Lezione 15 – da mcchina di Turing a grammatica

Lezione del 04/04/2025

Grammatiche e macchine di Turing

- La Grammatica è un modello di calcolo
  - una grammatica descrive come si possono generare le parole appartenenti a un insieme di parole
- La Macchina di Turing è un modello di calcolo
  - una macchina di Turing descrive come si fa a riconoscere le parole appartenenti a un insieme di parole
- La scorsa lezione abbiamo dimostrato che ogni linguaggio accettabile può essere generato da una grammatica formale
- oggi dimostriamo che se un linguaggio L è generato da una grammatica (di tipo 0) allora esiste una macchina di Turing che accetta L
- Questo dimostrerà che

  l'insieme dei linguaggi accettabili coincide con l'insieme dei linguaggi generabili da grammatiche di tipo 0
- ossia,

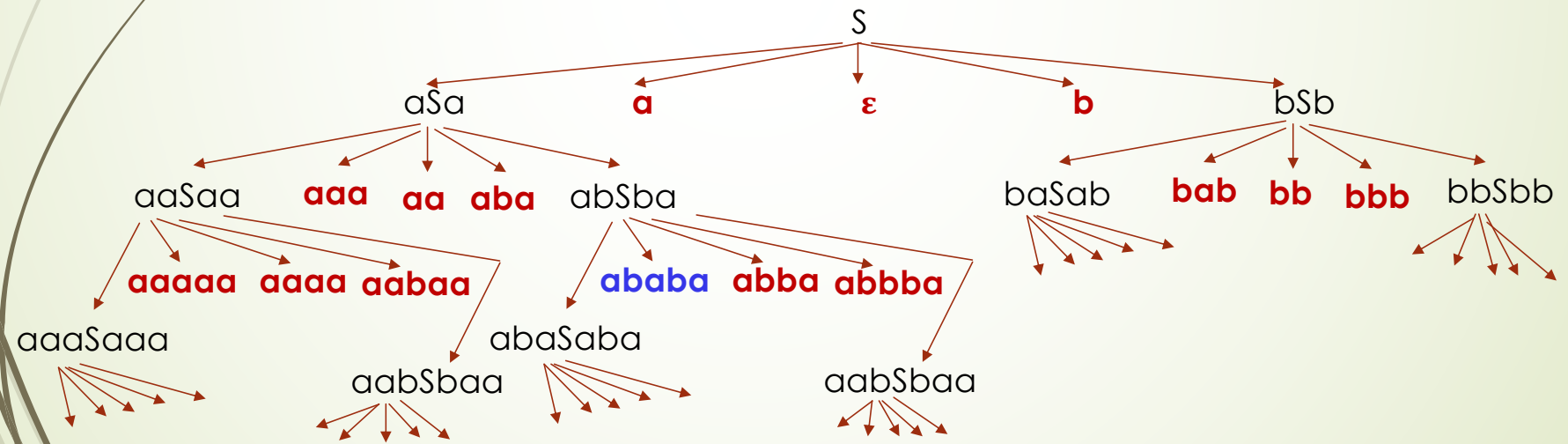
  le grammatiche di tipo 0 sono un modello di calcolo che soddisfa la Tesi di Church-Turing

Grammatiche e macchine di Turing

- **TEOREMA G.5**: per ogni grammatica G esiste una macchina di Turing che accetta L(G)
- DIMOSTRAZIONE: Sia G = $\langle V_T, V_N, P_G, S \rangle$ una grammatica
  - Definiamo, a partire da G, una macchina di Turing non deterministica NT che, con input x ∈ V_T^*, accetta se e soltanto se $S \Longrightarrow_G^* x$
  - A prima vista questo sembra un compito facile: definiamo una macchina a due nastri con grado di non determinismo pari a k = $|P_G|$:
    - all'inizio della computazione NT(x), sul primo nastro è scritto x e il secondo nastro è vuoto
    - il primo passo della computazione scrive S sul secondo nastro
    - poi, ad ogni passo, NT prova a simulare non deterministicamente l'applicazione di tutte le produzioni alla parola scritta sul secondo nastro modificando la parola quando ci riesce
      - ogni computazione deterministica verifica se la parola così ottenuta coincide con quella scritta sul primo nastro: in tal caso **accetta**, altrimenti se la parola così ottenuta non contiene alcun non terminale o ad essa non può essere applicata alcuna produzione **rigetta**, altrimenti esegue un nuovo passo non deterministico

