Lezione 4 – macchine non deterministiche

Lezione del 16/03/2023

A proposito dell'insieme delle quintuple

- Siamo al paragrafo 2.3 della dispensa 2 (pag. 4).
- Prendiamo una macchina di Turing:
  - cioè, un alfabeto Σ e un insieme degli stati Q
  - e, soprattutto, *l'insieme delle sue quintuple* P
  - osservate che è sufficiente avere l'insieme P per sapere tutto di T: da P possiamo ricavare sia Σ che Q
  - beh, in effetti P non ci dice proprio tutto tutto: per sapere tutto di T, oltre che P, dobbiamo conoscere anche quale sia lo stato iniziale e quali siano gli stati finali
  - e questa cosa, quello che ci occorre per sapere tutto di T, tenetelo a mente perché ci servirà nella prossima lezione
- Bene. Quindi, P è il "cuore" di T. Ora, andiamo a studiare la struttura di P
- Intanto, ricordiamo che possiamo vedere P come una funzione che associa ad una coppia (stato, simbolo) una tripla (stato, simbolo ,movimento) , ossia,
- $$P: Q \times \Sigma \rightarrow \Sigma \times Q \times \{S, F, D\}$$

P è una funzione totale?

- Una quintupla $\langle q_1 , a, b, q_2 , m\rangle$ ci dice che: se siamo nello stato q_1 e leggiamo il carattere a allora dobbiamo comportarci in un certo modo – e sappiamo in quale modo. Facile.
- Ma cosa succede se, trovandoci in uno stato q e leggendo un carattere s non troviamo in P alcuna quintupla i cui primi due simboli sono q e s?
  - non viene indicata alcuna azione da compiere!
- Non viene indicata alcuna azione da compiere. E, allora, T non può far altro che non compiere alcuna azione.
- Cioè, T interrompe la sua computazione - è come se avesse raggiunto uno stato finale
- però, uno stato finale non lo ha raggiunto
  - e questo fatto qualche conseguenza ce l'avrà pure
  - altrimenti, a cosa servono gli stati finali?
- Per capire, dobbiamo chiarire che cosa significa che ad una coppia (q,s) non è associata alcuna quintupla in P – e lo facciamo separatamente per i trasduttori e per i riconoscitori

P è una funzione totale? (Trasduttori)

- Se T è un trasduttore, che cosa significa che ad una coppia (q,s) non è associata alcuna quintupla in P?
- Facciamo un esempio: consideriamo una macchina T a 3 nastri che calcola il risultato dell'addizione in colonna di due interi **nel caso in cui il secondo addendo è costituito di sole cifre pari**
  - le quintuple di T sono molto simili a quelle della macchina che esegue la somma in colonna di due interi qualsiasi (che abbiamo analizzato abbondantemente)
  - l'unica differenza è le quintuple di T si aspettano di trovare solo cifre pari sul secondo nastro
  - ossia, contiene solo quintuple del tipo 〈 q , (x,y, □), (x,y,z), q_1 , sinistra〉 , dove x è una cifra qualsiasi e y è una cifra pari (e z è la cifra che si ottiene da x+y)
- *Se assumiamo che il secondo addendo (scritto sul secondo nastro) sia costituito di sole cifre pari,* allora
  - eseguendo tutte le quintuple che abbiamo visto nelle lezioni precedenti
  - T scrive, una alla volta, le cifre del risultato sul nastro di output
  - ossia, man mano che la computazione prosegue, le cifre del risultato compaiono sul nastro di output

P è una funzione totale? (Trasduttori)

- T è una macchina a 3 nastri che calcola il risultato dell'addizione in colonna di due interi **nel caso in cui il secondo addendo è costituito di sole cifre pari**
  - non esistono quintuple di T del tipo ⟨ q , (x,y, □), (x,y,z), q_1 , sinistra⟩ , dove y è una cifra dispari
- *Se assumiamo che il secondo addendo (scritto sul secondo nastro) sia costituito di sole cifre pari*, allora
  - eseguendo tutte le quintuple che abbiamo visto nelle lezioni precedenti, T scrive, una alla volta, le cifre del risultato sul nastro di output
  - ossia, man mano che la computazione prosegue, le cifre del risultato compaiiono sul nastro di output
- Cosa succede, però, se un utente distratto ha scritto 1234 sul primo nastro e 2560 sul secondo nastro?
  - T scrive 4 sul nastro di output
  - poi, scrive 9 sul nastro di output
  - poi... OPS! Non trova più alcuna quintupla da eseguire
  - ma il nastro di output non è vuoto...

