Capitolo 2

Macchine di Turing

SOMMARIO

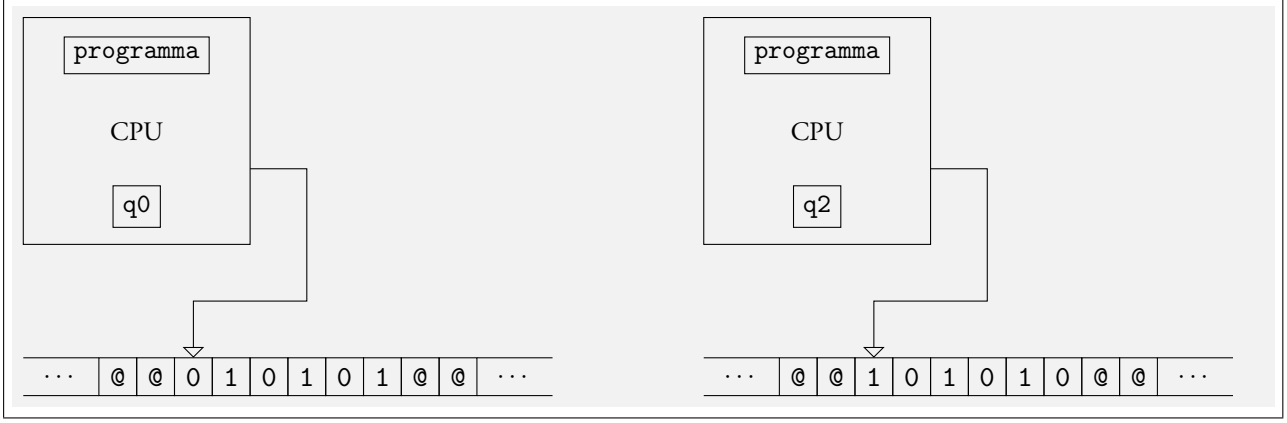
In questo capitolo introdurremo il modello di calcolo proposto dal logico matematico inglese Alan Turing, in un suo famoso articolo del 1936. Una volta definita la macchina di Turing, mostreremo diversi esempi di tali macchine introducendo due tecniche di programmazione comunemente utilizzate per sviluppare macchine di Turing.

2.1 Definizione di macchina di Turing

UNA MACCHINA di Turing è un modello di calcolo abbastanza simile agli odierni calcolatori. Analogamente a questi ultimi, infatti, essa possiede un'unità di elaborazione centrale (CPU, dall'inglese *Central Processing Unit*) e una memoria su cui poter leggere e scrivere. In particolare, la CPU di una macchina di Turing è composta da un **registro di stato**, contenente lo stato attuale della macchina, e da un **programma** contenente le istruzioni che essa deve eseguire. La memoria di una macchina di Turing è composta da un **nastro** infinito, suddiviso in **celle** e al quale la CPU può accedere attraverso una **testina** di lettura/scrittura (si veda la Figura 2.1).

Inizialmente, il nastro contiene la stringa di **input** preceduta e seguita da una serie infinita di simboli vuoti (in queste dispense, il **simbolo vuoto** è indicato con `@`), la testina è posizionata sul primo simbolo della stringa di input e la CPU si trova in uno stato speciale, detto **stato iniziale** (si veda la parte sinistra della Figura 2.1). Sulla base dello stato in cui si trova la CPU e del simbolo letto dalla testina, la macchina esegue un'istruzione del programma che può modificare il simbolo attualmente scandito dalla testina, spostare la testina a destra oppure a sinistra e cambiare lo stato della CPU. La macchina prosegue nell'esecuzione del programma fino a quando la CPU non viene a trovarsi in uno di un insieme di stati particolari, detti **stati finali**, oppure non esistono istruzioni del programma che sia possibile eseguire. Nel caso in cui il programma termini perché

Figura 2.1: rappresentazione schematica di una macchina di Turing.



la CPU ha raggiunto uno stato finale, il contenuto della porzione di nastro racchiusa tra la posizione della testina ed il primo `@` alla sua destra rappresenta la stringa di **output** (si veda la parte destra della Figura 2.1).

