Lezione 16 – grammatiche di tipo 1 e introduzione alle grammatiche di tipo 2

Lezione del 09/04/2025

Grammatiche e macchine di Turing

- **OSSERVAZIONE**: se L(G) contiene un numero infinito di parole allora, poiché la macchina NT_G derivata nel corso della dimostrazione del Teorema G.5 non rigetta le parole contenenti qualche non terminale di G, allora ogni computazione di NT_G contiene qualche computazione deterministica che non termina
- Infatti,
  - ogni computazione della macchina NT_G non è altro che una sequenza (non deterministica) di operazioni ***applica una produzione*** **–** ***verifica se la parola ottenuta coincide con la parola ricevuta in input***
  - che continua fino a quando l'applicazione di una produzione non porta a una parola che coincide con la parola ricevuta in input
  - e quindi se $x \notin L(G)$, per poter rigettare $NT_G(x)$ deve continuare ad ''applicare produzioni'' fino a quando non ha confrontato x con *tutte* le parole in L(G)
  - così che se L(G) contiene un numero infinito di parole
  - la computazione $NT_G(x)$ non ha termine!
  - ossia, non tutte le computazioni deterministiche di $NT_G(x)$ rigettano
- Dunque, in conclusione, **quando $x \notin L(G)$ $NT_G(x)$ non rigetta**

Grammatiche di tipo 1 e macchine di Turing

- Riflettiamo: quale caratteristica di una grammatica G di tipo 0 impedisce alla macchina NT_G di rigettare?
  - Il fatto che, per ogni parola x, esiste sempre una computazione di $NT_G(x)$ che non termina mai
  - e questo dipende dal fatto che, fino a quando la parola generata contiene qualche simbolo non terminale, NT_G non può sapere se, prima o poi, non verrà generata proprio x
  - perché, anche se la parola generata a un certo passo della computazione è più lunga di x – magari molto più lunga di x
  - nulla garantisce che, prima o poi, non si possano applicare produzioni che riducano la lunghezza della parola generata a quel passo
    - **perché una grammatica di tipo 0 ammette produzioni in cui la parte sinistra è più lunga della parte destra**
  - così che, riducendo e riducendo la lunghezza, non si giunga a x!
- Ma, come sappiamo bene, produzioni che diminuiscano la lunghezza della parola generata sono proibite nelle grammatiche di tipo 1!

Grammatiche di tipo 1 e macchine di Turing

- Produzioni che diminuiscano la lunghezza della parola generata sono proibite nelle grammatiche di tipo 1
- questo significa che, se G = $\langle V_T, V_N, P, S \rangle$ è una grammatica di tipo 1 e y è una parola in $(V_T \cup V_N)^*$ tale che $S \Longrightarrow_G y$, da y non è possibile derivare in G alcuna parola x la cui lunghezza è inferiore a quella di x (ossia, $|x| < |y|$)
- e questo implica che possiamo progettare una macchina di Turing (non deterministica) NT'_G che ***decide*** se una parola x è generata da una grammatica G di tipo 1 (ossia, se $x \in L(G)$) modificando la macchina NT_G introdotta nel **Teorema G.5** in modo tale che ciascuna computazione deterministica in $NT_G(x)$ rigetti non appena viene generata una parola che contiene più caratteri di x
  - oppure non appena riconosce di essere entrata in loop...
    - ????

vedi pagina seguente

Grammatiche di tipo 1 e loop

- Consideriamo la seguente grammatica di tipo 1 G = $\langle V_T, V_N, P_G, S \rangle$:

  V_T = { a,b,c}, V_N = { S,A },
  P = { 1) S → ASb , 2) S → A , 3) S → b, 4) A → a, 5) A → S }

- sia x = abab
- le computazioni $NT_G(x)$ della macchina NT_G (descritta nel **Teorema G.5**) e $NT'_G(x)$ della modifica NT'_G di NT_G (descritta alla pagina precedente) contengono la computazione deterministica

  S → ASb → SSb → bSb → bAb → bSb → bAb → bSb → ...