Grammatiche e macchine di Turing

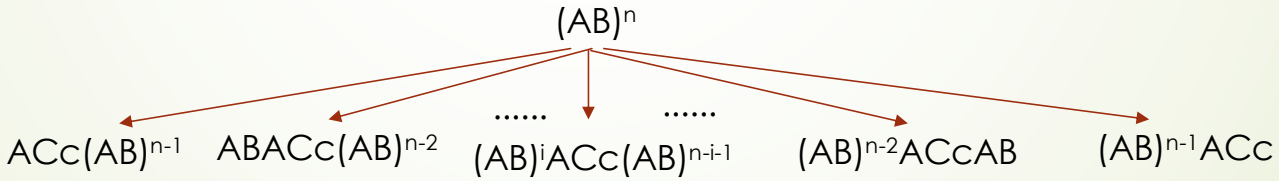
- **TEOREMA G.5**: per ogni grammatica G esiste una macchina di Turing che accetta L(G)
- DIMOSTRAZIONE: Vediamo un esempio di una macchina così costruita
  - G = $\langle V_T, V_N, P_G, S \rangle$ è la grammatica dell'ESEMPIO3, in cui V_T = { a,b }, V_N = { S }, e P = { S → aSa, S → bSb, S → a, S → b, S → ε }
  - sia x = ababa, allora NT(x) è la seguente computazione, in cui i nodi indicano il contenuto del secondo nastro e gli archi la produzione applicata



Grammatiche e macchine di Turing

- **TEOREMA G.5**: per ogni grammatica G esiste una macchina di Turing che accetta L(G)
- DIMOSTRAZIONE: Ma il procedimento non sempre funziona, infatti
  - sia G = $\langle V_T, V_N, P_G, S \rangle$ tale che V_T = { a,b,c}, V_N = { S,A,B,C }, e P = { S → ABS , S → ε , B → Cb, AC → Ca, BC → Cc , C → ε }
  - se x è abbastanza lunga, NT(x) contiene la seguente computazione deterministica:

    S → ABS → ABABS → ABABABS → ..... → ABABAB... $_{\text{n volte}}$ ... AB
  - a questo punto, può essere applicata applicata soltanto la produzione B → Cc
  - ma a ciascuna delle n B presenti nella parola
  - ossia, la computazione NT(x) dovrebbe contenere



  - ma se così fosse il grado di non determinismo di NT non sarebbe costante!

Grammatiche e macchine di Turing

- **TEOREMA G.5**: per ogni grammatica G esiste una macchina di Turing che accetta L(G)
- DIMOSTRAZIONE: Ma il procedimento non sempre funziona, infatti
  - Ricapitolando: in generale alla parola generata in n passi a partire dall'assioma possono essere applicate le produzioni di G <u>in molte posizioni diverse</u>
    - in un numero di posizioni non costante
  - e pertanto la macchina NT descritta in precedenza avrebbe grado di non determinismo non costante
    - ossia, NT non è una macchina di Turing non deterministica
  - Per trasformare NT in una macchina di Turing non deterministica NT_G dobbiamo apportare ad essa qualche modifica:
    - intanto, NT_G utilizza un terzo nastro sul quale è memorizzato un intero i
    - ad ogni passo della computazione prova ad applicare ciascuna produzione a partire dalla posizione i della parola generata fino a quel momento oppure a non applicare alcuna produzione a partire da quella posizione
  - Vediamo ora NT_G più in dettaglio