Esempio 2.1: una macchina di Turing per il complemento bit a bit

Consideriamo una macchina di Turing la cui CPU può assumere tre possibili stati `q0`, `q1` e `q2`, di cui il primo è lo stato iniziale e l'ultimo è l'unico stato finale. L'obiettivo della macchina è quello di calcolare la stringa binaria ottenuta eseguendo il complemento bit a bit della stringa binaria ricevuta in input: il programma di tale macchina è il seguente.

1. Se lo stato è `q0` e il simbolo letto non è `@`, allora complementa il simbolo letto e sposta la testina a destra.
2. Se lo stato è `q0` e il simbolo letto è `@`, allora passa allo stato `q1` e sposta la testina a sinistra.
3. Se lo stato è `q1` e il simbolo letto non è `@`, allora sposta la testina a sinistra.
4. Se lo stato è `q1` e il simbolo letto è `@`, allora passa allo stato `q2` e sposta la testina a destra.

La prima istruzione fa sì che la macchina scorra l'intera stringa di input, complementando ciascun bit incontrato, mentre le successive tre istruzioni permettono di riportare la testina all'inizio della stringa modificata. Ad esempio, tale macchina, con input la stringa binaria `010101`, inizia la computazione nella configurazione mostrata nella parte sinistra della Figura 2.1 e termina la sua esecuzione nella configurazione mostrata nella parte destra della Figura 2.1.

Alfabeti, stringhe e linguaggi

Un **alfabeto** è un qualunque insieme non vuoto $\Sigma = \{\sigma_1, \ldots, \sigma_k\}$: un **simbolo** è un elemento di un alfabeto. Una **stringa** su un alfabeto Σ è una sequenza finita $x = \langle \sigma_{i_1}, \ldots, \sigma_{i_n} \rangle$ di simboli in Σ (per semplicità, la stringa $\langle \sigma_{i_1}, \ldots, \sigma_{i_n} \rangle$ sarà indicata con $\sigma_{i_1} \ldots \sigma_{i_n}$): la **stringa vuota** è indicata con ϵ. L'insieme infinito di tutte le possibili stringhe su un alfabeto Σ è indicato con Σ^*. La **lunghezza** $|x|$ di una stringa $\sigma_{i_1} \ldots \sigma_{i_n}$ è il numero n di simboli contenuti in x: la stringa vuota ha lunghezza 0. Chiaramente, il numero di stringhe di lunghezza n su un alfabeto di k simboli è uguale a k^n. Date due stringhe x e y, la **concatenazione** di x e y (in simboli, xy) è definita come la stringa z formata da tutti i simboli di x seguiti da tutti i simboli di y: quindi, $|z| = |x| + |y|$. In particolare, la concatenazione di una stringa x con sè stessa h volte è indicata con x^h. Date due stringhe x e y, diremo che x è un **prefisso** di y se esiste una stringa z per cui $y = xz$. Dato un alfabeto Σ, un **linguaggio** L su Σ è un sottoinsieme di Σ^*: il complemento di L, in simboli L^c, è definito come $L^c = \Sigma^* - L$. Dato un alfabeto Σ, un ordinamento dei simboli di Σ induce un ordinamento delle stringhe su Σ nel modo seguente: (a) per ogni $n \geqslant 0$, le stringhe di lunghezza n precedono quelle di lunghezza $n + 1$, e (b) per ogni lunghezza, l'ordine è quello alfabetico. Tale ordinamento è detto **ordinamento lessicografico**. Ad esempio, dato l'alfabeto binario $\Sigma = \{0, 1\}$, le prime dieci stringhe nell'ordinamento lessicografico di Σ^* sono ϵ, 0, 1, 00, 01, 10, 11, 000, 001 e 010.

Formalmente, una macchina di Turing è definita specificando l'insieme degli stati (indicando quale tra di essi è quello iniziale e quali sono quelli finali) e l'insieme delle transizioni da uno stato a un altro: questo ci conduce alla seguente definizione basata sull'utilizzo del concetto di grafo.