- ossia, una sequenza infinita di applicazioni alternate delle produzioni 2) e 5)
- **nella quale la lunghezza della parola generata non aumenta**
  - così da impedire la terminazione di $NT'_G(x)$ in virtù della lunghezza eccessiva
  - e risultando in una computazione di lunghezza non limitata – un loop!
- Conseguentemente, poiché x ∉ L(G) e dunque nessuna computazione deterministica di $NT'_G(x)$ accetta, la computazione $NT'_G(x)$ non termina
- ossia, $NT'_G(x)$ non rigetta

Grammatiche di tipo 1 e loop

- Dunque, se vogliamo progettare una macchina che rigetti le parole non appartenenti a L(G) (di tipo 1) dobbiamo fare in modo che essa riconosca di essere entrata in loop
- E come si fa?! Facile: contando il massimo numero di parole nell'alfabeto $(V_T \cup V_N)$ di una data lunghezza!
- Ricordiamo che esistono $|V_T \cup V_N|^k$ parole di lunghezza k nell'alfabeto $(V_T \cup V_N)$
- allora, quando una computazione deterministica di $NT'_G(x)$ ha già generato $|V_T \cup V_N|^k$ parole di lunghezza k
  - senza, dunque, aver mai generato una parola di lunghezza maggiore di k
- e genera, nuovamente una parola di lunghezza k, questo significa che sta generando una parola che aveva già generato
  - ossia, possiamo esser certi che ha generato un loop
- e, poiché deve essere $k \leq |x|$, possiamo esser certi che è stato generato un loop quando sono state generate $|V_T \cup V_N|^{|x|}+1$ parole della stessa lunghezza

Grammatiche di tipo 1, loop e macchine di Turing

- Allora possiamo fare così: in aggiunta ai tre nastri già descritti per la macchina $\mathrm{NT_G}$ (e con lo stesso uso che avevano in quella macchina) possiamo dotare la macchina $\mathrm{NT_G^1}$ deputata a decidere il linguaggio L(G) generato da una grammatica G di tipo 1 di due nastri ulteriori
  - il quarto nastro conterrà, in unario, il valore $|V_T \cup V_N|^{|x|}$
  - il quinto nastro ''conterà'' in unario quante parole della stessa lunghezza sono state generate ad un dato istante; esso sarà inizializzato a 1 e, con una parola y scritta sul primo nastro, ogni volta che viene generata (non deterministicamente) una parola z: viene aggiunto un 1 a quelli che già contiene se $|z|=|y|$, mentre viene resettato a 1 se $|z|>|y|$
  - In questo modo, quando durante una delle computazioni deterministiche il valore scritto sul quinto nastro è maggiore del valore scritto sul quarto nastro questo significa che $\mathrm{NT_G^1}$ è tornata a scrivere una parola y che aveva già scritto in un passo precedente e quindi la computazione deterministica può essere interrotta perché
    - nell'interpretazione del non determinismo come macchina parallela: le altre parole generabili da y sono state generate in altre computazioni deterministiche
    - nell'interpretazione come macchina dotata di genio: la scelta migliore che ha saputo fare il genio è stata quella di generare un loop

Grammatiche di tipo 1 e macchine di Turing

- In dettaglio, la macchina di Turing $\mathrm{NT}_{\mathrm{G}}^{1}$ opera come segue:
  - sia x scritto sul primo nastro di $\mathrm{NT}_{\mathrm{G}}^{1}$,
  - INIZIALIZZAZIONE: $\mathrm{NT}_{\mathrm{G}}^{1}$ scrive S sul secondo nastro e 1 sul terzo nastro; poi calcola $|V_T \cup V_N|^{|x|}$ in unario* e lo scrive sul quarto nastro e scrive 1 sul quinto nastro; infine, posiziona le testine sui simboli non blank più a sinistra di ciascun nastro
  - ITERAZIONE:
    - 1) GENERAZIONE: [...]
    - 2) VERIFICA: [...]
      - segue alla pagina seguente

* la macchina di Turing di tipo trasduttore che calcola la funzione f(n,k) = nk viene descritta in una dispensa relativa alle *funzioni time-constructible* che verrà resa disponibile quando si affronterà tale argomento