Grammatiche e macchine di Turing

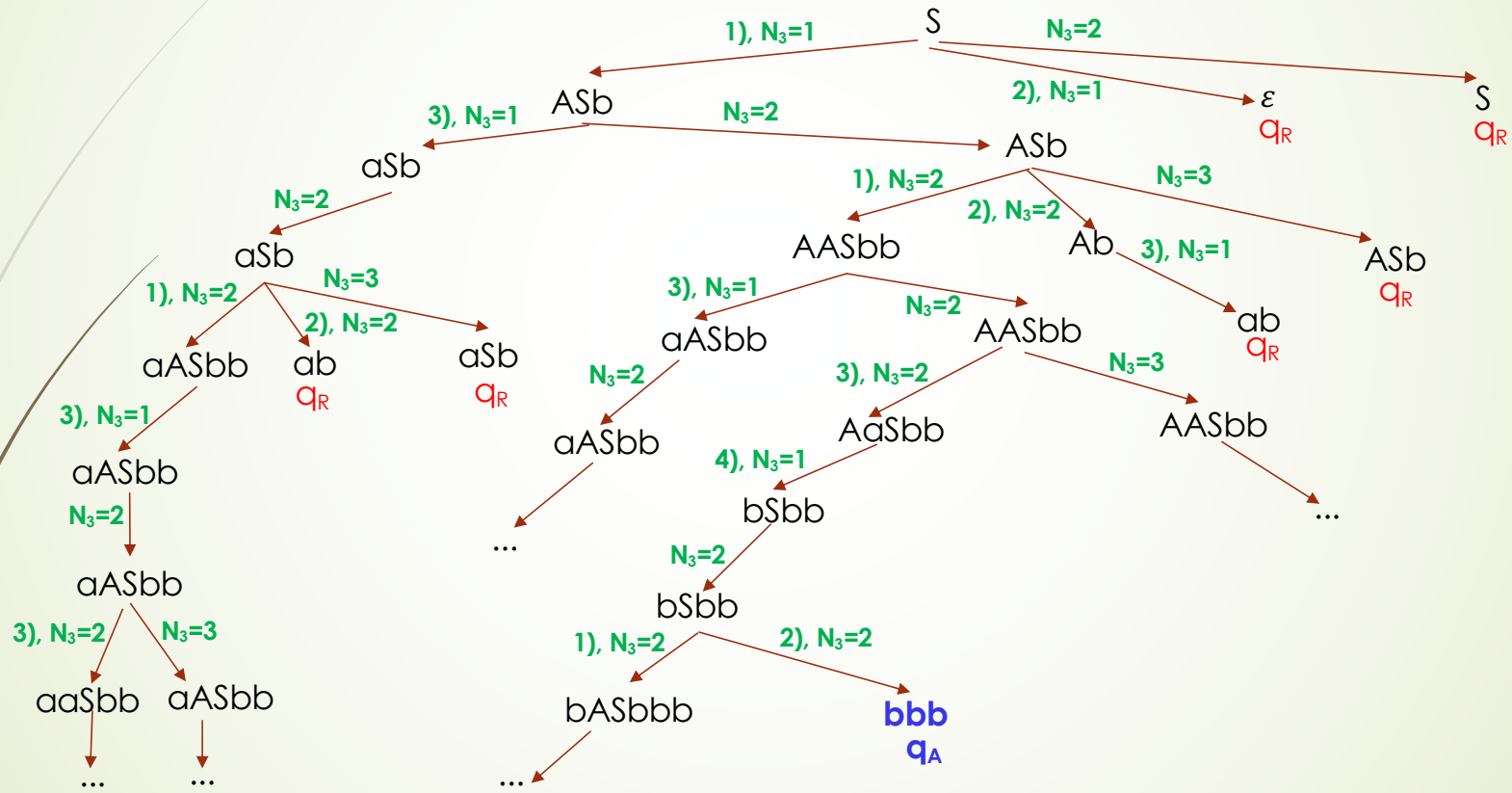
- **TEOREMA G.5**: per ogni grammatica G esiste una macchina di Turing che accetta L(G)
- DIMOSTRAZIONE: sia x scritto sul primo nastro di NT_G , allora:
  - INIZIALIZZAZIONE: NT_G scrive S sul secondo nastro e 1 sul terzo nastro
  - ITERAZIONE: NT_G simula non deterministicamente l'applicazione di tutte le produzioni possibili alla parola scritta sul secondo nastro **a partire dalla posizione indicata sul terzo nastro** (modificando la parola sul secondo nastro), e, in aggiunta, come ultima scelta non deterministica non applica alcuna produzione ma incrementa di 1 il valore sul terzo nastro
    - ogni volta che la parola sul secondo nastro viene modificata NT_G verifica se essa coincide con quella scritta sul primo nastro: in tal caso **accetta**, altrimenti se la parola così ottenuta non contiene alcun non terminale o se ad essa non può essere applicata alcuna produzione **rigetta**, altrimenti esegue un nuovo passo non deterministico ri-inizializzando a 1 il contenuto del terzo nastro

