Non Determinismo

Dipartimento di Elettronica, Informazione e Bioingegneria
Politecnico di Milano

13 marzo 2024

Modelli operazionali non deterministici

Modelli deterministici vs. modelli non deterministici

- Solitamente, un algoritmo è una *sequenza deterministica* di passi
  - Data una configurazionee un input, il passo successivo è unico
- È sempre utile? È sempre il modello più maneggevole?
- Non sempre l'ordine di esecuzione è fondamentale
  - Es. Trovare due calzini dello stesso colore e metterli
  - È importante che la computazione termini con due calzini dello stesso colore addosso, non "come ci sono arrivati"

Modelli operazionali non deterministici

La “comodità” del non determinismo

- Una descrizione di un algoritmo non deterministico è solitamente più compatta (sacrificando operatività)

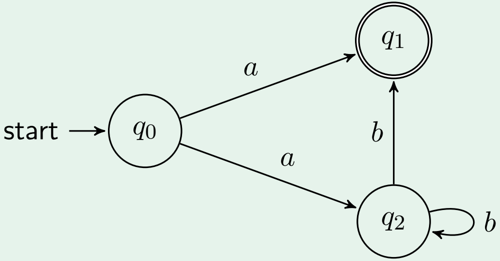
Non-det. “Trova una strada dall’entrata all’uscita”

Det. `https://en.wikipedia.org/wiki/Maze_solving_algorithm`

- Per una realizzazione pratica si possono sfruttare più esecutori
- Simmetricamente, un modello di esecuzione non-deterministico può essere utile come semantica per il calcolo parallelo (pthreads, Ada, OpenCL inter workgroups)

Versioni nondeterministiche (ND) di modelli noti

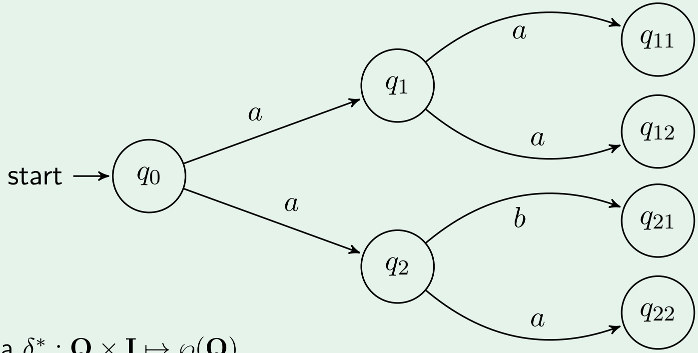
Automi a stati finiti ND (riconosce $L = ab^*$)



- La δ diventa $\delta : \mathbf{Q} \times \mathbf{I} \mapsto \wp(\mathbf{Q})$
- Nell'esempio abbiamo $\delta(q_0, a) = \{q_1, q_2\}$; $\delta(q_2, b) = \{q_1, q_2\}$

Automi a stati finiti ND

Formalizzazione della sequenza di mosse



- La δ diventa $\delta^* : \mathbf{Q} \times \mathbf{I} \mapsto \wp(\mathbf{Q})$
- Nell'esempio abbiamo $\delta(q_0, a) = \{q_1, q_2\}$; $\delta(q_2, b) = \{q_{21}\}$
- Di conseguenza, $\delta^*(q_0, aa) = \delta(q_1, a) \cup \delta(q_2, a) = \{q_{11}, q_{12}, q_{22}\}$

Automi a stati finiti ND

Formalizzazione della sequenza di mosse - 2

- Estensione di δ a stringhe:

$$\delta^*(q,x) = \begin{cases} \delta(q,\varepsilon) = \{q\}, \text{ con } x = \varepsilon \\ \delta(q,y.i) = \bigcup_{r \in \delta^*(q,y)} \delta(r,i), \text{ con } x = y.i, i \in \mathbf{I} \end{cases}$$

