazione ha il vantaggio di mettere ancora di più in evidenza la decidibilità dei linguaggi generati da queste grammatiche.

Conseguenza del precedente teorema:

Corollario
I linguaggi di tipo 2 sono decidibili.

La decidibilità dei linguaggi context-free può essere provata direttamente.

Definizione

Una grammatica context-free $G = (V, \Sigma, P, S)$ *è in forma normale di Chomsky (o in CNF) se ogni sua produzione ha la forma*

Definizione

Una grammatica context-free $G = (V, \Sigma, P, S)$ è in forma normale di Chomsky (o in CNF) se ogni sua produzione ha la forma

(i) $A \to BC$ *con* $A, B, C \in V$,

Definizione

Una grammatica context-free $G = (V, \Sigma, P, S)$ è in forma normale di Chomsky (o in CNF) se ogni sua produzione ha la forma

(i) $A \to BC$ *con* $A, B, C \in V$,

(ii) $A \to a$ *con* $A \in V$ *e* $a \in \Sigma$,

Definizione di forma normale di Chomsky

Definizione

Una grammatica context-free $G = (V, \Sigma, P, S)$ è in forma normale di Chomsky (o in CNF) se ogni sua produzione ha la forma

(i) $A \to BC$ *con* $A, B, C \in V$,

(ii) $A \to a$ *con* $A \in V$ *e* $a \in \Sigma$,

(iii) $S \to \epsilon$

Definizione di forma normale di Chomsky

Definizione

Una grammatica context-free $G = (V, \Sigma, P, S)$ è in forma normale di Chomsky (o in CNF) se ogni sua produzione ha la forma

(i) $A \to BC$ *con* $A, B, C \in V$,

(ii) $A \to a$ *con* $A \in V$ *e* $a \in \Sigma$,

(iii) $S \to \epsilon$

Inoltre, se $S \to \epsilon$ *è in* P, *allora* $B, C \in V \setminus \{S\}$ *in (i).*

Teorema

Per ogni grammatica context-free G esiste una grammatica G' tale che $L(G') = L(G)$ e G' è in forma normale di Chomsky.

Proposizione

Se $G = (V, \Sigma, P, S)$ *è una grammatica context-free in forma normale di Chomsky e* w *è una stringa in* $L(G)$ *di lunghezza* $n \geq 1$, *allora ogni derivazione di* w *richiede esattamente* $2n - 1$ *passi.*

$$A_{CFG} = \{\langle G, w \rangle \mid G \text{ è una CFG che genera la stringa } w\}.$$

Una TM M per A_{CFG}:
$M =$ "Su input $\langle G, w \rangle$, dove G è una CFG e w è una stringa:

1. Converte G in una grammatica equivalente in forma normale di Chomsky.
2. Lista tutte le derivazioni di $2n-1$ passi, dove n è la lunghezza di w; tranne se $n = 0$, in tal caso lista tutte le derivazioni di un passo.
3. Se una di tali derivazioni genera w, accetta; altrimenti, rifiuta."

M è un decider e $L(M) = A_{CFG}$.

Teorema

Ogni linguaggio context-free è decidibile.

Sia A un CFL. Una (cattiva) idea per mostrare che A è decidibile è quella di convertire un PDA per A direttamente in una TM.
L'idea non è buona perché alcuni rami della computazione del PDA (non deterministico) su alcuni input possono andare avanti per sempre, leggendo e scrivendo la pila senza mai arrestarsi.
La TM che simula il PDA non sarebbe un decisore.

Invece sfruttiamo la macchina M precedentemente costruita.

Sia A un CFL. Sia G una CFG per A, progettiamo una TM M_G che decide A. Costruiamo una copia di G in M_G. M_G funziona come segue.
$M_G =$ "Su input w:

1. Esegue la TM M su input $\langle G, w\rangle$.
2. Se questa macchina accetta, accetta; se essa rifiuta, rifiuta."

Conseguenza del precedente teorema:

Corollario

I linguaggi di tipo 3 sono decidibili.

Ma la decidibilità dei linguaggi regolari si può anche provare convertendo un DFA per un linguaggio in un decider. Oppure con un ragionamento analogo a quello visto per i CFL, utilizzando questa volta la decidibilità di A_{DFA}.

Un automa linear bounded o automa limitato linearmente (LBA) è una macchina di Turing non deterministica che può utilizzare esclusivamente la porzione di nastro su cui inizialmente si trova l'input.

Linguaggi di tipo 1 o dipendenti dal contesto

Un automa linear bounded o automa limitato linearmente (LBA) è una macchina di Turing non deterministica che può utilizzare esclusivamente la porzione di nastro su cui inizialmente si trova l'input.

La classe dei linguaggi riconosciuti da LBA è chiusa rispetto al complemento.

Un automa linear bounded o automa limitato linearmente (LBA) è una macchina di Turing non deterministica che può utilizzare esclusivamente la porzione di nastro su cui inizialmente si trova l'input.