Definizione 2.1: macchina di Turing

Una *macchina di Turing* è definita da un alfabeto Σ, con $\textvisiblespace \in \Sigma$, detto **alfabeto di lavoro**, e da un grafo etichettato $G = (V, E)$ tale che:

- $V = \{q_0\} \cup F \cup Q$ è l'insieme degli **stati** (q_0 è lo stato **iniziale**, F è l'insieme degli stati **finali** e Q è l'insieme degli stati che non sono iniziali né finali);
- $E \subseteq V \times V$ è l'insieme delle **transizioni** e, ad ogni transizione, è associata un'etichetta formata da una lista di triple $\langle \sigma, \tau, m \rangle$, in cui σ e τ sono simboli appartenenti a Σ e $m \in \{1, -1, 0\}$.

I simboli σ e τ che appaiono all'interno di una tripla dell'etichetta di una transizione indicano, rispettivamente, il simbolo attualmente letto dalla testina e il simbolo da scrivere, mentre il valore m specifica il movimento della testina: in particolare, 1 corrisponde allo spostamento a destra, -1 allo spostamento a sinistra e 0 a nessun spostamento.

Nel seguito di queste dispense, per evitare di dover specificare ogni volta quale sia lo stato iniziale e quali siano gli stati finali di una macchina di Turing, adotteremo la

Grafi

Un **grafo** G è una coppia di insiemi finiti $\langle V, E\rangle$ tale che $E \subseteq V \times V$: V è l'insieme dei **nodi**, mentre E è l'insieme degli **archi**. Se $\langle u, v\rangle \in E$, allora diremo che u è **collegato** a v: il numero di nodi a cui un nodo x è collegato è detto **grado in uscita** di x, mentre il numero di nodi collegati a x è detto **grado in ingresso** di x. Un grafo $G' = \langle V', E'\rangle$ è un **sotto-grafo** di un grafo $G = \langle V, E\rangle$ se $V' \subseteq V$ e $E' \subseteq E$. Un grafo **etichettato** è un grafo $G = \langle V, E\rangle$ con associata una funzione $\lambda : E \to \Lambda$ che fa corrispondere a ogni arco un elemento dell'insieme Λ delle etichette: tale insieme può essere, ad esempio, un insieme di valori numerici oppure un linguaggio. Un grafo $G = \langle V, E\rangle$ è detto **completo** se $E = V \times V$, ovvero se ogni nodo è collegato a ogni altro nodo. Dato un grafo G e due nodi v_0 e v_k, un **cammino** da v_0 a v_k di lunghezza k è una sequenza di archi $\langle v_0, v_1\rangle$, $\langle v_1, v_2\rangle$, $\ldots$, $\langle v_{k-1}, v_k\rangle$ tale che, per ogni i e j con $0 \leqslant i < j \leqslant k$, $v_i \neq v_j$: se $v_0 = v_k$ allora la sequenza di archi è detta essere un **ciclo**.

convenzione di rappresentare lo stato iniziale con un doppio cerchio e gli stati finali con un rettangolo.