Accettazione di un FSA-ND

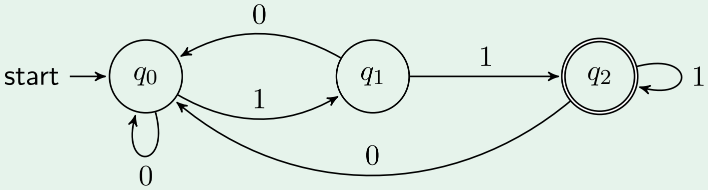
- Nostra convenzione: l'FSA-ND accetta x se almeno uno degli stati dell'insieme $\delta^*(q_0,x)$ è finale, ovvero

$$x \in L \Leftrightarrow (\delta^*(q_0,x) \cap \mathbf{F}) \neq \varnothing$$

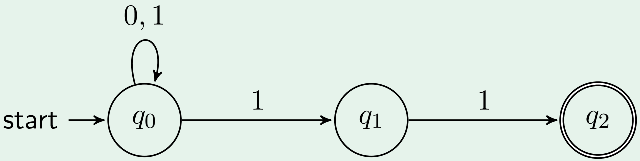
- È possibile anche considerare $\delta^*(q_0,x) \subseteq \mathbf{F}$

Automi a stati finiti ND e D a confronto

FSA-D per $L = \{0,1\}^*11$ (stringhe che terminano in 11)

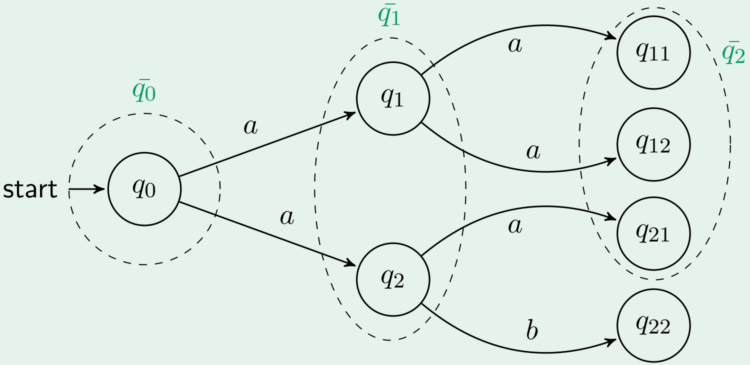


FSA-ND per $L = \{0,1\}^*11$ (stringhe che terminano in 11)



Automi a stati finiti ND e D a confronto

FSA ND ha maggior potere riconoscitivo di FSA? No...



- Intuitivamente: Otteniamo un FSA-D equivalente chiamando “stato” ogni insieme d’arrivo della δ e mantenendo gli archi

Automi a stati finiti ND e D a confronto

Sistematizzando il confronto

- Posso sempre costruire *automaticamente* un FSA-D equivalente a un ND dato:
- Da $\mathcal{A}_{nd} = \langle \mathbf{Q}_n, \mathbf{I}_n, \delta_n, q_{0n}, \mathbf{F}_n \rangle$ ricavo $\mathcal{A}_d = \langle \mathbf{Q}_d, \mathbf{I}_d, \delta_d, q_{0d}, \mathbf{F}_d \rangle$
  - $\mathbf{Q}_d = \wp(\mathbf{Q}_n)$, $\mathbf{I}_d = \mathbf{I}_n$
  - $\delta_d(q_d, i) = \bigcup_{q_n \in q_d} \delta_n(q_n, i)$
  - $q_{d0} = \{q_{n0}\}$
  - $\mathbf{F}_d = \{s \in \wp(\mathbf{Q}_n) | s \cap \mathbf{F}_n \neq \varnothing\}$
- Gli FSA-ND *non sono* più potenti degli FSA-D