P è una funzione totale? (Trasduttori)

- Se, trovandoci in uno stato q e leggendo un carattere s non troviamo in P alcuna quintupla i cui primi due simboli sono q e s, poiché non viene indicata alcuna azione da compiere, T non può far altro che non compiere alcuna azione!
- Cioè, T interrompe la sua computazione - è come se avesse raggiunto uno stato finale
- Quindi potremmo pensare che: ogni qualvolta ad una coppia (q,s) non è associata alcuna quintupla in P, è possibile aggiungere a P la quintupla $\langle q , s, s, q_F , F \rangle$
- Tuttavia...
  - mentre un trasduttore lavora, man mano che esegue le sue quintuple, è possibile che scriva qualcosa sul nastro di output – la prima parte del risultato
  - però se la computazione di T è terminata non perché T è entrata in uno stato finale ma perché non ha trovato quintuple da eseguire allora *quel che è scritto sul nastro di output non è il risultato cercato*!
  - E l'utente come fa a capirlo????

P è una funzione totale? (Trasduttori)

- Come fa l'utente come fa a capire se quel che è scritto sul nastro di output è il risultato cercato oppure no?
- Abbiamo due possibilità a disposizione:
  - chi ha progettato T, con la santa pazienza, ha considerato **tutte le possibilità (stato,simbolo), anche quelle "impossibili"** (che si incontrano quando l'utilizzatore non legge il libretto di istruzioni di T e scrive sul nastro un input non conforme alle specifiche): per ciascuna di queste coppie impossibili ha scritto una serie di quintuple che prima cancellano il contenuto del nastro di output e poi portano T in q_F
    - o, equivalentemente, per ciascuna di queste coppie, viene scritto un messaggio di errore sul nastro di output prima di raggiungere lo stato q_F
  - chi ha progettato T ha deciso che ***se un utilizzatore è stato poco accorto e non ha rispettato le specifiche… peggio per lui!*** E, semplicemente, chi ha progettato T ha scritto solo le quintuple per le coppie (stato,simbolo) significative. E, così, ha progettato una funzione P non totale

P è una funzione totale? (Riconoscitori)

- Per capire cosa accade quando T è un riconoscitore dobbiamo prima chiarire…
- Cosa significa, nel caso in cui T è un riconoscitore, che ad una coppia (q,s) non è associata alcuna quintupla in P?
- Facciamo un esempio: consideriamo una macchina T che decide se il risultato dell'addizione di due interi sarà un numero pari
  - non deve calcolare il risultato: deve solo terminare in q_A qualora il risultato sia pari, in q_R qualora il risultato sia dispari
- Quindi, *se assumiamo che l'input sia scritto sul nastro di T nella forma* "primo addendo + secondo addendo" (dove ciascun addendo è una sequenza di cifre), allora
  - T deve spostare la testina sulla cifra meno significativa del primo addendo (quella a sinistra del '+') e ricordarsi se è pari o dispari
  - poi deve spostare la testina sulla cifra meno significativa del secondo addendo (ossia seve superare la sequenza di ***cifre*** che compongono il secondo addendo e posizionarsi sulla cifra a sinistra del '□') e, dipendentemente da quello che si ricordava e dal fatto che tale cifra sia pari o dispari, terminare in q_A o in q_R.

P è una funzione totale? (Riconoscitori)

- Dunque, **se assumiamo che l'input sia scritto sul nastro di T nella forma "primo addendo + secondo addendo",** allora la nostra macchina T, buona buona, tric trac tric trac, esegue la sua computazione e ci dà la risposta corretta – memorizzata nel suo stato. Fantastico.

- Ma che succede se, invece, un utilizzatore poco accorto scrive sul nastro di T la parola "576+48+"? Come si comporta T?
  - non sappiamo come si comporta, ma certamente ci aspettiamo che essa non termini in q_A
  - perché T deve terminare in q_A solo se la somma di <u>due</u> numeri è pari
  - e qui, invece, le viene proposta la somma di *quasi* <u>*tre*</u> *numeri* (ossia, due numeri e un +)
  - dall'utente che non rispetta *le specifiche di T* (e questa cosa è discussa bene nel paragrafo 2.3)!)