La classe dei linguaggi riconosciuti da LBA è chiusa rispetto al complemento.

Non è noto se la versione deterministica degli LBA sia computazionalmente equivalente agli LBA.

$A_{LBA} = \{\langle M, w\rangle \mid M$ è un LBA che accetta la stringa $w\}$.

$$A_{LBA} = \{\langle M, w\rangle \mid M \text{ è un } LBA \text{ che accetta la stringa } w\}.$$

Proposizione

Sia M un LBA con q stati e g simboli nell'alfabeto di nastro. Esistono esattamente qng^n configurazioni distinte di M per un nastro di lunghezza n.

$$A_{LBA} = \{\langle M, w\rangle \mid M \text{ è un } LBA \text{ che accetta la stringa } w\}.$$

Proposizione

Sia M un LBA con q stati e g simboli nell'alfabeto di nastro. Esistono esattamente qng^n configurazioni distinte di M per un nastro di lunghezza n.

Teorema

A_{LBA} è decidibile.

Teorema

Per ogni grammatica G di tipo 1 esiste un LBA M tale che $L(G) = L(M)$.

Teorema

Per ogni grammatica G di tipo 1 esiste un LBA M tale che $L(G) = L(M)$.

Per ogni LBA M esiste una grammatica G di tipo 1 tale che $L(M) = L(G)$.

Più in generale si può dimostrare che la classe dei linguaggi context-sensitive coincide con la classe dei linguaggi accettati da macchine di Turing non deterministiche che utilizzano una quantità di spazio limitata da una funzione lineare rispetto alla lunghezza dell'input.

Di conseguenza gli LBA sono equivalenti alle macchine di Turing non deterministiche che utilizzano una quantità di spazio limitata da una funzione lineare rispetto alla lunghezza dell'input.

- **Nota.** Nella teoria della complessità vengono introdotte le nozioni di complessità di tempo e complessità di spazio.

- **Nota.** Nella teoria della complessità vengono introdotte le nozioni di complessità di tempo e complessità di spazio.

  Tali nozioni sono entrambe riferite a macchine di Turing deterministiche, a nastro singolo, e che si arrestano su ogni input.

- **Nota.** Nella teoria della complessità vengono introdotte le nozioni di complessità di tempo e complessità di spazio.

  Tali nozioni sono entrambe riferite a macchine di Turing deterministiche, a nastro singolo, e che si arrestano su ogni input.
- Usando la proposizione sugli LBA è possibile provare che per ogni LBA M, esiste un LBA equivalente N che si arresta su ogni input.

- **Nota.** Nella teoria della complessità vengono introdotte le nozioni di complessità di tempo e complessità di spazio.

  Tali nozioni sono entrambe riferite a macchine di Turing deterministiche, a nastro singolo, e che si arrestano su ogni input.
- Usando la proposizione sugli LBA è possibile provare che per ogni *LBA M*, esiste un *LBA* equivalente N che si arresta su ogni input.
- In generale N è una macchina di Turing non deterministica ma usando noti teoremi di complessità di spazio è possibile mettere in relazione i linguaggi dipendenti dal contesto con classi di complessità di spazio.

Linguaggi di tipo 1 o dipendenti dal contesto

Il linguaggio formato da tutte le stringhe di 0 la cui lunghezza è una potenza di 2

$$A = \{0^{2^n} \mid n \geq 0\}$$

è context-sensitive ma non context-free.

Il linguaggio formato da tutte le stringhe di 0 la cui lunghezza è una potenza di 2

$$A = \{0^{2^n} \mid n \geq 0\}$$

è context-sensitive ma non context-free.

Possiamo provare che A non è context-free grazie ai seguenti due fatti.

Linguaggi di tipo 1 o dipendenti dal contesto

Il linguaggio formato da tutte le stringhe di 0 la cui lunghezza è una potenza di 2

$$A = \{0^{2^n} \mid n \geq 0\}$$

è context-sensitive ma non context-free.

Possiamo provare che A non è context-free grazie ai seguenti due fatti.

- Applicando il pumping lemma (per i linguaggi regolari) si può dimostrare che A non è regolare (vedi Sipser, p.85).

Teorema

Un linguaggio su un alfabeto a una lettera è context-free se e solo se è regolare.

I linguaggi regolari su un alfabeto a una lettera hanno automi riconoscitori di forma particolare e struttura particolare.

Il linguaggio

$$A = \{0^{2^n} \mid n \geq 0\}$$

è context-sensitive. Vogliamo definire un LBA M_2 che lo accetta.

Il linguaggio

$$A = \{0^{2^n} \mid n \geq 0\}$$

è context-sensitive. Vogliamo definire un LBA M_2 che lo accetta.

M_2 è descritto nell'Esempio 3.7 del testo di Sipser.