Esempio 2.2: il grafo della macchina per il complemento bit a bit

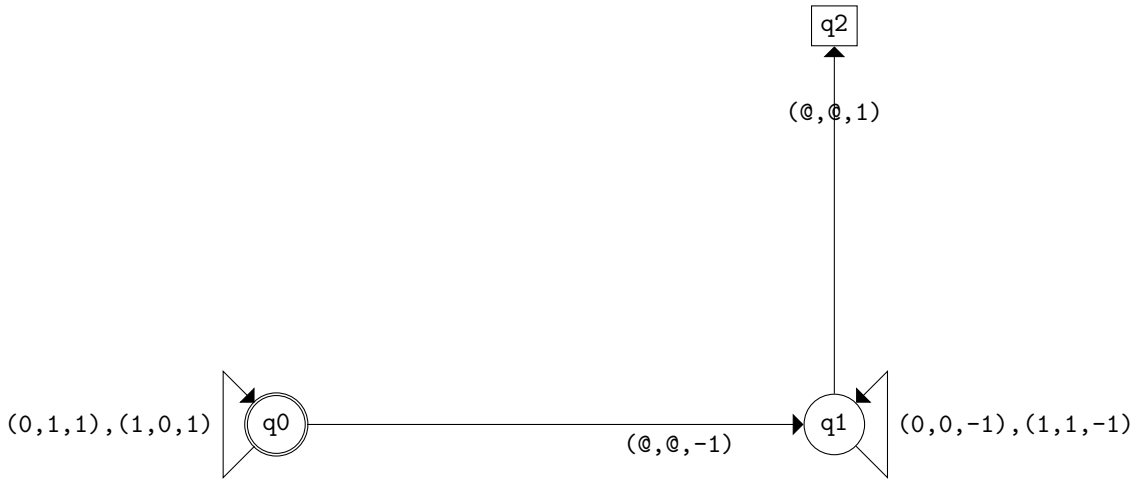
Il grafo corrispondente alla macchina di Turing introdotta nell'Esempio 2.1 è mostrato nella Figura 2.2. Come si può vedere, ciascuna transizione corrisponde a un'istruzione del programma descritto nell'Esempio 2.1. Ad esempio, l'arco che congiunge lo stato `q0` allo stato `q1` corrisponde alla seconda istruzione: in effetti, nel caso in cui lo stato attuale sia `q0` e il simbolo letto sia `@`, allora il simbolo non deve essere modificato, la testina si deve spostare a sinistra e la CPU deve passare nello stato `q1`.

2.1.1 Rappresentazione tabellare di una macchina di Turing

Oltre alla rappresentazione grafica a cui abbiamo fatto riferimento nella Definizione 2.1, una macchina di Turing viene spesso specificata facendo uso di una rappresentazione tabellare. In base a tale rappresentazione, a una macchina di Turing viene associata una tabella di tante righe quante sono le triple contenute nelle etichette degli archi del grafo corrispondente alla macchina. Se $\langle \sigma, \tau, m\rangle$ è una tripla contenuta nell'etichetta dell'arco che collega lo stato q allo stato p, la corrispondente riga della tabella sarà formata da cinque colonne, contenenti, rispettivamente, q, σ, p, τ e m. In altre parole, la prima colonna della tabella specifica lo stato attuale, la seconda specifica il simbolo letto, la terza specifica il nuovo stato, la quarta specifica il simbolo che deve essere scritto e la quinta specifica il movimento della testina.

Osserviamo che, in generale, la tabella non contiene tante righe quante sono le possibili coppie formate da uno stato e un simbolo dell'alfabeto di lavoro della macchina,

Figura 2.2: macchina di Turing per il complemento bit a bit.



in quanto per alcune (generalmente molte) di queste coppie il comportamento della macchina può non essere stato specificato.

Esempio 2.3: rappresentazione tabellare della macchina per il complemento bit a bit

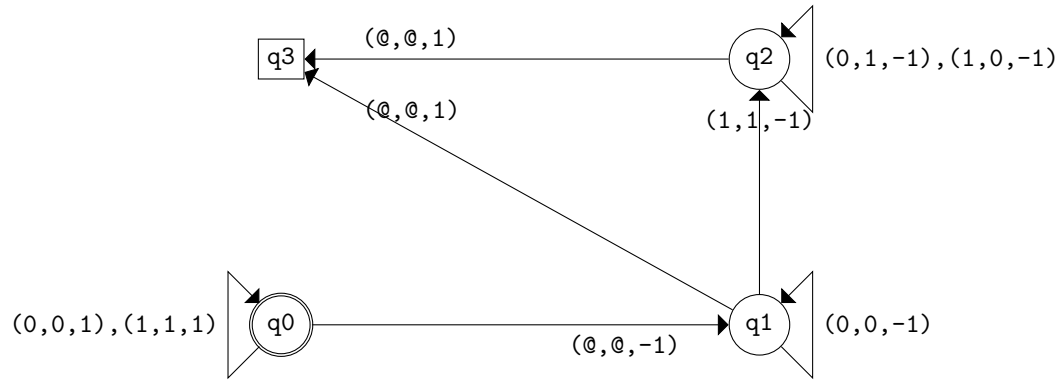
Facendo riferimento alla macchina di Turing introdotta nell'Esempio 2.1 e mostrata nella Figura 2.2, la corrispondente rappresentazione tabellare di tale macchina è la seguente.