P è una funzione totale? (Riconoscitori)

- Ma che succede se, invece, un utilizzatore poco accorto scrive sul nastro di T la parola "576+48+"? Come si comporta T?
  - ci aspettiamo che la computazione di T termini in q_R
- Allora, ci sono due possibilità:
  - chi ha progettato T, con la santa pazienza, ha considerato **tutte le possibilità (stato,simbolo), anche quelle "impossibili"** (quando l'utilizzatore non legge il libretto di istruzioni di T e scrive sul nastro un input non conforme alle specifiche): per ciascuna di queste coppie impossibili ha scritto una quintupla che porta T in q_R .
  - chi ha progettato T ha deciso che ***se un utilizzatore è stato poco accorto e non ha rispettato le specifiche… peggio per lui!*** E, semplicemente, chi ha progettato T ha scritto solo le quintuple per le coppie (stato,simbolo) significative. E, così, ha progettato una funzione P non totale
- Possiamo ora rispondere alla domanda "ma se T è un riconoscitore e ad una coppia (q,s) non è associata alcuna quintupla in P?: in questo caso, possiamo assumere che, in tal caso, T rigetti
  - è come se, *implicitamente*, aggiungessimo a P la quintupla $\langle q , s, s, q_R , F \rangle$

P è una funzione totale? (Riconoscitori)

- Possiamo ora rispondere alla domanda "ma se T è un riconoscitore e ad una coppia (q,s) non è associata alcuna quintupla in P?: in questo caso, possiamo assumere che, in tal caso, T rigetti
  - è come se, *implicitamente*, aggiungessimo a P la quintupla $\langle q , s, s, q_R , F\rangle$
- Attenzione però: tutto ciò ha una importante conseguenza
- Torniamo a considerare la macchina che decide se la somma di due numeri dati in input è un numero pari completata con le quintuple che portano la macchina in q_R se l'input è scritto in un formato errato
  - scriviamo il nostro input sul nastro e facciamo partire la computazione
  - se la macchina termina in q_A allora possiamo essere certi: la somma dei due numeri è proprio un numero pari
  - ma se la macchina termina in q_R allora non possiamo concludere che la somma dei due numeri è un numero dispari
    - infatti la macchina potrebbe aver terminato in q_R perché il formato dell'input era errato
- Perciò, se la macchina termina in q_R possiamo concludere soltanto che la somma dei due numeri **non è un numero pari**!

E se P non fosse una funzione?

- E che vuol dire "e se P non fosse una funzione?"?!
- Ma, prima ancora, cosa vuol dire che P è una *funzione*?
- Beh, questo è facile: se P è una funzione, allora, per ogni stato q e per ogni carattere a, non possono esistere due quintuple che iniziano con la coppia (q,a)
- In effetti, una quintupla $\langle q_1 , a, b, q_2 , m \rangle$, per come la abbiamo definita, ci dice che: se siamo nello stato q_1 e leggiamo il carattere a allora dobbiamo comportarci in un certo modo – e non abbiamo scelta: trovandoci nello stato q_1 e leggendo il carattere a non possiamo far altro che scrivere b, entrare nello stato q_2 e muovere come specificato da m la testina.
- Quindi, una quintupla è un ordine – senza se e senza ma, se vogliamo giungere alla soluzione (dell'istanza) del problema, dobbiamo obbedire!
- E, quindi, da quello che abbiamo detto fino ad ora, non avrebbe senso avere due quintuple $\langle q_1 , a, b, q_2 , m \rangle$ e $\langle q_1 , a, b', q'_2 , m' \rangle$: come dovremmo mai comportarci trovandoci nello stato q_1 e leggendo il carattere a?!

E se P non fosse una funzione?

- Possiamo anche vedere una quintupla come una indicazione precisa e non ambigua circa quale operazione eseguire per giungere alla soluzione
  - siamo certi che, se agiamo come specificato nella quintupla, arriviamo alla soluzione.
- Una indicazione che viene fornita da chi ha progettato la macchina di Turing
  - e che è una conseguenza della sua analisi del problema che lo ha condotto ad individuare un certo procedimento di soluzione
- E se costui, il progettista, arrivato ad un certo punto non sapesse bene che pesci pigliare? O se si scocciasse di fare il precisino per indicarci le istruzioni per filo e per segno?!
- Potrebbe, che so, dirci "se sei nello stato q e leggi il simbolo a, non so bene quale è la cosa giusta da fare ma, di certo, devi fare una di queste cose: [elenco di cose da fare fra cui scegliere]. Decidi un po' tu…"
- E come farebbe costui a comunicarci questa cosa? Ma con tante quintuple che iniziano con la stessa coppia stato interno – simbolo letto!