Il linguaggio

$$A = \{0^{2^n} \mid n \geq 0\}$$

è context-sensitive. Vogliamo definire un LBA M_2 che lo accetta.

M_2 è descritto nell'Esempio 3.7 del testo di Sipser.

L'algoritmo si basa sull'osservazione che un numero è una potenza di 2 se e solo se diviso per due dà ancora una potenza di due, fino a ottenere $1 = 2^0$.

Esempio: LBA M_2 per $A = \{0^{2^n} \mid n \geq 0\}$

$M_2 =$ "su input w:

Esempio: LBA M_2 per $A = \{0^{2^n} \mid n \geq 0\}$

$M_2 =$ "su input w:

1. Muove la testina da sinistra a destra sul nastro, rimpiazzando uno 0 sì e uno no con x.

Esempio: LBA M_2 per $A = \{0^{2^n} \mid n \geq 0\}$

$M_2 =$ "su input w:

1. Muove la testina da sinistra a destra sul nastro, rimpiazzando uno 0 sì e uno no con x.
2. Se al passo 1, il nastro conteneva un solo 0, accetta.

Esempio: LBA M_2 per $A = \{0^{2^n} \mid n \geq 0\}$

$M_2 =$ “su input w:

1. Muove la testina da sinistra a destra sul nastro, rimpiazzando uno 0 sì e uno no con x.
2. Se al passo 1, il nastro conteneva un solo 0, accetta.
3. Se al passo 1, il nastro conteneva più di un singolo 0 ed il loro numero era dispari, rifiuta.

Esempio: LBA M_2 per $A = \{0^{2^n} \mid n \geq 0\}$

$M_2 =$ "su input w:

1. Muove la testina da sinistra a destra sul nastro, rimpiazzando uno 0 sì e uno no con x.
2. Se al passo 1, il nastro conteneva un solo 0, accetta.
3. Se al passo 1, il nastro conteneva più di un singolo 0 ed il loro numero era dispari, rifiuta.
4. Riporta la testina all'estremità sinistra del nastro.

Esempio: LBA M_2 per $A = \{0^{2^n} \mid n \geq 0\}$

$M_2 =$ "su input w:

1. Muove la testina da sinistra a destra sul nastro, rimpiazzando uno 0 sì e uno no con x.
2. Se al passo 1, il nastro conteneva un solo 0, accetta.
3. Se al passo 1, il nastro conteneva più di un singolo 0 ed il loro numero era dispari, rifiuta.
4. Riporta la testina all'estremità sinistra del nastro.
5. Va al passo 1."

Esempio: LBA M_2 per $A = \{0^{2^n} \mid n \geq 0\}$

$M_2 =$ "su input w:

1. Muove la testina da sinistra a destra sul nastro, rimpiazzando uno 0 sì e uno no con x.
2. Se al passo 1, il nastro conteneva un solo 0, accetta.
3. Se al passo 1, il nastro conteneva più di un singolo 0 ed il loro numero era dispari, rifiuta.
4. Riporta la testina all'estremità sinistra del nastro.
5. Va al passo 1."

Ogni iterazione del passo 1, dimezza il numero di 0.

Esempio: LBA M_2 per $A = \{0^{2^n} \mid n \geq 0\}$

$M_2 =$ "su input w:

1. Muove la testina da sinistra a destra sul nastro, rimpiazzando uno 0 sì e uno no con x.
2. Se al passo 1, il nastro conteneva un solo 0, accetta.
3. Se al passo 1, il nastro conteneva più di un singolo 0 ed il loro numero era dispari, rifiuta.
4. Riporta la testina all'estremità sinistra del nastro.
5. Va al passo 1."

Ogni iterazione del passo 1, dimezza il numero di 0.

Quando la macchina esegue il passo 1, ricorda se il numero di 0 è pari o dispari (e maggiore di 1).

Esempio: LBA M_2 per $A = \{0^{2^n} \mid n \geq 0\}$

Descrizione formale di $M_2 = (Q, \Sigma, \Gamma, \delta, q_1, q_{accept}, q_{reject})$:

Esempio: LBA M_2 per $A = \{0^{2^n} \mid n \geq 0\}$

Descrizione formale di $M_2 = (Q, \Sigma, \Gamma, \delta, q_1, q_{accept}, q_{reject})$:

- $Q = \{q_1, q_2, q_3, q_4, q_5, q_{accept}, q_{reject}\}$,

Esempio: LBA M_2 per $A = \{0^{2^n} \mid n \geq 0\}$

Descrizione formale di $M_2 = (Q, \Sigma, \Gamma, \delta, q_1, q_{accept}, q_{reject})$:

- $Q = \{q_1, q_2, q_3, q_4, q_5, q_{accept}, q_{reject}\}$,
- $\Sigma = \{0\}$,

Esempio: LBA M_2 per $A = \{0^{2^n} \mid n \geq 0\}$

Descrizione formale di $M_2 = (Q, \Sigma, \Gamma, \delta, q_1, q_{accept}, q_{reject})$:

