Automi a pila

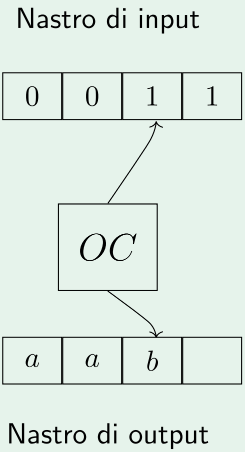
Dipartimento di Elettronica, Informazione e Bioingegneria
Politecnico di Milano

16 marzo 2020

Aumentiamo la potenza di un FSA

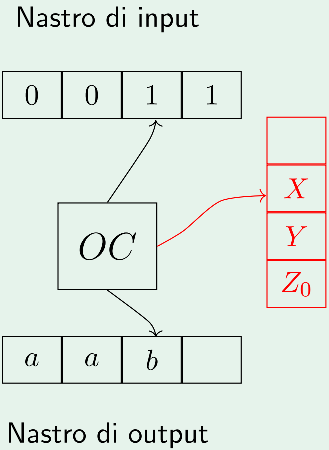
Descrizione operativa dell'automa a pila

- Un FSA ha un Organo di Controllo (OC) con memoria finita e un nastro di input infinito su cui non può scrivere
- Se traduttore ha un nastro di output in cui può solo scrivere
- La "memoria" dello stato del calcolo è finita



Aumentiamo la potenza di un FSA

Una memoria estesa



Memoria a pila:

- Infinita
- Accesso alla sola cima
- La lettura cancella

$\rightarrow$ Funzionamento LIFO

Automa a pila

Descrizione operativa

- L'automa a pila compie una mossa in funzione di:
  - Simbolo letto sulla pila
  - Stato corrente nell'FSA di controllo
  - Simbolo letto dall'ingresso (può non leggere nulla)
- L'automa a pila passa alla configurazione successiva:
  - cambiando stato nell'OC
  - sostituendo al simbolo in cima allo stack una stringa α di simboli (potenzialmente, $\alpha = \varepsilon$)
  - spostando (opzionalmente) la testina di lettura
  - se l'automa è un traduttore, scrivendo una stringa (potenzialmente nulla)

Riconoscitori e traduttori

Automa riconoscitore

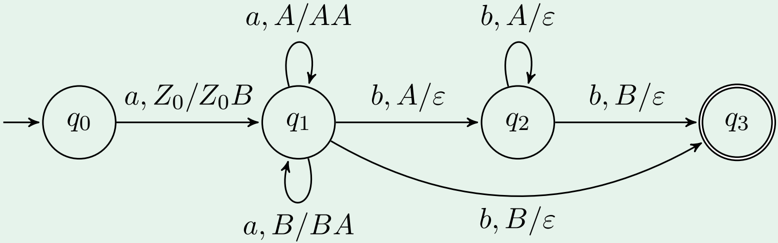
- La stringa x in ingresso è riconosciuta (accettata) se
  - L'automa scandisce completamente x
  - Una volta scandita tutta, lo stato dell'OC è di accettazione

Automa traduttore

- Se la stringa è accettata, il nastro di scrittura contiene la sua traduzione al termine del calcolo $\tau(x)$
- Se la x non è accettata la traduzione è indefinita $\tau(x) = \perp$

Esempio: Riconoscere $\{a^n b^n | n > 0\}$

Automa riconoscitore

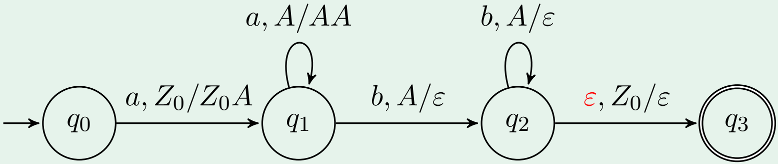


Convenzioni notazione

- $\langle$lettura input, cima della pila/riscrittura in pila$\rangle$
- Z_0 marcatore per il fondo della pila

Esempio: Riconoscere $\{a^n b^n | n > 0\}$

Un'alternativa

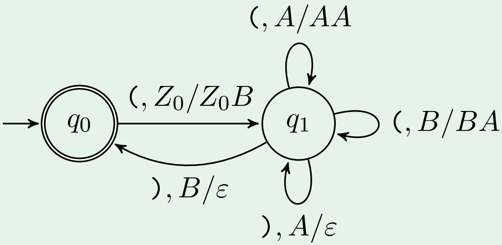


$\varepsilon - mossa$

- Questo automa effettua una mossa senza leggere dall'input
- Posso evitare di usare B come "marcatore della prima a"

Stringhe ben parentetizzate...