stato	simbolo	stato	simbolo	movimento
q0	0	q0	1	1
q0	1	q0	0	1
q0	@	q1	@	-1
q1	0	q1	0	-1
q1	1	q1	1	-1
q1	@	q2	@	1

2.2 Esempi di macchine di Turing

IN QUESTA SEZIONE mostriamo alcuni esempi di macchine di Turing. Vedremo molti altri esempi nel resto delle dispense, ma già alcuni di quelli che verranno presentati in questa sezione evidenziano il tipico comportamento di una macchina di Turing:

Figura 2.3: macchina per il calcolo del complemento a due.



in generale, infatti, queste macchine implementano una strategia a *zig-zag* che consente loro di attraversare ripetutamente il contenuto del nastro, combinata con una strategia di *piggy-backing* che consiste nel "memorizzare" nello stato il simbolo o i simboli che sono stati letti.

2.2.1 Una macchina per il complemento a due

Definiamo una macchina che calcola la funzione di complemento a due. Ricordiamo che il complemento a due di una stringa binaria può essere calcolato individuando anzitutto il simbolo `1` più a destra, e complementando poi tutti i simboli alla sinistra di quest'ultimo. Pertanto, una macchina che calcola tale funzione può operare nel modo seguente (si veda la Figura 2.3).

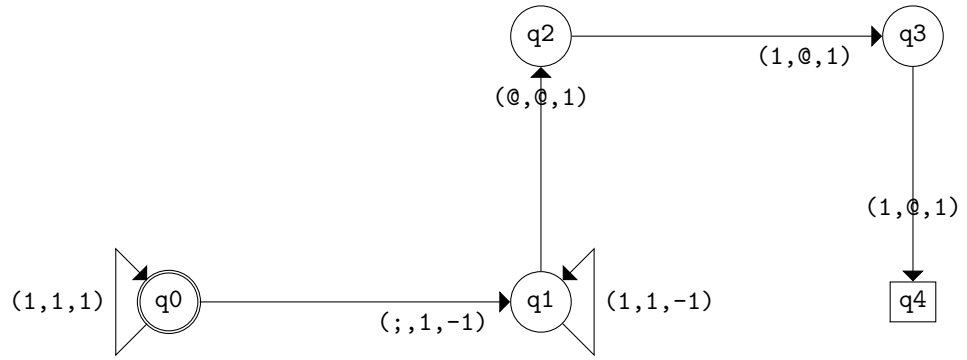
1. Attraversa la stringa in input fino a raggiungere il primo `@` alla sua destra.

2. Si muove verso sinistra fino a trovare e superare un `1`: se tale `1` non esiste (ovvero, la stringa in input è costituita da soli simboli `0`), la macchina può terminare la sua esecuzione posizionando la testina sul simbolo `0` più a sinistra.

3. Si muove verso sinistra complementando ogni simbolo incontrato, fino a raggiungere il primo `@`.

4. Posiziona la testina sul primo simbolo diverso da `@`.

Figura 2.4: macchina per per il calcolo della somma di due numeri interi.



2.2.2 Una macchina per la somma di due numeri interi

Definiamo una macchina che calcola la somma di due numeri interi non negativi. A tale scopo possiamo assumere che un numero naturale x sia rappresentato da $x+1$ simboli `1`: quindi, 0 è rappresentato da `1`, 1 è rappresentato da `11`, 2 è rappresentato da `111` e così via. Inoltre, assumiamo che i due numeri interi siano inizialmente presenti sul nastro uno di seguito all'altro e separati dal simbolo `;`. Una macchina che calcola la somma di due numeri naturali può operare nel modo seguente (si veda la Figura 2.4).