Grammatiche di tipo 1 e macchine di Turing

- In dettaglio, la macchina di Turing $\mathrm{NT}_{\mathrm{G}}^{1}$ opera come segue:
  - sia x scritto sul primo nastro di $\mathrm{NT}_{\mathrm{G}}^{1}$,
  - INIZIALIZZAZIONE: [alla pagina precedente]
  - ITERAZIONE:
    - 1) GENERAZIONE: $\mathrm{NT}_{\mathrm{G}}^{1}$ simula non deterministicamente l'applicazione di tutte le produzioni possibili alla parola scritta sul secondo nastro a partire dalla posizione indicata sul terzo nastro
      - in tal caso, modifica la parola sul secondo nastro e: se la produzione applicata non aumenta il numero di caratteri della parola generata allora somma in binario 1 al valore contenuto sul quinto nastro altrimenti ri-inizializza a 1 il contenuto del quinto nastro

      oppure, se il carattere scritto sul secondo nastro alla posizione indicata sul terzo nastro è blank, non applica alcuna produzione
      - in tal caso, incrementa di 1 il valore sul terzo nastro
    - 2) VERIFICA: [...]
      - segue alla pagina seguente

Grammatiche di tipo 1 e macchine di Turing

- In dettaglio, la macchina di Turing $\mathrm{NT}_{\mathrm{G}}^{1}$ opera come segue:
  - sia x scritto sul primo nastro di $\mathrm{NT}_{\mathrm{G}}^{1}$,
  - INIZIALIZZAZIONE: [alle pagine precedenti]
  - ITERAZIONE:
    - 1) GENERAZIONE: [alla pagina precedente]
    - 2) VERIFICA: ogni volta che la parola sul secondo nastro viene modificata:
      - se essa coincide con quella scritta sul primo nastro $\mathrm{NT}_{\mathrm{G}}^{1}$ **accetta**,
      - altrimenti se la parola così ottenuta non contiene alcun non terminale o se ad essa non può essere applicata alcuna produzione oppure **se è più lunga della parola x** oppure **se il valore scritto sul quinto nastro è maggiore di quello scritto sul quarto nastro** allora $\mathrm{NT}_{\mathrm{G}}^{1}$ **rigetta**,
      - altrimenti $\mathrm{NT}_{\mathrm{G}}^{1}$ esegue un nuovo passo non deterministico ri-inizializzando a 1 il contenuto del terzo nastro e riposizionando a sinistra la testina sul quarto nastro

Grammatiche di tipo 1 e macchine di Turing

- Poiché NT_G^1 accetta una parola x se e soltanto se G genera x, allora possiamo concludere che NT_G^1 accetta L(G)

- Per dimostrare che NT_G^1 decide L(G) non resta dunque che mostrare che NT_G^1 rigetta ogni parola non appartenente a L(G)

Grammatiche di tipo 1 e macchine di Turing

- Per dimostrare che NT_G^1 decide L(G) non resta dunque che mostrare che NT_G^1 rigetta ogni parola non appartenente a L(G)
- sia, allora, x una parola non appartenente a L(G) e sia y ∈ $(V_T \cup V_N)^*$ la parola scritta sul nastro ad un certo passo di una computazione deterministica di NT_G^1(x):
  - 1) se y contiene solo caratteri terminali, poiché per costruzione di NT_G^1, S $\Longrightarrow_G$ y allora y ∈L(G) e dunque, poiché x ∉ L(G), y ≠ x e la computazione deterministica di NT_G^1(x) rigetta
  - 2) se y contiene almeno un carattere non terminale e |y|>|x| allora la computazione deterministica di NT_G^1(x) rigetta
  - 3) se y contiene almeno un carattere non terminale e |y|≤|x| allora NT_G^1 sostituisce (non deterministicamente) y con un'altra parola z tale che |z| ≥ |y| fino a quando non viene generata una parola che cade nei casi 1) o 2) o fino a quando sono state generate $|V_T \cup V_N|^{|x|}$ +1 parole di lunghezza |y|
  - dunque, tutte le la computazioni deterministiche di NT_G^1(x) originate da y rigettano
  - in conclusione, tutte le la computazioni deterministiche di NT_G^1(x) rigettano
  - ossia, NT_G^1(x) rigetta