E se P non fosse una funzione?

- Tante quintuple che iniziano con la stessa coppia (stato interno – simbolo letto):
  - $\langle q, a, b_1, q_1, m_1 \rangle$, $\langle q, a, b_2, q_2, m_2 \rangle$, … $\langle q, a, b_k, q_k, m_k \rangle$,
  - chiamiamo, fra noi, questa struttura (tante quintuple che iniziano con la stessa coppia) una multi-quintupla
- Cosa accade quando l'insieme delle quintuple di una macchina T ha la multi-quintupla sopra descritta e, durante una computazione T(x), si trova nello stato interno q e legge il carattere a?
- Possiamo descrivere il comportamento di T in due modi diversi:
  - T diventa una macchina super-iper-ultra parallela
  - T chiede l'intervento di un genio della lampada (burlone e pasticcione)
  - i due modi diversi sono **equivalenti**
- Andiamo con ordine...

Una macchina super-iper-ultra parallela

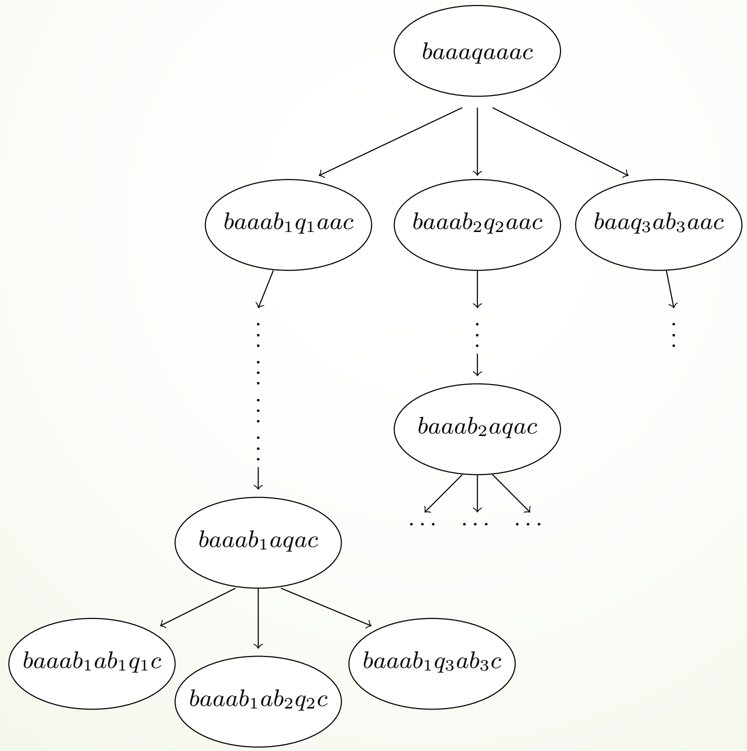
- In questo caso, quando T si trova nello stato q e legge il simbolo a, le k quintuple $\langle q, a, b_1, q_1, m_1 \rangle$, $\langle q, a, b_2, q_2, m_2 \rangle$, … $\langle q, a, b_k, q_k, m_k \rangle$, T le esegue… tutte! In parallelo!
- Succede una specie di magia e... ta-dah! Si moltiplicano i nastri, e si moltiplica l'unità di controllo
- così che avviene la transizione dallo stato globale di partenza a k stati globali differenti
  - che proseguono la computazione, ognuno per conto suo
- e se, successivamente, da uno di questi stati globali ci si troverà a dover eseguire un'altra multi-quintupla, il processo di moltiplicazione si ripeterà
  - diventerà una specie di albero
- vediamolo

Una macchina super-iper-ultra parallela

P contiene le tre quintuple seguenti che iniziano con (q, a):

$$\langle q, a, b_1, q_1, D\rangle, \langle q, a, b_2, q_2, D\rangle, \langle q, a, b_3, q_3, S\rangle$$

La macchina si trova nello stato globale $SG = baaaqaaac$



Ciascun “**ramo**” dell’albero è una ***computazione deterministica*** della macchina