Usi degli FSA-ND

Comodità di specifica

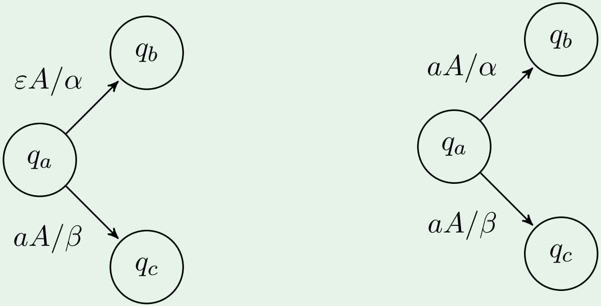
- Se FSA-D e FSA-ND sono equivalenti come potere riconoscitore, perchè mantenere gli ND (più "problematici")?
- Può essere più comodo specificare un FSA ND e poi far ricavare automaticamente il corrispondente D ad un programma, risparmiando la fatica di concepire quello D
- Attenzione: la determinizzazione di un FSA ha un costo: alla peggio $|\mathbf{Q}_d| = |\wp(\mathbf{Q}_n)| = 2^{|\mathbf{Q}_n|}$
  - É il caso pessimo, non tutti gli stati sono necessariamente raggiungibili, ma può succedere
  - Esempio: l'FSA riconoscitore di stringhe con un dato suffisso

Automi a pila non deterministici

Come ottenerli?

- "Naturalmente" non deterministici, basta eliminare la restrizione imposta fino ad ora

Transizioni ammesse per AP-ND



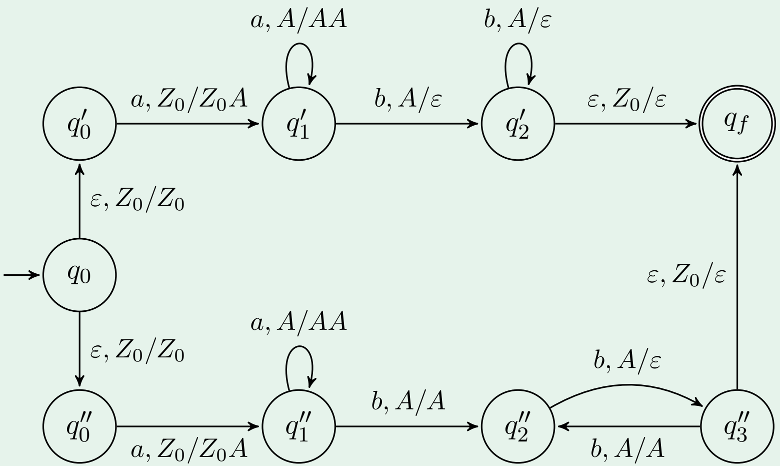
Automi a pila non deterministici

Effetti della generalizzazione

- Otteniamo la $\delta_{\text{AP-ND}} : \mathbf{Q} \times (\mathbf{I} \cup \{\varepsilon\}) \times \Gamma \to \wp_F(\mathbf{Q} \times \Gamma^*)$
  - Come mai il pedice F? I possibili sottoinsiemi di $\mathbf{Q} \times \Gamma^*$ sono infiniti, consideriamo solo quelli ottenibili dalle coppie nell'immagine di $\delta_{\text{AP-ND}}$, ottenendo un insieme delle parti *finito*
- L'AP ND accetta x se esiste una sequenza $c_o \overset{*}{\vdash} \{c_1, \ldots c_n\}$ con $c_0 = \langle q_0, x, Z_0 \rangle, c_1 = \langle q, \varepsilon, \gamma \rangle, q \in \mathbf{F}$
- N.B. $\vdash$ non è più univoca!

Automi a pila non deterministici

Un esempio di riconoscitore



Conseguenze del riconoscitore d'esempio

Proprietà degli AP ND

- L'automa precedente è in grado di riconoscere $\{a^n b^n\} \cup \{a^n b^{2n}\}$
  - Gli AP ND sono più potenti degli AP D
- Generalizzando l'AP precedente si ottiene una dimostrazione *costruttiva* della chiusura di AP ND rispetto all'unione
  - Proprietà *non* condivisa dagli AP D
- Gli AP ND non sono chiusi rispetto all'intersezione
  - $\{a^n b^n c^*\} \cap \{a^* b^n c^n\} = \{a^n b^n c^n\}$ non è riconoscibile neppure in modo ND da un AP (il pumping lemma per AP vale anche per quelli ND)