.. di sole parentesi tonde

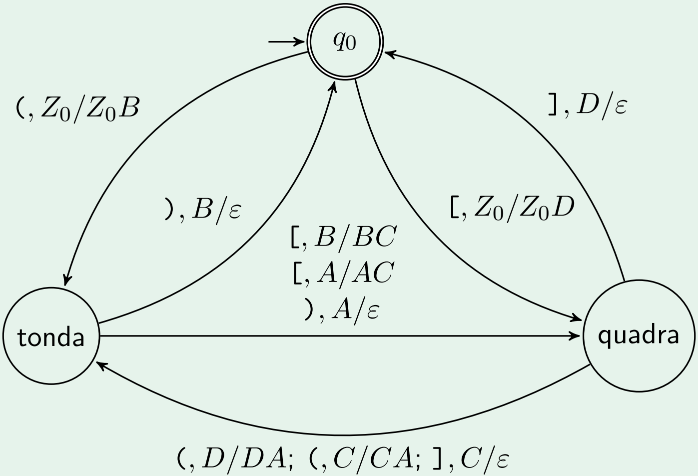


Note

- È una "semplificazione" del riconoscitore di $L = \{a^n b^n\}$
- Verifica solamente che il numero di a coincida con quello di b

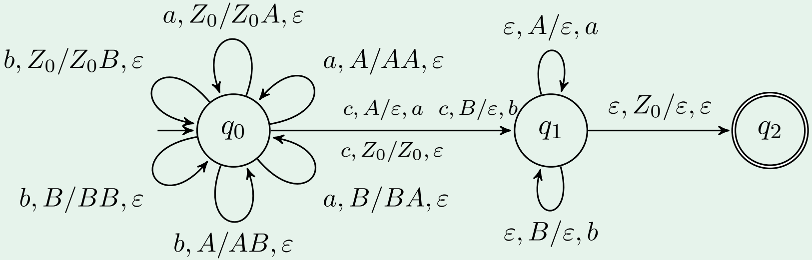
Stringhe ben parentetizzate...

.. di parentesi tonde e quadre



Un traduttore

Da $L_1 \subset \{a,b,c\}^*$ a $L_2 \subset \{a,b,c\}^*$



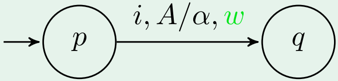
- Che traduzione effettua? (impila A e B fino alla prima c ...)

Formalizzazione

Riconoscitore e traduttore

- Automa [traduttore] a Pila : $\langle \mathbf{Q}, \mathbf{I}, \Gamma, \delta, q_0, Z_0, \mathbf{F}[, \mathbf{O}, \eta] \rangle$
- $\mathbf{Q}, \mathbf{I}, \delta, q_0, \mathbf{F}[, \mathbf{O}]$ come nell'FSA [traduttore]
- Γ alfabeto di pila (per comodità, disgiunto da $\mathbf{I}, [, \mathbf{O}]$)
- $Z_0 \in \Gamma$ simbolo iniziale di pila
- $\delta : \mathbf{Q} \times (\mathbf{I} \cup \{\varepsilon\}) \times \Gamma \to \mathbf{Q} \times \Gamma^*$ (n.b. δ è parziale)
- $\eta : \mathbf{Q} \times (\mathbf{I} \cup \{\varepsilon\}) \times \Gamma \to \mathbf{O}^*$ (η è definita solo dove lo è δ)

Convenzione grafica



$\delta(p, i, A) = \langle q, \alpha \rangle$
$[\eta(q, i, A) = w]$

Generalizzare lo stato

Il concetto di *configurazione*

- Catturare lo stato di un Automa a Pila (AP)[a] richiede più informazione di quella di un FSA
- Chiamiamo lo stato di un AP *configurazione* $c = \langle q, x, \gamma, [z] \rangle$
  - $q \in \mathbf{Q}$: stato dell'organo di controllo
  - $x \in \mathbf{I}^*$: stringa ancora da leggere (testina sul 1º carattere di x)
  - $\gamma \in \Gamma^*$ stringa dei caratteri in pila; convenzione: la pila cresce da sinistra (basso) a destra (alto)
  - $z \in \mathbf{O}^*$ stringa scritta in output

[a] Alternativamente, PDA, Push-Down Automaton

Formalizzare la transizione

Transizione tra configurazioni

- Transizione di un AP: $c \vdash c'$: $\langle q, x, \gamma, [z] \rangle \vdash \langle q', x', \gamma', [z'] \rangle$
- Per chiarezza abbiamo $\gamma = \beta A$, definiamo, a seconda dei casi:
  1. **Lettura effettiva**: con $x = i.y$ e $\delta(q, i, A) = \langle q', \alpha \rangle$ (definita, non $\perp$) $[\eta(q, i, A) = w]$ abbiamo $x' = y$, $\gamma' = \beta\alpha$, $[z' = z.w]$
  2. ε-**Lettura**: con $x = y$ e $\delta(q, \varepsilon, A) = \langle q', \alpha \rangle$ (definita, non $\perp$) $[\eta(q, \varepsilon, A) = w]$ abbiamo $x' = y$, $\gamma' = \beta\alpha$, $[z' = z.w]$
- Nota bene: $\forall q, A, \delta(q, \varepsilon, A) \neq \perp \Rightarrow \forall i, \delta(q, i, A) = \perp$
- Se ciò non accade, l'AP è *non-deterministico* (approfondimento del concetto più avanti)