Una macchina super-iper-ultra parallela

- Già, ma, allora, quale è l'esito di una computazione di una macchina capace di auto-replicarsi in innumerevoli copie parallele?
- Come facciamo a dire quando una computazione di siffatta macchina accetta e quando rigetta? Come facciamo a sapere se la soluzione all'istanza x del nostro problema è 0 oppure 1? A quale delle copie parallele dobbiamo dar credito?
- Per rispondere, dobbiamo prima chiarire una questione: anche se stiamo parlando di funzioni a valori in {0,1}, c'è, in realtà, una certa asimmetria fra i due valori. O meglio, c'è una asimmetria fra gli stati q_A e q_R
  - ad un riconoscitore è richiesto di riconoscere le parole che **soddisfano** una certa proprietà - ad esempio, deve riconoscere le parole palindrome
  - non di riconoscere le parole che ***non*** soddisfano quella proprietà – per questo insieme di parole, se ci interessa individuarle, potremmo progettare un altro riconoscitore!
- Quindi, in effetti, quel che ci interessa "di più" è lo stato q_A – possiamo dire che arriviamo alla soluzione quando la macchina raggiunge lo stato q_A
  - come se fossimo in un groviglio di strade e dovessimo trovare il percorso che ci porta a destinazione: dei percorsi che non arrivano a destinazione, che ci importa?

Una macchina super-iper-ultra parallela

- E allora, come facciamo a dire quando una computazione di siffatta macchina accetta e quando rigetta? Come facciamo a sapere se la soluzione all'istanza x del nostro problema è 0 oppure 1? A quale delle copie parallele dobbiamo dar credito?
- Ragioniamo: l'idea delle multi-quintuple è che ci vengono mostrate tante strade possibili che *potrebbero* "portarci a destinazione" – ossia, allo stato q_A
- Naturalmente, non tutte le strade portano alla soluzione.
- ***Ma basta che ce ne sia una, di strada che porta a destinazione!***
- Quindi, diciamo che: la computazione di una macchina super-iper-ultra parallela
  - **accetta se <u>esiste</u> almeno un percorso nell'albero che porta la macchina nello stato q_A**
  - **rigetta se <u>tutti</u> i percorsi nell'albero portano nello stato q_R– ossia se il percorso che porta a destinazione proprio non esiste!**

Arriva il genio (burlone e pasticcione)

- In questo caso, quando T si trova nello stato q e legge il simbolo a, e P contiene le k quintuple $\langle q, a, b_1, q_1, m_1 \rangle$, $\langle q, a, b_2, q_2, m_2 \rangle$, … $\langle q, a, b_k, q_k, m_k \rangle$, T chiama un **genio** e quello **sceglie** quale di queste quintuple eseguire!
- Così, la computazione diventa una sequenza di **scelte** del genio
  - e, neanche a dirlo, geni diversi possono fare scelte diverse
- Per questo la computazione di T prende il nome di ***non deterministica***
  - **perché il suo esito non è completamente determinato dal suo input**
- Cioè: **se una macchina di Turing non ha multi-quintuple, allora, per ogni input x, la computazione T(x) avrà sempre lo stesso esito**
  - se eseguiamo T(x) ed essa termina in q_A, e poi ripetiamo T(x) un milione di volte, ogni ripetizione terminerà in q_A
  - e lo stesso vale se la prima esecuzione di T(x) termina in q_R
  - per questo una macchina che non ha multi-quintuple è ***deterministica***
- **Se, invece, T contiene multi-quintuple, se T è *non deterministica*, allora esecuzioni diverse di T(x) possono avere esiti diversi!**

Arriva il genio (burlone e pasticcione)

- Già, ma, allora, quale è l'esito di una computazione di una macchina non deterministica T nel modello in cui interviene il genio?
  - Anzi, visto che la macchina è non deterministica, chiamiamola NT
- Come facciamo a dire quando NT(x) accetta e quando rigetta? Come facciamo a sapere se la soluzione all'istanza x del nostro problema è 0 oppure 1?
- La risposta è analoga al modello super-iper-ultra parallelo e, come in quel caso, è asimmetrica:
  - **NT(x) accetta se <u>esiste almeno una scelta di multi-quintuple</u> (o, se preferite, almeno una scelta di almeno un genio) che porta la macchina nello stato q_A**
  - **NT(x) rigetta se <u>qualunque scelta di multi-quintuple</u> (o, se preferite, qualunque scelta di qualunque genio) porta la macchina nello stato q_R**