- $Q = \{q_1, q_2, q_3, q_4, q_5, q_{accept}, q_{reject}\}$,
- $\Sigma = \{0\}$,
- $\Gamma = \{0, x, \sqcup\}$,

Esempio: LBA M_2 per $A = \{0^{2^n} \mid n \geq 0\}$

Descrizione formale di $M_2 = (Q, \Sigma, \Gamma, \delta, q_1, q_{accept}, q_{reject})$:

- $Q = \{q_1, q_2, q_3, q_4, q_5, q_{accept}, q_{reject}\}$,
- $\Sigma = \{0\}$,
- $\Gamma = \{0, x, \sqcup\}$,
- Descriviamo δ mediante un diagramma di stato,

Esempio: LBA M_2 per $A = \{0^{2^n} \mid n \geq 0\}$

Descrizione formale di $M_2 = (Q, \Sigma, \Gamma, \delta, q_1, q_{accept}, q_{reject})$:

- $Q = \{q_1, q_2, q_3, q_4, q_5, q_{accept}, q_{reject}\}$,
- $\Sigma = \{0\}$,
- $\Gamma = \{0, x, \sqcup\}$,
- Descriviamo δ mediante un diagramma di stato,
- Gli stati iniziale, di accettazione e di rifiuto sono rispettivamente $q_1, q_{accept}, q_{reject}$.

Esempio: LBA M_2 per $A = \{0^{2^n} \mid n \geq 0\}$

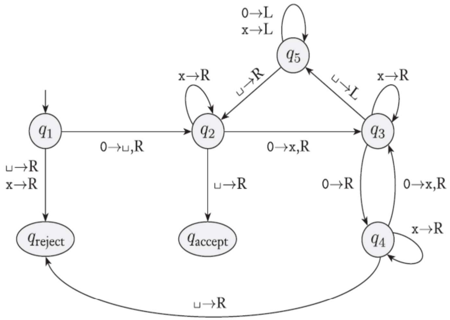


FIGURA 3.8
Diagramma di stato per la macchina di Turing M_2

Il linguaggio

$$B = \{w\#w \mid w \in \{0,1\}^*\}$$

è context-sensitive ma non context-free.

Il linguaggio

$$B = \{w\#w \mid w \in \{0,1\}^*\}$$

è context-sensitive ma non context-free.

Per provare che B non è context-free basta applicare il pumping lemma (per i linguaggi context-free) alla stringa $0^p1^p\#0^p1^p$.

Il linguaggio

$$B = \{w\#w \mid w \in \{0,1\}^*\}$$

è context-sensitive ma non context-free.

Per provare che B non è context-free basta applicare il pumping lemma (per i linguaggi context-free) alla stringa $0^p1^p\#0^p1^p$.

Il ragionamento è lo stesso di quello illustrato nella soluzione dell'esercizio 2.42c, Sipser, p.169 per il linguaggio

$$B' = \{w\#t \mid w \text{ è una sottostringa di } t, \text{ dove } w, t \in \{a,b\}^*\}$$

Linguaggi di tipo 1 o dipendenti dal contesto

Il linguaggio

$$B = \{w\#w \mid w \in \{0,1\}^*\}$$

è context-sensitive. Vogliamo definire un LBA M_1 che lo accetta. L'algoritmo è descritto nell'Esempio 3.9 del testo di Sipser.

Il linguaggio

$$B = \{w\#w \mid w \in \{0,1\}^*\}$$

è context-sensitive. Vogliamo definire un LBA M_1 che lo accetta. L'algoritmo è descritto nell'Esempio 3.9 del testo di Sipser.

$M_1 =$ "su input y:

Il linguaggio
$$B = \{w\#w \mid w \in \{0,1\}^*\}$$
è context-sensitive. Vogliamo definire un LBA M_1 che lo accetta. L'algoritmo è descritto nell'Esempio 3.9 del testo di Sipser.

$M_1 =$ "su input y:

1. Si muove lungo il nastro, raggiungendo posizioni corrispondenti sui due lati del simbolo $\#$, per controllare se queste posizioni contengono lo stesso simbolo. In caso negativo, o nel caso in cui non si trovi il simbolo $\#$, rifiuta. Rimpiazza gli elementi già controllati con x.

Linguaggi di tipo 1 o dipendenti dal contesto

Il linguaggio

$$B = \{w\#w \mid w \in \{0,1\}^*\}$$

è context-sensitive. Vogliamo definire un LBA M_1 che lo accetta. L'algoritmo è descritto nell'Esempio 3.9 del testo di Sipser.

