Lezione 19 – grammatiche regolari

Lezione del 16/04/2025

Grammatiche e linguaggi di programmazione

- I linguaggi di programmazione sono ‘‘grosso modo’’ linguaggi di tipo 2
- In effetti, le regole ‘‘grosso modo’’ di tipo 2 sono quelle che definiscono la sintassi vera e propria di un linguaggio di programmazione
  - ossia, quelle regole che permettono di affermare che un programma è sintatticamente corretto (che verificano, ad esempio, che ad ogni parentesi { corrisponda una })
- Tuttavia, la definizioni di alcune componenti di un programma risponde a regole che sono descritte da grammatiche di tipo 3
- e queste sono le componenti lessicali di un programma:
  - quelle che definiscono, ad esempio, i nomi delle variabili,
  - le parole chiave di un linguaggio di programmazione
  - ...
- e quindi la correttezza di questi elementi può essere verificata da algoritmi più semplici di quelli che si occupano della correttezza sintattica vera e propria
  - e che eseguono la cosiddetta analisi lessicale
  - basata su grammatiche di tipo 3

Grammatiche di tipo 3

- Ricordiamo che le **Grammatiche di tipo-3** (**grammatiche regolari**) possiedono solo produzioni nella forma forma A → a e A → aB oppure A → Ba con A, B ∈ V_N e a ∈ V_T (ma non sia A → aB che A → Ba)
  - nel seguito considereremo soltanto grammatiche regolari in cui tutte le produzioni sono nella forma A → a e A → aB

- I linguaggi generati da grammatiche di tipo 3 sono detti **linguaggi regolari**

- ESEMPIO: sia G = ⟨ V_T, V_N, P, S ⟩ la grammatica di tipo 3 così definita
  - V_T = {a, b} , V_N = {S, A, B}
  - P = {S → aA, A → aA, A → bB, B → bB, B →b }
  - è facile verificare che tale grammatica genera parole che iniziano con una sequenza non vuota di a e terminano con una sequenza di almeno 2 b
  - ossia, L(G)= { $a^h b^k$: h ≥ 1 ∧ k ≥ 2 }

Grammatiche regolari: pumping lemma

- Anche i linguaggi regolari hanno un loro pumping lemma – che descrive una proprietà soddisfatta da tutti i linguaggi linguaggi regolari
- **Pumping lemma per i linguaggi regolari:** per ogni linguaggio regolare L esiste un intero $p_L > 0$ (dipendente esclusivamente da L) tale che per ogni parola $z \in L$ con $|z| \geq p_L$ esistono tre parole u, v, w tali che
  1. z = uvw (z si può esprimere come concatenazione di u, v, w)
  2. $|uv| \leq p_L$
  3. $|v| \geq 1$ (v non può essere la parola vuota)
  4. $uv^h w$ è in L per ogni $h \geq 0$.
- Come per i linguaggi context-free, il pumping lemma stabilisce una *condizione necessaria* che deve essere soddisfatta da un linguaggio affinché esso sia regolare
  - ossia, *ogni linguaggio* regolare soddisfa 1., 2., 3. e 4.
- ma, di nuovo, tale condizione non è una condizione sufficiente

Grammatiche regolari: pumping lemma

- Come per i linguaggi context-free, il pumping lemma non è una *condizione sufficiente* per stabilire che un linguaggio è regolare
  - ossia, *esistono linguaggi* non regolari le cui parole soddisfano 1., 2., 3. e 4.
- per questa ragione, di nuovo, il pumping lemma si utilizza ''al negativo''
- ossia, per dimostrare che un linguaggio ***non*** è regolare
- E come si fa? Esattamente come si fa per i linguaggi context-free:
  - si mostra che le sue parole non soddisfano il lemma perché

  poiché ogni linguaggio regolare soddisfa il lemma, allora

  un linguaggio che non soddisfa il lemma non è regolare!
- E ora vediamo un esempio