Equivalenza fra i due modelli

- In definitiva
  - una macchina super-iper-ultra parallela accetta se *esiste un percorso* nell'albero che la fa entrare nello stato q_A e rigetta se *tutti i percorsi* la fanno entrare nello stato q_R
  - una macchina genio-dotata accetta se *esiste una scelta di quintuple* che la fa entrare nello stato q_A e rigetta se *tutte le scelte di quintuple* la fanno entrare nello stato q_R
- I due modelli sono equivalenti!
- Sono due modelli, due modi, in cui possiamo descrivere una ***macchina di Turing non deterministica***

Determinismo e non determinismo

- Ricapitolando:
  - una macchina di Turing T è **deterministica** se, per ogni stato q e per ogni carattere a, l'insieme P delle sue quintuple non contiene più di una quintupla che inizia con (q,a)
  - una macchina di Turing NT è **non deterministica** se esistono uno stato q e un carattere a tali che l'insieme P delle sue quintuple contiene due o più quintuple che iniziano con (q,a)
  - **consideriamo solo macchine non deterministiche di tipo riconoscitore**
- Una computazione non deterministica contiene tante computazioni deterministiche – una per ciascun ramo dell'albero
- Il ***grado di non determinismo*** di una macchina non deterministica NT è il massimo numero di quintuple che iniziano con la stessa coppia stato-carattere, ossia,

$$\max_{q,a} |\{ \langle q, a, b, q_1, m \rangle \in P \}|$$

- Naturalmente, il grado di non determinismo di una macchina definita sull'alfabeto Σ e sull'insieme degli stati Q può essere al massimo

$$|\Sigma| \times |Q| \times 3$$

  - ed è, quindi, (indovinate un po'?) COSTANTE!!!!
- Osservazione: **una macchina deterministica è una particolare macchina non deterministica con grado di non determinismo uguale a 1**

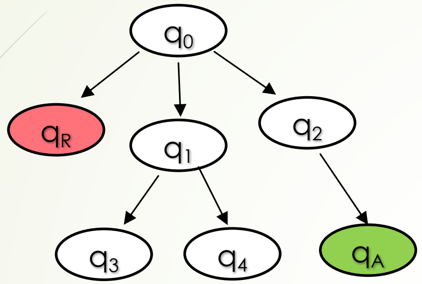
Determinismo e non determinismo

- Certo, che questa idea del non determinismo – della macchina che si auto-replica o che dispone di un genio – pare potente assai
- Chissà quante belle cose possiamo fare con una macchina non deterministica che non potremmo fare con una macchina deterministica…
- E invece no! Se sappiamo risolvere un problema con una macchina non deterministica allora sappiamo risolverlo anche con una macchina deterministica! (Teorema 2.1)
- Infatti possiamo sempre costruire una macchina deterministica T che **simula** una macchina non deterministica NT

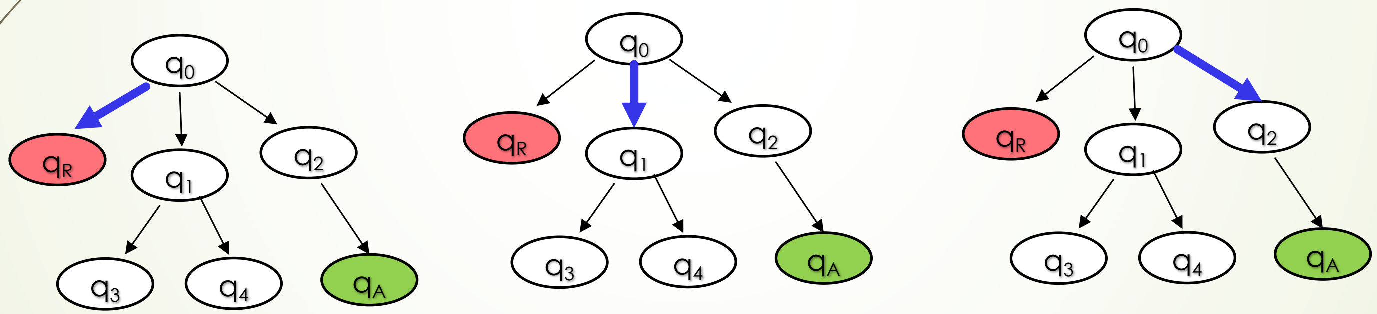
Determinismo e non determinismo

- Infatti possiamo sempre costruire una macchina deterministica T che simula una macchina non deterministica NT: con input x
  - simula tutte le computazioni deterministiche di NT(x) di un solo passo: se qualcuna accetta allora T accetta, se tutte rigettano allora T rigetta, altrimenti
  - simula tutte le computazioni deterministiche di NT(x) di due passi: se qualcuna accetta allora T accetta, se tutte rigettano allora T rigetta, altrimenti
  - simula tutte le computazioni deterministiche di NT(x) di tre passi, ecc. ecc. ecc.
- Detto ciò, andate a studiarvi il Teorema 2.1 (dispensa 2, pag. 5-6)
- Noi, intanto, vediamo con qualche esempio come funziona questa tecnica di simulazione
- che prende il nome di ***coda di rondine con ripetizioni***