$M_1 =$ "su input y:

1. Si muove lungo il nastro, raggiungendo posizioni corrispondenti sui due lati del simbolo $\#$, per controllare se queste posizioni contengono lo stesso simbolo. In caso negativo, o nel caso in cui non si trovi il simbolo $\#$, rifiuta. Rimpiazza gli elementi già controllati con x.
2. Quando tutti gli elementi a sinistra di $\#$ sono stati controllati, verifica la presenza di eventuali simboli rimanenti a destra di $\#$. Se rimane qualche elemento, allora rifiuta altrimenti accetta."

Esempio: LBA M_1 per $B = \{w\#w \mid w \in \{0,1\}^*\}$

Descrizione formale di $M_1 = (Q, \Sigma, \Gamma, \delta, q_1, q_{accept}, q_{reject})$:

Esempio: LBA M_1 per $B = \{w\#w \mid w \in \{0,1\}^*\}$

Descrizione formale di $M_1 = (Q, \Sigma, \Gamma, \delta, q_1, q_{accept}, q_{reject})$:

- $Q = \{q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_{accept}, q_{reject}\}$,

Esempio: LBA M_1 per $B = \{w\#w \mid w \in \{0,1\}^*\}$

Descrizione formale di $M_1 = (Q, \Sigma, \Gamma, \delta, q_1, q_{accept}, q_{reject})$:

- $Q = \{q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_{accept}, q_{reject}\}$,
- $\Sigma = \{0, 1, \#\}$,

Esempio: LBA M_1 per $B = \{w\#w \mid w \in \{0,1\}^*\}$

Descrizione formale di $M_1 = (Q, \Sigma, \Gamma, \delta, q_1, q_{accept}, q_{reject})$:

- $Q = \{q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_{accept}, q_{reject}\}$,
- $\Sigma = \{0, 1, \#\}$,
- $\Gamma = \{0, 1, \#, x, \sqcup\}$,

Esempio: LBA M_1 per $B = \{w\#w \mid w \in \{0,1\}^*\}$

Descrizione formale di $M_1 = (Q, \Sigma, \Gamma, \delta, q_1, q_{accept}, q_{reject})$:

- $Q = \{q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_{accept}, q_{reject}\}$,
- $\Sigma = \{0, 1, \#\}$,
- $\Gamma = \{0, 1, \#, x, \sqcup\}$,
- Descriviamo δ mediante un diagramma di stato,

Esempio: LBA M_1 per $B = \{w\#w \mid w \in \{0,1\}^*\}$

Descrizione formale di $M_1 = (Q, \Sigma, \Gamma, \delta, q_1, q_{accept}, q_{reject})$:

- $Q = \{q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_{accept}, q_{reject}\}$,
- $\Sigma = \{0, 1, \#\}$,
- $\Gamma = \{0, 1, \#, x, \sqcup\}$,
- Descriviamo δ mediante un diagramma di stato,
- Gli stati iniziale, di accettazione e di rifiuto sono rispettivamente $q_1, q_{accept}, q_{reject}$.

Esempio: LBA M_1 per $B = \{w\#w \mid w \in \{0,1\}^*\}$

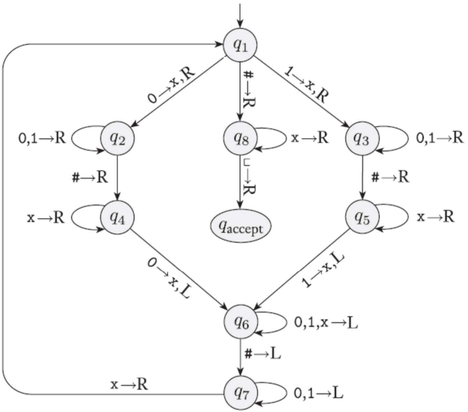
Descrizione formale di $M_1 = (Q, \Sigma, \Gamma, \delta, q_1, q_{accept}, q_{reject})$:

- $Q = \{q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_{accept}, q_{reject}\}$,
- $\Sigma = \{0, 1, \#\}$,
- $\Gamma = \{0, 1, \#, x, \sqcup\}$,
- Descriviamo δ mediante un diagramma di stato,
- Gli stati iniziale, di accettazione e di rifiuto sono rispettivamente $q_1, q_{accept}, q_{reject}$.

Nota: conveniamo che le transizioni mancanti nel diagramma vadano implicitamente in q_{reject}.

Esempio: LBA M_1 per $B = \{w\#w \mid w \in \{0,1\}^*\}$

Diagramma di stato per la macchina di Turing M_1



Teorema

La classe dei linguaggi di tipo $i+1$ è contenuta strettamente in quella dei linguaggi di tipo i, $0 \leq i \leq 2$.

Teorema

La classe dei linguaggi di tipo $i+1$ è contenuta strettamente in quella dei linguaggi di tipo i, $0 \leq i \leq 2$.

Per provare questo teorema occorre esibire:

Teorema

La classe dei linguaggi di tipo $i+1$ è contenuta strettamente in quella dei linguaggi di tipo i, $0 \leq i \leq 2$.