Grammatiche regolari: pumping lemma

- **ESEMPIO**: dimostriamo che il linguaggio $L_{a=b} = \{ a^n b^n : n \geq 1 \}$ non è di tipo 3
  - supponiamo per assurdo che $L_{a=b}$ sia regolare e sia $p_L > 0$ la costante definita nel pumping lemma
  - consideriamo la parola $z = a^m b^m$ in $L_{a=b}$ con $m > p_L$
  - allora, per il lemma z può essere scritta nella forma uvw dove $|uv| \leq p_L$, e $v \neq \varepsilon$
  - poiché $|uv| \leq p_L < m$ allora v deve essere costituita da soli caratteri a, ossia, esistono due interi r e k tali che $u = a^r$, $v = a^k$, $w = a^{m-r-k}b^m$, $r+k \leq p_L$ e $k > 0$
  - inoltre, il lemma ci dice che $uv^h w$ è in $L_{a=b}$ per ogni $h \geq 0$
    - ***in particolare, per h= 0,*** **$uw = a^{m-k}b^m$** ***è in*** **$L_{a=b}$**
  - ma, poiché $k > 0$ (perché $v \neq \varepsilon$) per definizione di $L_{a=b}$ $uw = a^{m-k}b^m$ non appartiene a $L_{a=b}$: un assurdo!
  - Quindi $L_{a=b}$ non è regolare.
- Come conseguenza del pumping lemma e dell'esempio sopra e osservando che $L_{a=b}$ è un linguaggio context-free, otteniamo che

G3 $\subset$ **G2**

Grammatiche regolari: automi a stati finiti

- I linguaggi regolari sono un sottoinsieme dei linguaggi di tipo 2 e, pertanto, già sappiamo che essi sono decidibili – o anche accettabili da PDA
- Ci occupiamo ora di mostrare che, in effetti, i linguaggi regolari sono decisi da un modello di calcolo *strettamente meno potente* della Macchina di Turing di tipo riconoscitore (e anche del PDA): l'Automa a stati finiti
  - e questa volta sarà evidente dalla definizione che un automa a stati finiti è *strettamente meno potente* di una macchina di Turing e di un PDA!
- Informalmente, un automa a stati finiti è una macchina di Turing a un solo nastro *sul quale non può essere scritto altro che l'input*
  - la testina di lettura/scrittura è dunque, di fatto, una testina di sola lettura
  - a ogni istante, dipendentemente dallo stato interno e da ciò che è letto dalla testine, viene modificato lo stato interno come specificato dall'insieme delle quintuple (che, non devono modificare il carattere letto dalla testina) che, come per i PDA, descriviamo mediante una *funzione di transizione* δ
  - una computazione di un automa a stati finiti è una sequenza di cambiamenti di stato che termina non appena viene letto l'ultimo carattere della parola input: l'esito di una computazione è lo stato interno in cui termina tale computazione

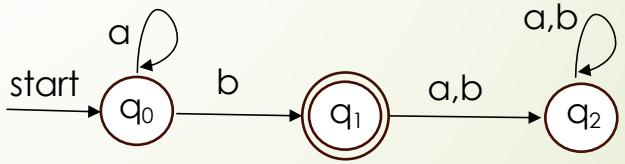
Grammatiche regolari: automi a stati finiti

- Formalmente: un **automa a stati finiti deterministico (ASFD)** è una quintupla $\langle \Sigma, Q, q_0, Q_F, \delta \rangle$ in cui Σ, Q, q_0 e F hanno lo stesso significato che nella Macchina di Turing e $\delta: Q \times \Sigma \rightarrow Q$ è la funzione ***totale*** di transizione che, ad ogni coppia (stato, carattere) associa un nuovo stato
- La funzione di transizione può essere descritta mediante una tabella

δ	a	b
q_0	q_0	q_1
q_1	q_2	q_2
q_2	q_2	q_2

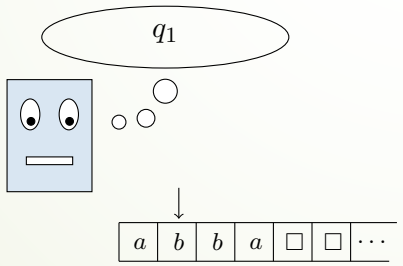
  - osserviamo che la notazione $\delta(q,a)=q'$ è equivalente alla quintupla $\langle q,a,a,q',D \rangle$

- oppure mediante un diagramma degli stati, che permette di rappresentare anche lo stato iniziale (mediante una freccia entrante marcata start) e gli stati finali (indicati con un doppio cerchio)



Grammatiche regolari: automi a stati finiti

- Poiché un automa a stati finiti è una particolare macchina di Turing, possiamo estendere agli automi a stati finiti le definizioni di stato globale, transizione, computazione, esito di una computazione
  - tuttavia, poiché la testina sul nastro di un automa non può scrivere e può muoversi soltanto a destra, tali definizioni risultano semplificate per gli automi a stati finiti
- una configurazione di un automa a stati finiti (l'analogo di uno stato globale di una macchina di Turing) consiste di una coppia (q,x) in cui q è uno stato interno dell'automa e x è la parola costituita dal carattere letto dalla testina e dai caratteri alla sua destra



configurazione (q_1,bba)