Simulare una computazione accettante

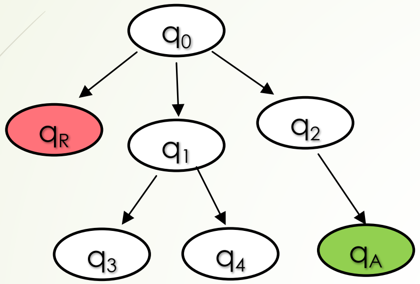


Computazione non deterministica
che accetta in due *passi*

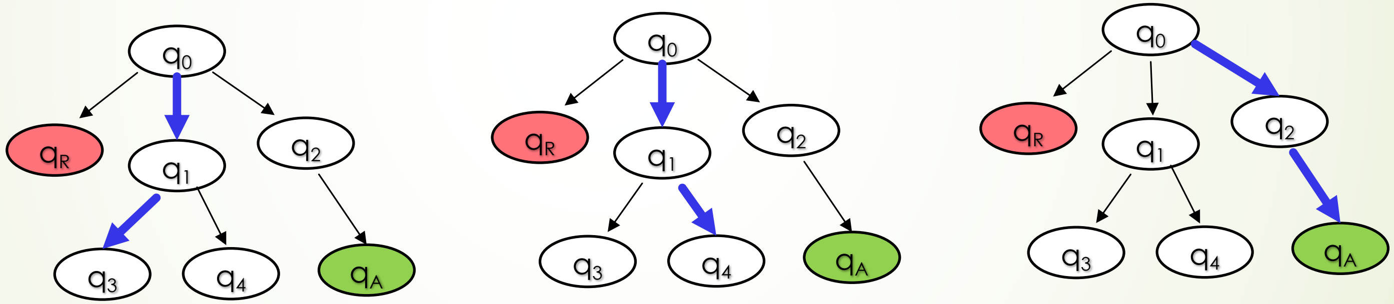


Simulazione delle tre computazioni deterministiche lunghe 1:
nulla si può concludere circa l'accettazione o il rigetto

Simulare una computazione accettante



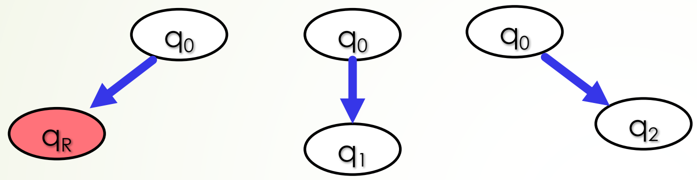
Computazione non deterministica
che accetta in due *passi*



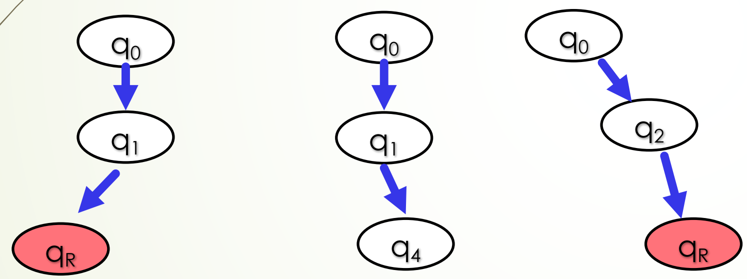
Simulazione delle tre computazioni deterministiche lunghe 2:
la computazione che entra in q_R non viene ripetuta, e si può concludere circa l'accettazione solo al termine dell'ultima simulazione.
Il passaggio attraverso gli stati globali contraddistinti da q_1 e q_2 si *ripete* più volte (coda di rondine *con ripetizioni*)

Simulare una computazione che rigetta

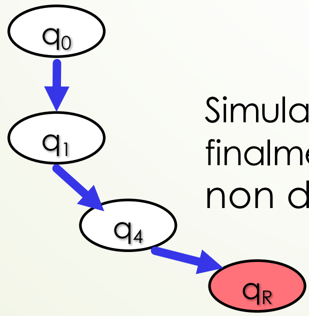
- Questa volta, però, lavoriamo "al buio" – ossia, senza conoscere a priori la computazione non deterministica