Per provare questo teorema occorre esibire:

- Un linguaggio generato da una grammatica di tipo zero che non può essere generato da una grammatica dipendente dal contesto.

Teorema

La classe dei linguaggi di tipo $i+1$ è contenuta strettamente in quella dei linguaggi di tipo i, $0 \leq i \leq 2$.

Per provare questo teorema occorre esibire:

- Un linguaggio generato da una grammatica di tipo zero che non può essere generato da una grammatica dipendente dal contesto.
- Un linguaggio generato da una grammatica dipendente dal contesto che non può essere generato da una grammatica libera dal contesto.

Teorema

La classe dei linguaggi di tipo $i+1$ è contenuta strettamente in quella dei linguaggi di tipo i, $0 \leq i \leq 2$.

Per provare questo teorema occorre esibire:

- Un linguaggio generato da una grammatica di tipo zero che non può essere generato da una grammatica dipendente dal contesto.
- Un linguaggio generato da una grammatica dipendente dal contesto che non può essere generato da una grammatica libera dal contesto.
- Un linguaggio generato da una grammatica libera dal contesto che non può essere generato da una grammatica regolare.

Alla luce di quanto visto si tratta di esibire:

Alla luce di quanto visto si tratta di esibire:

- Un linguaggio Turing riconoscibile che non sia dipendente dal contesto, cioè che non può essere generato da una grammatica dipendente dal contesto.

Alla luce di quanto visto si tratta di esibire:

- Un linguaggio Turing riconoscibile che non sia dipendente dal contesto, cioè che non può essere generato da una grammatica dipendente dal contesto.
- Un linguaggio dipendente dal contesto che non sia context-free.

Alla luce di quanto visto si tratta di esibire:

- Un linguaggio Turing riconoscibile che non sia dipendente dal contesto, cioè che non può essere generato da una grammatica dipendente dal contesto.
- Un linguaggio dipendente dal contesto che non sia context-free.
- Un linguaggio context-free che non sia regolare.

Un linguaggio Turing riconoscibile che non è dipendente dal contesto, cioè che non può essere generato da una grammatica dipendente dal contesto è il linguaggio

Un linguaggio Turing riconoscibile che non è dipendente dal contesto, cioè che non può essere generato da una grammatica dipendente dal contesto è il linguaggio

$$A_{TM} = \{\langle M, w\rangle \mid M \text{ è una TM e } M \text{ accetta } w\}$$

Un linguaggio Turing riconoscibile che non è dipendente dal contesto, cioè che non può essere generato da una grammatica dipendente dal contesto è il linguaggio

$$A_{TM} = \{\langle M, w\rangle \mid M \text{ è una TM e } M \text{ accetta } w\}$$

A_{TM} è un linguaggio di tipo 0 perché A_{TM} è Turing riconoscibile.

Un linguaggio Turing riconoscibile che non è dipendente dal contesto, cioè che non può essere generato da una grammatica dipendente dal contesto è il linguaggio

$$A_{TM} = \{\langle M, w\rangle \mid M \text{ è una TM e } M \text{ accetta } w\}$$

A_{TM} è un linguaggio di tipo 0 perché A_{TM} è Turing riconoscibile.

A_{TM} non è decidibile. Dunque A_{TM} non è dipendente dal contesto perché i linguaggi dipendenti dal contesto sono tutti decidibili.

Un esempio di linguaggio dipendente dal contesto che non è context-free:

$$B = \{a^n b^n c^n \mid n \geq 1\}$$

Un esempio di linguaggio dipendente dal contesto che non è context-free:

$$B = \{a^n b^n c^n \mid n \geq 1\}$$

$B = L(G)$, dove $G = (V, T, P, S)$, con $V = \{S, B\}$,
$T = \{a, b, c\}$ e P che consiste nelle seguenti produzioni

Un esempio di linguaggio dipendente dal contesto che non è context-free:

$$B = \{a^n b^n c^n \mid n \geq 1\}$$

$B = L(G)$, dove $G = (V, T, P, S)$, con $V = \{S, B\}$, $T = \{a, b, c\}$ e P che consiste nelle seguenti produzioni

1. $S \rightarrow aSBc \mid abc$

Un esempio di linguaggio dipendente dal contesto che non è context-free:

$$B = \{a^n b^n c^n \mid n \geq 1\}$$

$B = L(G)$, dove $G = (V, T, P, S)$, con $V = \{S, B\}$, $T = \{a, b, c\}$ e P che consiste nelle seguenti produzioni

1. $S \to aSBc \mid abc$
2. $cB \to Bc$

Un esempio di linguaggio dipendente dal contesto che non è context-free:

$$B = \{a^n b^n c^n \mid n \geq 1\}$$

$B = L(G)$, dove $G = (V, T, P, S)$, con $V = \{S, B\}$, $T = \{a, b, c\}$ e P che consiste nelle seguenti produzioni

1. $S \to aSBc \mid abc$
2. $cB \to Bc$
3. $bB \to bb$

Un esempio di linguaggio dipendente dal contesto che non è context-free:

$$B = \{a^n b^n c^n \mid n \geq 1\}$$

$B = L(G)$, dove $G = (V, T, P, S)$, con $V = \{S, B\}$, $T = \{a, b, c\}$ e P che consiste nelle seguenti produzioni

1. $S \to aSBc \mid abc$
2. $cB \to Bc$
3. $bB \to bb$

G è di tipo 1.