Grammatiche regolari: automi a stati finiti

- Poiché un automa a stati finiti è una particolare macchina di Turing, possiamo estendere agli automi a stati finiti le definizioni di stato globale, transizione, computazione, esito di una computazione
  - tuttavia, poiché la testina sul nastro di un automa non può scrivere e può muoversi soltanto a destra, tali definizioni risultano semplificate per gli automi a stati finiti
- una transizione di un automa a stati finiti da una configurazione (q,x) a una configurazione (q',y) si indica con (q,x) ⊢ (q',y) e avviene quando x=ay (per qualche carattere a dell'alfabeto) e δ(q,a) = q'

transizione (q_1,bba) ⊢ (q_2,ba)
(ossia, dalla configurazione (q_1,bba) alla configurazione (q_2,ba))

Grammatiche regolari: automi a stati finiti

- Data una parola $x=x_0x_1...x_n \in \Sigma^*$, la computazione di un ASFD a partire dalla *configurazione iniziale* (q_0,x) (in cui, cioè, lo stato interno è lo stato iniziale) è la sequenza di n transizioni

  $(q_0, x_0x_1...x_n) \vdash (q_1, x_1...x_n) \vdash (q_2, x_2...x_n) \vdash ... (q_{n-1}, x_{n-1}...x_n) \vdash (q_n, x_n) \vdash (q, \square)$

  in cui, per i= 0, n-1, $\delta(q_i,x_i) = q_{i+1}$ e $\delta(q_n,x_n) = q$
- tale sequenza di transizioni viene indicata con $(q_0, x_0x_1...x_n) \vdash^* (q, \square)$
- oppure, se definiamo ricorsivamente la funzione di transizione estesa δ^* come

  $\delta^*(q, \square) = q$

  $\delta^*(q, a_1...a_n) = \delta(\delta^*(q, a_1...a_{n-1}), a_n)$

  non in programma

  allora, la computazione di un automa a stati finiti a partire dalla *configurazione iniziale* (q_0,x) è $\delta^*(q_0, x) = \delta^*(q_0, x_0...x_n)$
  - osserviamo che $\delta^*(q_0, x)$ è uno stato interno dell'automa e che rappresenta l'esito della computazione di A sull'input x

Grammatiche regolari: automi a stati finiti

- **OSSERVAZIONE**: la funzione di transizione non è definita quando sul nastro viene letto blank (□)

- conseguentemente, tutte le computazione di un ASFD terminano
  - e terminano non appena è stata letta l'intera parola in input

- una parola x è **accettata** da un ASFD se **$q = \delta^*(q_0, x)$ è uno stato finale**
- e, poiché tutte le computazioni di un ASFD terminano, una parola x è **rigettata** da un ASFD quando **$q = \delta^*(q_0, x)$ non è uno stato finale**

Grammatiche regolari: automi a stati finiti

- una parola x è accettata da un ASFD se $q = \delta^*(q_0, x)$ è uno stato finale

- il linguaggio accettato da un ASFD $A = \langle \Sigma, Q, q_0, Q_F, \delta \rangle$ è l'insieme delle parole accettate da A, ossia

$$L(A) = \{ x \in \Sigma^* : (q_0, x) \vdash^* (q, \square) \text{ con } q \in Q_F \}$$

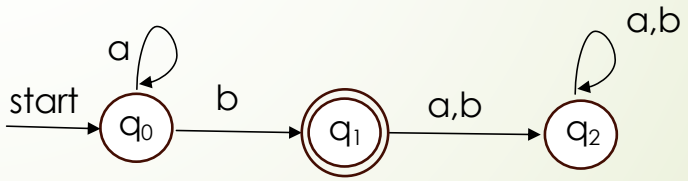
o, equivalentemente,

$$L(A) = \{ x \in \Sigma^* : \delta^*(q_0, x) \in Q_F \}$$

- sottolineiamo che, poiché le computazioni di un ASFD terminano sempre, **L(A) è in effetti il linguaggio deciso da A**