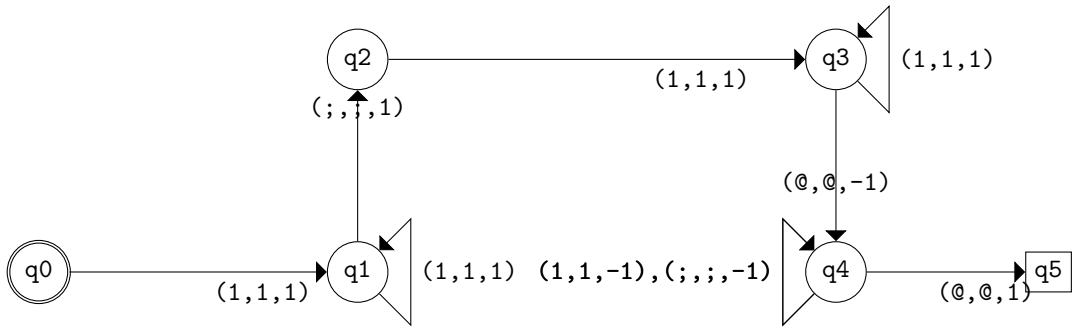
1. Attraversa la stringa in input fino a raggiungere il primo `;` alla sua destra: sostituisce tale simbolo con `1`.

2. Si muove verso sinistra fino a trovare il primo `@` e cancella i due simboli `1` alla sua destra (che sicuramente esistono).

3. Posiziona la testina sul primo simbolo `1`.

Osserviamo che, negli esempi precedenti, abbiamo assunto che l'input sia fornito in modo corretto. Ad esempio, la descrizione della macchina di Turing che calcola la somma di due numeri interi, assume che l'input sia sempre costituito da una sequenza non vuota di simboli `1` seguita dal simbolo `;` e da un'altra sequenza non vuota di simboli `1`. Se così non fosse, potrebbero verificarsi diverse situazioni di errore. Ad esempio, se la stringa in ingresso fosse `1011;11`, si avrebbe che la macchina nello stato `q0` leggerebbe il simbolo `0`: in tal caso, la macchina si arresterebbe in modo anomalo, in quanto non sarebbe definita una transizione in corrispondenza di tale situazione. Se, invece, la stringa in ingresso fosse `11111;`, allora la macchina, invece di segnalare in qualche modo la non

Figura 2.5: macchina per la verifica della correttezza dell'input.



presenza del secondo numero intero, si arresterebbe nello stato finale `q4` producendo come output la stringa `1111` (ovvero, $3 = 4 + (-1)$).

In tutti gli esempi che faremo in queste dispense, comunque, sarà sempre facile verificare l'esistenza di una macchina di Turing in grado di controllare che l'input sia stato fornito in modo corretto. Ad esempio, una macchina di Turing che verifichi che l'input sia costituito da una sequenza non vuota di simboli `1` seguita dal simbolo `;` e da un'altra sequenza non vuota di simboli `1`, può facilmente essere definita nel modo seguente (si veda la Figura 2.5).

1. Controlla che il primo simbolo sia un `1`: se così non fosse, termina segnalando un errore (ad esempio, non definendo le transizioni corrispondenti a simboli diversi da `1`).