Simulazione delle computazioni di lunghezza 1: nulla possiamo concludere



Simulazione delle computazioni di lunghezza 2: anche ora, nulla possiamo concludere



Simulazione delle computazioni di lunghezza 3: finalmente, possiamo concludere che la macchina non deterministica rigetta il suo input

Determinismo e non determinismo

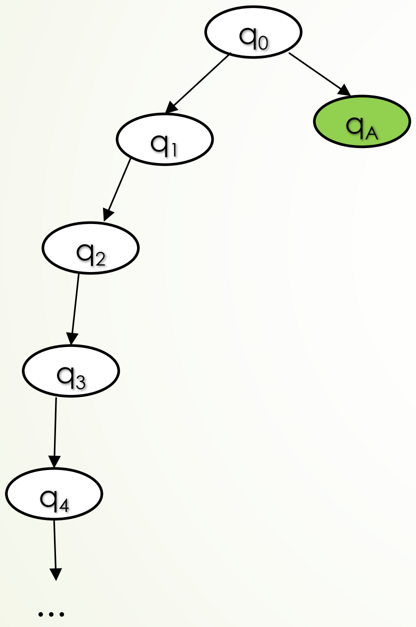
- Possiamo sempre costruire una macchina deterministica T che simula una macchina non deterministica NT: con input x
  - simula tutte le computazioni deterministiche di NT(x) di un solo passo: se qualcuna accetta allora T accetta, se tutte rigettano allora T rigetta, altrimenti
  - simula tutte le computazioni deterministiche di NT(x) di due passi: se qualcuna accetta allora T accetta, se tutte rigettano allora T rigetta, altrimenti
  - simula tutte le computazioni deterministiche di NT(x) di tre passi, ecc. ecc. ecc.

- DOMANDINA: ma perché non possiamo far simulare a T prima l'intero ramo più a sinistra dell'albero, poi quello accanto, e così via?
  - la risposta alla prossima pagina…

Computazioni che non terminano

- Perché alcune computazioni possono non terminare
  - ad esempio, se P contiene le due quintuple $\langle q_1, a, a, q_2, D\rangle$ e $\langle q_2, b, b, q_1, S\rangle$, e la macchina si trova nello stato globale **zzzz q_1 abzzzz** (zzzz sono caratteri qualsiasi)
  - allora la computazione non termina (va in loop!)
- Allora, potrebbe accadere che la computazione deterministica più a sinistra di NT(x) non termini, mentre quella più a destra termini in q_A
  - se simulassimo per prima la computazione non deterministica più a sinistra non termineremmo mai
  - e non riusciremmo mai a raggiungere lo stato q_A
  - quindi, mentre NT(x) accetta, la nostra macchina deterministica non terminerebbe mai (ARGH!)
  - Invece, simulando ogni volta computazioni di lunghezza fissata, prima o poi arriveremo a beccare lo stato di accettazione nella computazione più a destra!
  - Capito il punto? (in figura alla prossima pagina)
  - Studiate il Teorema!

Computazioni che non terminano



Quella illustrata in figura è una computazione non deterministica che accetta.

La computazione deterministica di sinistra non termina:
se noi provassimo a simulare l'intera computazione di sinistra tale simulazione non avrebbe termine

perciò non riusciremmo mai ad iniziare la simulazione della computazione di destra e non raggiungeremmo mai lo stato di accettazione

Ecco perché si utilizza la tecnica della coda di rondine!

Tanti modelli

- In conclusione, abbiamo visto tanti modelli di Macchine di Turing
  - dipendentemente dal numero di nastri di cui le dotiamo
  - e da come si muovono le testine
  - e da quanto è ricco l'alfabeto del quale dispongono
  - e dalla struttura dell'insieme delle quintuple (macchine deterministiche / non deterministiche)
- E abbiamo dimostrato che tutti questi modelli sono fra loro equivalenti
  - quello che sappiamo fare con una macchina "ricca" sappiamo farlo anche con una macchina "povera"
  - ossia, una macchina deterministica dotata di un solo nastro e che utilizza un alfabeto binario
- In effetti, tanti modelli calcolo sono stati definiti
  - e ciascuno di essi è stato dimostrato essere equivalente alla Macchina deterministica dotata di un solo nastro e che utilizza un alfabeto binario
- ma di questo parleremo più avanti...