Grammatiche regolari: automi a stati finiti

- **ESEMPIO**: sia L = { $a^n b$: n ≥ 0 }; costruiamo un ASFD A = $\langle \Sigma, Q, q_0, Q_F, \delta \rangle$ che decide L
  - se nello stato (iniziale) q_0 legge a allora, affinché la parola in input appartenga a L, deve leggere necessariamente almeno un altro carattere (perché le parole in L terminano con una – e una sola – b): quindi, $\delta(q_0,a)= q_0$
  - se nello stato q_0 legge b allora, poiché le parole in L contengono una sola b che è il carattere finale di ogni parola in L: quindi, A deve entrare in un nuovo stato che è anche uno stato finale, ossia, $\delta(q_0,b)= q_1$ e q_1 è uno stato finale
  - se nello stato q_1 legge ancora caratteri a o b, allora certamente la parola non appartiene a L: quindi A entra in un nuovo stato che non è finale e rimane in esso fino al termine della computazione, ossia $\delta(q_1,a)= \delta(q_1,b)= q_2$ e $\delta(q_2,a)= \delta(q_2,b)= q_2$
  - dunque, ecco il diagramma degli stati di A:



Macchina di Turing vs Automa a stati finiti

- Abbiamo già osservato come un ASFD sia una particolare MdT che non può scrivere sul proprio nastro
- Tuttavia la descrizione di un ASFD differisce da quella di una macchina di Turing in due aspetti:
  - 1) mentre un ASFD è descritto mediante una funzione di transizione δ, una macchina di Turing è descritta mediante un insieme delle quintuple P:
    - SOLUZIONE: banalmente, eseguire la transizione **δ(q,a)=q'** è equivalente a eseguire la quintupla **⟨q,a,a,q',D ⟩** e, quindi, le due descrizioni sono equivalenti
  - 2) in un ASFD possono esistere transizioni che iniziano con uno stato finale (lo stato q_1 nella figura in una delle pagine precedenti) mentre non esistono quintuple il cui stato di partenza è uno stato finale:
    - questo si risolve facilmente, aggiungendo i due nuovi stati **q_A** e **q_R** all'insieme degli stati della macchina di Turing che modella un ASFD,
    - aggiungendo a P, **per ogni stato finale q** dell'automa, le quintuple **⟨q, □, □,q_A, F ⟩**
    - e, infine, aggiungendo a P, **per ogni stato <u>non</u> finale q** dell'automa, le quintuple **⟨q, □, □, q_R, F ⟩**

Macchina di Turing vs Automa a stati finiti

- Abbiamo già osservato come un ASFD sia una particolare MdT che non può scrivere sul proprio nastro
- Tuttavia la descrizione di un ASFD differisce da quella di una macchina di Turing in due aspetti:
  - 1) mentre un ASFD è descritto mediante una funzione di transizione δ, una macchina di Turing è descritta mediante un insieme delle quintuple
  - 2) in un ASFD possono esistere transizioni che iniziano con uno stato finale (lo stato q_2 nella figura in una delle pagine precedenti) mentre non esistono quintuple il cui stato di partenza è uno stato finale
- In definitiva, sono equivalenti l'ASFD $A = \langle \Sigma, Q_A, q_0, Q_F, \delta \rangle$ e la macchina di Turing $T = \langle \Sigma, \underline{Q}, q_0, \{q_A, q_R\}, P \rangle$ in cui $\underline{Q} = Q_A \cup \{q_A, q_R\}$ e

$$P = \{ \langle q,a,a,q',D \rangle : q \in Q_A - Q_F \wedge a \in \Sigma \wedge \delta(q,a)=q' \} \cup \{ \langle q, \square, \square, q_A, F \rangle : q \in Q_F \} \cup \{ \langle q, \square, \square, q_R, F \rangle : q \in Q - Q_F \}$$

  - ESERCIZIO: dimostrare l'equivalenza di A e T

Grammatiche regolari: automi a stati finiti

- Resta da mostrare che la classe dei linguaggi regolari coincide con la classe dei linguaggi decisi da automi a stati finiti
- la dimostrazione procederà in 4 passi
  - intanto, poiché sappiamo che le computazioni di un automa a stati finiti terminano sempre, dimostreremo che la classe dei linguaggi regolari coincide con la classe dei linguaggi accettati da automi a stati finiti deterministici, poi
  - PASSO 1) dimostreremo che per ogni automa a stati finiti deterministico A esiste una grammatica G_A tale che $L(A) = L(G_A)$
  - PASSO 2) per dimostrare l'inverso del PASSO 1 dovremo definire una classe più ampia di automi a stati finiti, gli *automi a stati finiti non deterministici*
  - PASSO 3) dimostreremo che per ogni grammatica G esiste un automa a stati finiti non deterministico NA_G esiste tale che $L(G) = L(NA_G)$
  - PASSO 4) dimostreremo che per ogni automa a stati finiti non deterministico NA esiste un automa a stati finiti deterministico A tale che L(A) = L(NA)