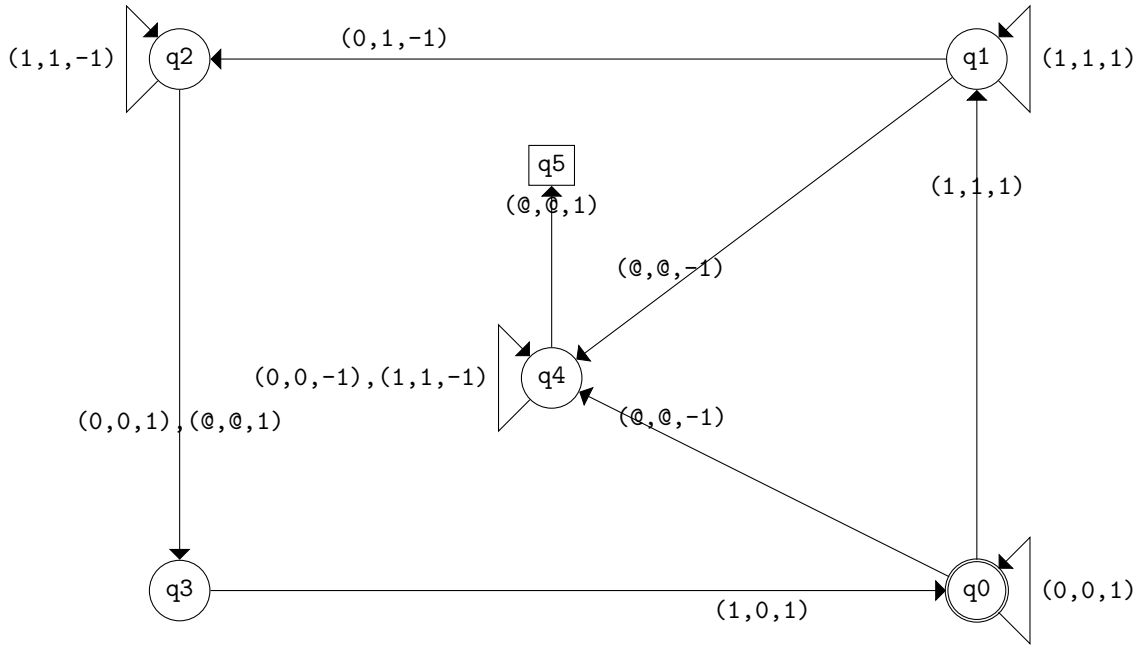
2. Scorre la stringa in input verso destra fino al primo simbolo `;`, verificando allo stesso tempo che i simboli letti prima del `;` siano tutti uguali a `1`: se così non fosse, termina segnalando un errore.

3. Supera il simbolo `;` e scorre la stringa in input verso destra fino al primo simbolo `@`, verificando allo stesso tempo che i simboli letti prima del `@` siano tutti uguali a `1`: se così non fosse, termina segnalando un errore.

4. Posiziona nuovamente la testina sul primo simbolo della stringa in input.

In generale, esplicitare una macchina di Turing che verifichi la correttezza dell'input all'interno di un esempio, complicherebbe solamente l'esposizione senza aggiungere nulla di particolarmente interessante alla comprensione dell'esempio stesso. Per questo motivo, nel resto di queste dispense continueremo ad assumere che l'input fornito in ingresso a una macchina di Turing sia sempre corretto.

Figura 2.6: macchina per l'ordinamento di stringhe binarie.



2.2.3 Una macchina per l'ordinamento di stringhe binarie

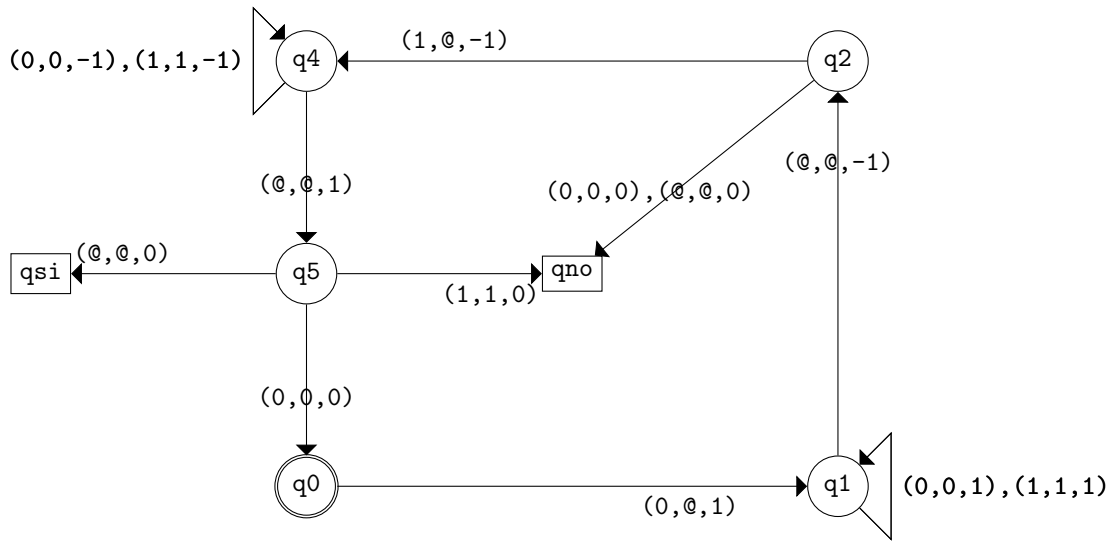
Data una stringa x sull'alfabeto $\Sigma = \{0, 1\}$, la stringa ordinata corrispondente a x è la stringa su Σ ottenuta a partire da x mettendo tutti i simboli 0 prima dei simboli 1. Ad esempio, la stringa ordinata corrispondente a 0101011 è la stringa 0001111.