Accettazione e traduzione

Sequenza di mosse

- Definiamo $\overset{*}{\vdash}$ come chiusura riflessiva, transitiva di $\vdash$

Accettazione e traduzione di $x \in L$

$$x \in L \wedge [z = \tau(x)]$$

$$\Leftrightarrow$$

$$c_0 = \langle q_o, x, Z_0, [\varepsilon] \rangle \overset{*}{\vdash} c_f = \langle q, \varepsilon, \gamma, [z] \rangle, q \in \mathbf{F}$$

N.b. attenzione alle ε mosse, soprattutto a fine stringa!

AP in pratica

Usi degli AP nel software

- Parte fondamentale degli analizzatori sintattici dei compilatori
- Macchina astratta a run-time dei linguaggi con ricorsione (Python, interprete Java)
- Può essere usato per controllare la correttezza di molti data-description languages (JSON,BSON,XML,HTML-4)
- È possibile generare le implementazioni a partire da specifiche sintetiche (corso di Formal Languages and Compilers)

Proprietà degli AP (riconoscitori)

Cosa posso riconoscere?

- Un AP è in grado di riconoscere $\{a^n b^n | n > 0\}$
- Posso riconoscere $\{a^n b^n c^n | n > 0\}$?:
  - NO. Intuitivamente: Dopo aver impilato un simbolo per ogni a e spilato uno per ogni b, come conto le c?
  - Per la dimostrazione formale si usa l'estensione del pumping lemma per i linguaggi riconosciuti dagli AP
  - Esiste un $p \geq 1$ tale per cui, data $x = pvcws \in L_{AP}, |x| \geq p$ con $|vcw| \leq p$, $|vc| \geq 1 \ \Leftrightarrow \forall n \in \mathbb{N}, pv^n cw^n s \in L_{AP}$
- La pila è una memoria distruttiva: per leggere (sotto la cima) occorre cancellare elementi!

Proprietà degli AP (riconoscitori) - 2

Cosa posso riconoscere?

- Un AP riconosce sia $\{a^n b^n | n > 0\}$ che $\{a^n b^{2n} | n > 0\}$
- Posso riconoscere $\{a^n b^n | n > 0\} \cup \{a^n b^{2n} | n > 0\}$
  - NO. Intuitivamente "simile" a prima:
  - Se svuoto la pila per contare le prime n b perdo memoria per le successive
  - Se ne svuoto solo metà, e ce ne sono solo n non so se sono a metà pila
  - Intuitivamente: mi servirebbe "dare un' occhiata" in avanti sull'input, per un numero arbitrariamente grande di caratteri
  - Formalizzazione diversa dal precedente (piuttosto complessa, non c'è sul libro)

Conseguenze delle proprietà

La famiglia $\mathbf{L}_{AP}$

- $\mathbf{L}_{AP}$: la famiglia di linguaggi riconosciuti dagli AP (determ.)
- $\mathbf{L}_{AP}$ non è chiusa rispetto all'unione per quanto detto
- $\mathbf{L}_{AP}$ è chiusa rispetto al complemento? Sì.
  - Il principio della dimostrazione è lo stesso degli FSA, scambiare $\mathbf{F}$ con $\mathbf{Q} \setminus \mathbf{F}$
- $\mathbf{L}_{AP}$ non è chiusa rispetto all'intersezione (perché?)

Costruire il complemento

Difficoltà nella costruzione

- La δ dell'automa va completata come per gli FSA con lo stato di errore
  - Le ε mosse possono introdurre non-determinismo
- Un ciclo di ε mosse può evitare che l'automa proceda (stringa non accettata, neppure dall'automa con $\mathbf{F}' = \mathbf{Q} \setminus \mathbf{F}$)
  - Si può trasformare ogni AP con cicli di ε mosse in uno equivalente privo di essi
- Se esiste una sequenza $\langle q_1, \varepsilon, \gamma_1 \rangle \vdash \langle q_2, \varepsilon, \gamma_2 \rangle \vdash \langle q_3, \varepsilon, \gamma_3 \rangle$ dove solo $q_1, q_3 \in \mathbf{F}$, ma $q_2 \notin \mathbf{F}$ cosa succede?
  - Serve "forzare" l'automa ad accettare solo alla fine di una sequenza (necessariamente finita) di ε mosse
- *Più della tecnica di dimostrazione è importante: per impiegare la macchina che risolve il "problema positivo" per risolvere quello "negativo" serve essere sicuri di "arrivare in fondo"*