Un esempio di linguaggio dipendente dal contesto che non è context-free:

$$B = \{a^n b^n c^n \mid n \geq 1\}$$

$B = L(G)$, dove $G = (V, T, P, S)$, con $V = \{S, B\}$, $T = \{a, b, c\}$ e P che consiste nelle seguenti produzioni

1. $S \to aSBc \mid abc$
2. $cB \to Bc$
3. $bB \to bb$

G è di tipo 1.

Inoltre, utilizzando il pumping lemma è possibile provare che il linguaggio $B = \{a^n b^n c^n \mid n \geq 1\}$ non è context-free.

Un linguaggio context-free che non è regolare è

$$C = \{a^n b^n \mid n \geq 1\}$$

Un linguaggio context-free che non è regolare è

$$C = \{a^n b^n \mid n \geq 1\}$$

Utilizzando il pumping lemma è possibile provare che il linguaggio C non è regolare.

Un linguaggio context-free che non è regolare è

$$C = \{a^n b^n \mid n \geq 1\}$$

Utilizzando il pumping lemma è possibile provare che il linguaggio C non è regolare.

C è context-free. È generato dalla grammatica di tipo 2 definita dalle produzioni

$$S \to aSb \mid ab$$

Un linguaggio context-free che non è regolare è

$$C = \{a^n b^n \mid n \geq 1\}$$

Utilizzando il pumping lemma è possibile provare che il linguaggio C non è regolare.

C è context-free. È generato dalla grammatica di tipo 2 definita dalle produzioni

$$S \to aSb \mid ab$$

Quindi la classe dei linguaggi regolari è inclusa strettamente nella classe dei linguaggi context-free.

Abbiamo visto un risultato più preciso.

Abbiamo visto un risultato più preciso.
La classe dei linguaggi regolari è inclusa strettamente nella classe dei linguaggi context-free deterministici. Un linguaggio context-free deterministico che non è regolare è

$$\{wcw^R \mid w \in \{a, b\}^*\}$$

Abbiamo visto un risultato più preciso.
La classe dei linguaggi regolari è inclusa strettamente nella classe dei linguaggi context-free deterministici. Un linguaggio context-free deterministico che non è regolare è

$$\{wcw^R \mid w \in \{a,b\}^*\}$$

Inoltre la classe dei linguaggi context-free deterministici è inclusa strettamente nella classe dei linguaggi context-free non ambigui. Un linguaggio context-free non ambiguo che non è context-free deterministico è

$$L_{wwr} = \{w\, w^R \mid w \in \{0,1\}^*\}.$$

Abbiamo visto un risultato più preciso.
La classe dei linguaggi regolari è inclusa strettamente nella classe dei linguaggi context-free deterministici. Un linguaggio context-free deterministico che non è regolare è

$$\{wcw^R \mid w \in \{a, b\}^*\}$$

Inoltre la classe dei linguaggi context-free deterministici è inclusa strettamente nella classe dei linguaggi context-free non ambigui. Un linguaggio context-free non ambiguo che non è context-free deterministico è

$$L_{wwr} = \{w\, w^R \mid w \in \{0, 1\}^*\}.$$

Infine la classe dei linguaggi context-free non ambigui è inclusa strettamente nella classe dei linguaggi context-free.
Un linguaggio context-free inerentemente ambiguo è

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

Linguaggi context-sensitive e linguaggi decidibili

La classe dei linguaggi context-sensitive è inclusa propriamente nella classe dei linguaggi decidibili. Questo risultato è una conseguenza della proposizione seguente.

Proposizione

Sia $M_0, M_1, \ldots$ un insieme enumerabile di Macchine di Turing che si fermano su ogni input. Esiste un linguaggio decidibile L tale che $L \neq L(M_j)$ per ogni j.

Sia $M_0, M_1, \ldots$ un insieme enumerabile di Macchine di Turing che si fermano su ogni input. Quindi possiamo supporre che esiste una funzione calcolabile biettiva h tale che $M_0 = h(0)$, $M_1 = h(1)$ e in generale $M_i = h(i)$.

Anche Σ^* è enumerabile, sia g una biezione di $\mathbb{N}$ in Σ^* e siano $w_0, w_1, \ldots$ gli elementi di Σ^*, con $w_i = g(i)$.