Grammatiche e macchine di Turing

- **TEOREMA G.5**: per ogni grammatica G esiste una macchina di Turing che accetta L(G)
- DIMOSTRAZIONE: sia x scritto sul primo nastro di NT_G , allora:
  - Vediamo un esempio di una computazione di NT_G
  - sia G = $\langle V_T, V_N, P_G, S \rangle$ tale che V_T = { a,b,c}, V_N = { S,A }, e P = { 1) S → ASb , 2) S → ε , 3) A → a, 4) Aa → b }
  - e sia x = bbb
  - nella pagina seguente viene mostrata *parte della computazione* $NT_G(x)$: per chiarezza, gli archi sono etichettati (in verde)
    - con il numero della produzione e il contenuto del nastro N_3, nel caso in cui una produzione venga applicata (esempio: **1), N_3=2**)
    - con il solo contenuto del nastro N_3 nel caso in cui esso venga modificato (senza applicare alcuna produzione) (esempio: **N_3=1**)
  - viene mostrata solo *parte della computazione*: osserviamo, infatti, che alcune computazioni deterministiche in $NT_G(x)$ non terminano!

Grammatiche e macchine di Turing

- **computazione** $NT_G(x)$:



Grammatiche e macchine di Turing

- **TEOREMA G.5**: per ogni grammatica G esiste una macchina di Turing che accetta L(G)
- DIMOSTRAZIONE:
  - ora NT_G è finalmente una macchina di Turing non deterministica
  - Infatti:
    - ad ogni passo simula non deterministicamente l'applicazione di tutte le produzioni possibili e infine incrementa il valore su N_3 senza simulare l'applicazione di alcuna produzione
  - pertanto, il grado di non determinismo di NT_G è $|P_G|+1$
  - e $|P_G|$ è una caratteristica della grammatica G
    - **non dipende dalla parola x input di NT_G!**
  - ossia, è **costante**!
  - E, quindi, $NT_G(x)$ ha grado di non determinismo costante
    - così come deve essere!

Grammatiche e macchine di Turing

- **TEOREMA G.5**: per ogni grammatica G esiste una macchina di Turing che accetta L(G)
- DIMOSTRAZIONE: resta da mostrare che NT_G accetta il linguaggio L(G) generato dalla grammatica G:
  - per costruzione, NT_G accetta una parola x se e soltanto se esiste una sequenza di produzioni che, applicate in una dopo l'altra a partire dall'assioma S permette di generare x
  - ossia, $NT_G(x)$ accetta se e soltanto se esiste in G una derivazione $S \Longrightarrow_G x_1 \Longrightarrow_G x_2 \ldots \Longrightarrow_G x_n = x$
  - ossia, $NT_G(x)$ accetta se e soltanto se $x \in L(G)$
  - e questo significa che NT_G accetta L(G)

QED

Grammatiche e macchine di Turing

- In conclusione, i Teoremi G.4 e G.5 dimostrano che

COROLLARIO: *la classe dei linguaggi generati da grammatiche di tipo 0 (linguaggi di tipo 0) coincide con la classe dei linguaggi* ***accettabili***

  - che, nel gergo dei linguaggi formali sono chiamati ***semi-decidibili***
  - o, ancora più rigorosamente, ***semi-ricorsivi***
  - o ***ricorsivamente enumerabili***

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 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!
  - *e la prossima lezione vedremo le conseguenze di questo vincolo*

Esercizio

- Terminiamo questa lezione con il seguente esercizio
- **ESERCIZIO**: progettare una grammatica G = $\langle V_T, V_N, P_G, S \rangle$ che generi il linguaggio L = { x ∈ {0,1}*: x contiene un numero pari di '1'}
- SOLUZIONE:
  - poniamo V_T = {0,1}, V_N = {S, U} e P è l'insieme delle seguenti produzioni
    - S → 0S | 1US | ε , U0 → 0U , U1 → 1U , e U → 1
  - !) se x ∈ L allora x è generata da G: infatti, se x contiene 2k '1' e h '0' allora
    - se x inizia con i ≥ 0 caratteri '0' allora $S \Longrightarrow_G 0^i (1U)^{2k} 0^{h-i}$ utilizzando le sole produzioni che hanno S come parte sinistra, e poi posizioniamo gli 'U' nelle posizioni in cui x contiene i suoi ultimi k '1' mediante le produzioni U0 → 0U e U1 → 1, e infine trasformiamo gli 'U' in '1' mediante la produzione U → 1
  - 2) se x è generata da G allora x ∈ L: infatti ogni '1' è generato congiuntamente a un 'U' e l'unico modo per rimuovere il non terminale 'U' dalla parola generata da G è applicare la produzione U → 1 risultando così in una parola di L(G) che contiene un numero pari di '1'