Grammatiche regolari e ASFD: PASSO 1

- **TEOREMA G.14**: per ogni ASFD A = $\langle \Sigma, Q, q_0, Q_F, \delta \rangle$ esiste una grammatica $G_A = \langle V_T, V_N, P, S \rangle$ tale che L(A) = L(G_A)
- Dimostrazione. 1) definiamo la grammatica G_A :
  - poniamo $V_T = \Sigma$;
  - sia Q = {q_0, q_1, ..., q_k}: poniamo V_N = {A_0, ..., A_k};
  - A_0 è l'assioma di G, ossia, S = A_0
  - per ogni a ∈ Σ e q_i , q_h∈ Q , se $\delta(q_i, a) = q_h$ aggiungiamo a P la produzione $A_i \rightarrow aA_h$
  - per ogni a ∈ Σ, q_i ∈ Q e q_h ∈ Q_F, se $\delta(q_i, a) = q_h$ aggiungiamo a P la produzione $A_i \rightarrow a$
  - se $\delta(q_0, \square) = q_h \in Q_F$ (cioè, se L(A) contiene ε) aggiungiamo a P la produzione $A_0 \rightarrow \varepsilon$

Grammatiche regolari e ASFD: PASSO 1

- **TEOREMA G.14**: per ogni ASFD A = $\langle \Sigma, Q, q_0, Q_F, \delta \rangle$ esiste una grammatica $G_A = \langle V_T, V_N, P, S \rangle$ tale che $L(A) = L(G_A)$
- Dimostrazione. 1) Abbiamo definito G_A

  2) Dimostriamo che se $x \in L(A)$ allora $x \in L(G_A)$:
  - sia $x = x_0 x_1 \ldots x_n$; se $x \in L(A)$ allora $\delta^*(q_0, x) = q_j \in Q_F$
  - allora esistono $q_{i_1}, q_{i_2}, \ldots, q_{i_n} \in Q$ tali che $\delta(q_0, x_0) = q_{i_1}$,

    per h=1, ..., n-1 $\delta(q_{i_h}, x_h) = q_{i_{h+1}}$, e

    $\delta(q_{i_n}, x_n) = q_j$;
  - allora, per come abbiamo definito G_A,

    per h=1, ..., n: $A_{i_h} \rightarrow x_h A_{i_{h+1}}$, e

    $A_{i_n} \rightarrow x_n$
  - dunque, $A_0 \rightarrow x_0 A_{i_1} \rightarrow x_0 x_1 A_{i_2} \rightarrow \ldots. \rightarrow x_0 \ldots x_{n-1} A_{i_n} \rightarrow x_0 \ldots x_{n-1} x_n$, ossia $x \in L(G_A)$

Grammatiche regolari e ASFD: PASSO 1

- **TEOREMA G.14**: per ogni ASFD A = $\langle \Sigma, Q, q_0, Q_F, \delta \rangle$ esiste una grammatica $G_A = \langle V_T, V_N, P, S \rangle$ tale che L(A) = L(G_A)
- Dimostrazione. 1) Abbiamo definito G_A e 2) dimostrato che $x \in L(A) \Longrightarrow x \in L(G_A)$

  3) Dimostriamo che se $x \in L(G_A)$ allora $x \in L(A)$:
  - sia $x = x_0x_1...x_n$; se $x \in L(G_A)$ allora $A_0 \Longrightarrow^* x$
  - allora esistono $A_{i_1}, A_{i_2}, ..., A_{i_n} \in V_N$ tali che $A_0 \rightarrow x_0 A_{i_1}$,

    per h=1, ..., n $A_{i_1} \rightarrow x_0 A_{i_{h+1}}$, e

    $A_{i_1} \rightarrow x_n$
  - allora, per come abbiamo definito G_A, $\delta(q_0, x_0) = q_{i_1}$,

    per h=1, ..., n-1 $\delta(q_{i_h}, x_h) = q_{i_{h+1}}$, e

    $\delta(q_{i_n}, x_n) \in Q_F$
  - dunque, $x \in L(A)$

QED