Grammatiche di tipo 1 e macchine di Turing

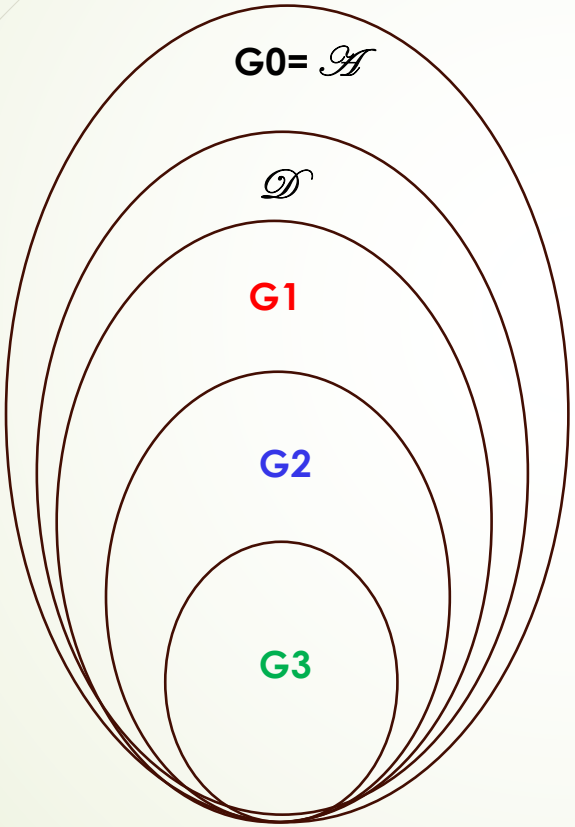
- Quel che abbiamo mostrato costruendo la macchina $\mathrm{NT}_{\mathrm{G}}^{1}$ dimostra il seguente
- **TEOREMA G.6**: *per ogni grammatica di tipo 1 G esiste una macchina di Turing che* ***decide*** *L(G)*
- che implica il seguente corollario

- **COROLLARIO**: *L'insieme dei linguaggi di tipo 1 è un sottoinsieme dei linguaggi* ***decidibili***
  - o **ricorsivi**

- OSSERVAZIONE: mentre i teoremi G.4 e G.5 implicano che un linguaggio è accettabile se e solo se esso è generato da una grammatica (ossia, gli insiemi dei linguaggi decidibili e l'insieme dei linguaggi generati da grammatiche coincidono), il teorema G.6 dimostra soltanto che tutti i linguaggi generati da grammatiche di tipo 1 sono decidibili, ma non il viceversa!

Una visione d'insieme

- Quel che abbiamo osservato è descritto in figura



G0 = insieme dei linguaggi di tipo 0
G1 = insieme dei linguaggi di tipo 1
G2 = insieme dei linguaggi di tipo 2
G3 = insieme dei linguaggi di tipo 3
$\mathscr{A}$= insieme dei linguaggi accettabili
$\mathscr{D}$= insieme dei linguaggi decidibili

G3 $\subseteq$ **G2** $\subseteq$ **G1** $\subseteq \mathscr{D} \subset$ **G0** = $\mathscr{A}$

Grammatiche context-free (di tipo 2)

- Ricordiamo che le **Grammatiche di tipo-2** (**grammatiche context-free**) possiedono solo produzioni nella forma $A \rightarrow \alpha$ con $A \in V_N$ e $\alpha \in (V_T \cup V_N)^*$
- I linguaggi generati da grammatiche di tipo 2 sono detti linguaggi context-free

- ESEMPIO: sia $G_{a=b,c} = \langle V_T, V_N, P, S \rangle$ la grammatica di tipo 2 così definita
  - V_T = {a, b, c} , V_N = {S, A, C}
  - P = {S → AC, A → aAb, A → ab, C → cC, C →c }
- il linguaggio generato da G è $L_{a=b,c} = L(G_{a=b,c}) = \{ a^n b^n c^m : n \geq 1 \wedge m \geq 1 \}$ che, dunque, è un linguaggio context-free

- ESERCIZIO: dimostrare che il linguaggio $L_{a,b=c} = \{ a^m b^n c^n : m \geq 1 \wedge n \geq 1 \}$ è context free
  - SOLUZIONE: S → aA, A → aA |B, B → bBc |bc