Conseguenze delle proprietà

Chiusura rispetto al complemento

- Se la famiglia di linguaggi $\mathbb{L}_{\text{APND}}$ è chiusa rispetto a $\cup$ e non rispetto a $\cap$ non può esserlo rispetto al complemento
- Il nondeterminismo impatta direttamente sulla complementazione:
  - Se un automa è deterministico e termina sempre la sua computazione è sufficiente scambiare accettazione e non accettazione per avere il complemento
- Con gli APND la computazione termina sempre ma se ho:
  - $\langle q_0, x, Z_0 \rangle = c_0 \overset{*}{\vdash} \{\langle q_1, \varepsilon, \gamma_1 \rangle, \langle q_2, \varepsilon, \gamma_2 \rangle\},\ q_1 \in \mathbf{F}, q_2 \notin \mathbf{F}$
  - x è accettata nella computazione precedente, ma continua ad esserlo anche se scambio $\mathbf{F}$ con $\mathbf{Q} \setminus \mathbf{F}$

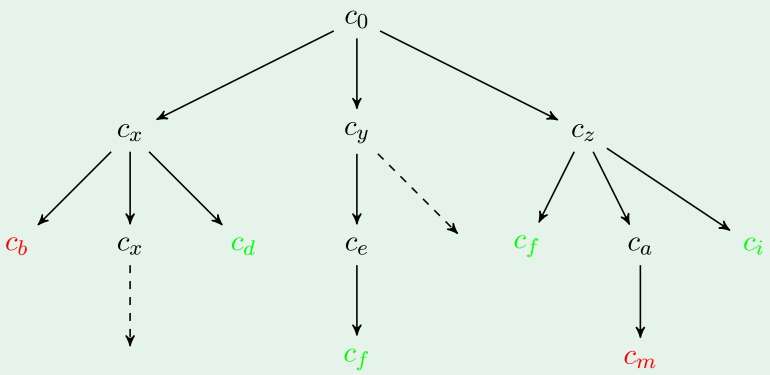
Macchine di Turing nondeterministiche

Definizione e caratteristiche

- $\delta : \mathbf{Q} \times \mathbf{I} \times \Gamma^k \to \wp(\mathbf{Q} \times \Gamma^k \times \{\mathtt{L}, \mathtt{S}, \mathtt{R}\})$ (perchè $\wp$ e non $\wp_F$?)
- Configurazione, transizione, sequenza di transizioni e accettazione definite come negli altri casi
- Prima domanda: le MT ND sono più potenti delle MT D?

Albero delle computazioni

Rappresentare una computazione ND



Verde: Computazione accettante, Rosso: Computazione non accettante

Equivalenza tra MT ND e MT D

Emulare una MT ND con una MT D

- x è accettata da una MT ND solo se esiste un calcolo che termina in uno stato di accettazione
- Come emulare una MT ND con una D?
  - Percorrere l'albero delle computazioni ND per stabilire se esiste un percorso che termina in uno stato di accettazione
  - Nel caso di un albero "normale", esistono algoritmi consolidati per effettuare questa visita
  - Come gestisco le computazioni che non terminano?
- Visita dell'albero "in ampiezza"
  - Costruisco una MT D che scandisce le configurazioni della ND a partire dalle più vicine a c_0
  - Intuitivamente: se la MT ND termina, termina anche la mia MT D con lo stesso esito

Conclusioni sul nondeterminismo

Un utile formalismo

- Utile per rappresentare problemi/algoritmi dove alcune scelte locali non sono fattibili al momento/importanti
- Aumenta la potenza dei soli AP (tra i formalismi visti)
- Può essere applicato praticamente a tutti i modelli di calcolo (estensione facile ai traduttori)
- N.B.: nondeterministico $\neq$ probabilistico
  - La computazione procede sempre con *certezza* verso l'insieme di stati successivo
  - Esistono modelli di calcolo probabilistico, ma sono ben diversi (e.g., FSA-prob $\approx$ catene di Markov)