Sia $\mathcal{B}$ l'insieme delle sequenze binarie infinite
cioè delle sequenze infinite di 0 e 1.

Sia $\mathcal{B}$ l'insieme delle sequenze binarie infinite
cioè delle sequenze infinite di 0 e 1.

È possibile associare a ogni linguaggio L su Σ una sequenza infinita s_L (la sequenza caratteristica di L) così definita:

Sia $\mathcal{B}$ l'insieme delle sequenze binarie infinite
cioè delle sequenze infinite di 0 e 1.

È possibile associare a ogni linguaggio L su Σ una sequenza infinita s_L (la sequenza caratteristica di L) così definita:

il bit i-esimo di s_L è 1 se l'i-esima stringa w_i è in L, il bit i-esimo di s_L è 0 se $w_i \notin L$.

Sia $\mathcal{B}$ l'insieme delle sequenze binarie infinite
cioè delle sequenze infinite di 0 e 1.

È possibile associare a ogni linguaggio L su Σ una sequenza infinita s_L (la sequenza caratteristica di L) così definita:

il bit i-esimo di s_L è 1 se l'i-esima stringa w_i è in L, il bit i-esimo di s_L è 0 se $w_i \notin L$.

L'applicazione $f : \mathcal{P}(\Sigma^*) \rightarrow \mathcal{B}$ definita da $f(L) = s_L$ è biettiva.

	w_0	w_1	$\ldots$	w_j	
M_0	0	1	$\ldots$	0	$\ldots$
M_1	1	1	$\ldots$	1	$\ldots$
M_2	0	0	$\ldots$	1	$\ldots$
.	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
M_i	1	0	$\ldots$	0	$\ldots$
.	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
.	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$

	w_0	w_1	$\dots$	w_j	
M_0	0	1	$\dots$	0	$\dots$
M_1	1	1	$\dots$	1	$\dots$
M_2	0	0	$\dots$	1	$\dots$
.	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
M_i	1	0	$\dots$	0	$\dots$
.	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
.	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$

Le TM M_i sono decider. Scriviamo 1 all'incrocio tra la riga corrispondente ad M_i e la colonna w_j se $w_j \in L(M_i) = L_i$, altrimenti scriviamo 0. Quindi la riga corrispondente ad M_i è la sequenza caratteristica di $L_i = L(M_i)$.

Definiamo L:

$$\forall i \geq 0 \quad w_i \in L \Leftrightarrow w_i \notin L_i$$

Definiamo L:

$$\forall i \geq 0 \quad w_i \in L \Leftrightarrow w_i \notin L_i$$

L è decidibile: per ogni stringa w_i generiamo M_i e decidiamo se $w_i \in L_i$. Ma per ogni $i \geq 0$, $L \neq L_i$.

Linguaggi context-sensitive e linguaggi decidibili

Definiamo L:

$$\forall i \geq 0 \quad w_i \in L \Leftrightarrow w_i \notin L_i$$

L è decidibile: per ogni stringa w_i generiamo M_i e decidiamo se $w_i \in L_i$. Ma per ogni $i \geq 0$, $L \neq L_i$.

Infatti, sia h tale che $L = L_h$. La domanda "$w_h \in L$?" conduce a una contraddizione.

Definiamo L:

$$\forall i \geq 0 \quad w_i \in L \Leftrightarrow w_i \notin L_i$$

L è decidibile: per ogni stringa w_i generiamo M_i e decidiamo se $w_i \in L_i$. Ma per ogni $i \geq 0$, $L \neq L_i$.

Infatti, sia h tale che $L = L_h$. La domanda "$w_h \in L$?" conduce a una contraddizione.

Assumiamo $w_h \in L$. Per la definizione di L deve essere $w_h \notin L_h$. Ma per ipotesi $L = L_h$, assurdo.

Definiamo L:

$$\forall i \geq 0 \quad w_i \in L \Leftrightarrow w_i \notin L_i$$

L è decidibile: per ogni stringa w_i generiamo M_i e decidiamo se $w_i \in L_i$. Ma per ogni $i \geq 0$, $L \neq L_i$.

Infatti, sia h tale che $L = L_h$. La domanda "$w_h \in L$?" conduce a una contraddizione.

Assumiamo $w_h \in L$. Per la definizione di L deve essere $w_h \notin L_h$. Ma per ipotesi $L = L_h$, assurdo.

Assumiamo $w_h \notin L$. Per la definizione di L deve essere $w_h \in L_h$. Ma per ipotesi $L = L_h$, assurdo.

Teorema

Esiste un linguaggio decidibile che non è context-sensitive.

Prova (cenni).

Possiamo enumerare le grammatiche context-sensitive.

Sia G_j una enumerazione di tali grammatiche e sia M_j il corrispondente decider per $L(G_j)$.

La conclusione discende dalla precedente proposizione.