Definiamo una macchina di Turing che, data in input una sequenza di simboli 0 e 1, termina producendo in output la sequenza ordinata corrispondente all'input. La macchina opera nel modo seguente (si veda la Figura 2.6).

1. Scorre il nastro verso destra alla ricerca di un simbolo 1. Se non lo trova, vuol dire che la testina è posizionata sul primo simbolo @ successivo alla stringa che è ora ordinata: in tal caso, scorre il nastro verso sinistra fino a posizionare la testina sul primo simbolo diverso da @ e termina l'esecuzione.

2. Scorre il nastro verso destra alla ricerca di un simbolo 0. Se non lo trova, vuol dire che la testina è posizionata sul primo simbolo @ successivo alla stringa che è ora ordinata: in tal caso, scorre il nastro verso sinistra fino a posizionare la testina sul primo simbolo diverso da @ e termina l'esecuzione.

Figura 2.7: macchina per 0^n1^n.



3. Complementa il simbolo `0` e scorre il nastro verso sinistra alla ricerca di un simbolo `0` oppure di un simbolo `@`, posizionando la testina immediatamente dopo tale simbolo, ovvero, sul simbolo `1` trovato in precedenza.

4. Complementa il simbolo `1` e ricomincia dal primo passo.

2.2.4 Una macchina per 0^n1^n

Consideriamo il seguente linguaggio L sull'alfabeto $\Sigma = \{0, 1\}$:

$$L = \{0^n1^n : n > 0\}$$

Ad esempio, la stringa `00001111` appartiene a L, mentre la stringa `0001111` non vi appartiene. Tale linguaggio, come vedremo nella seconda parte di queste dispense, svolge un ruolo molto importante nell'ambito della teoria dei linguaggi formali, in quanto rappresenta il classico esempio di linguaggio generabile da una grammatica contestuale, ma non generabile da una grammatica regolare.

Definiamo una macchina di Turing che, data in input una sequenza di simboli `0` seguita da una sequenza di simboli `1`, termina nello stato `qsi` se il numero di simboli `0` è uguale a quello dei simboli `1`, altrimenti termina nello stato `qno`. Ad esempio, ricevendo

in input la stringa `00001111` la macchina termina nello stato `qsi`, mentre ricevendo in input la stringa `0001111` la macchina termina nello stato `qno`. In questo caso, la posizione finale della testina è ininfluente, in quanto l'output prodotto dalla macchina viene codificato attraverso lo stato finale. La macchina opera nel modo seguente (si veda la Figura 2.7).

1. Partendo dall'estremità sinistra della stringa in input, sostituisce un simbolo `0` con il simbolo `@` e si muove verso destra, ignorando tutti i simboli `0` e `1`, fino ad incontrare il simbolo `@`.

2. Verifica che il simbolo immediatamente a sinistra del simbolo `@` sia un `1`: se così non è, allora termina nello stato `qno`.

3. Sostituisce il simbolo `1` con il simbolo `@` e scorre il nastro verso sinistra fino ad incontrare un simbolo `@`.

4. Se il simbolo immediatamente a destra del `@` è anch'esso un simbolo `@`, allora termina nello stato `qsi`. Se, invece, è un `1`, allora termina nello stato `qno`. Altrimenti, ripete l'intero procedimento a partire dal primo passo.

Osserviamo che, in base alla descrizione precedente, la stringa in ingresso viene modificata dalla macchina in modo tale da non poter essere successivamente ricostruita. Non è difficile, comunque, modificare la macchina in modo che, facendo uso di simboli ausiliari, sia possibile ricostruire, al termine del procedimento sopra descritto, la stringa ricevuta in ingresso.

2.2.5 Una macchina per la differenza propria

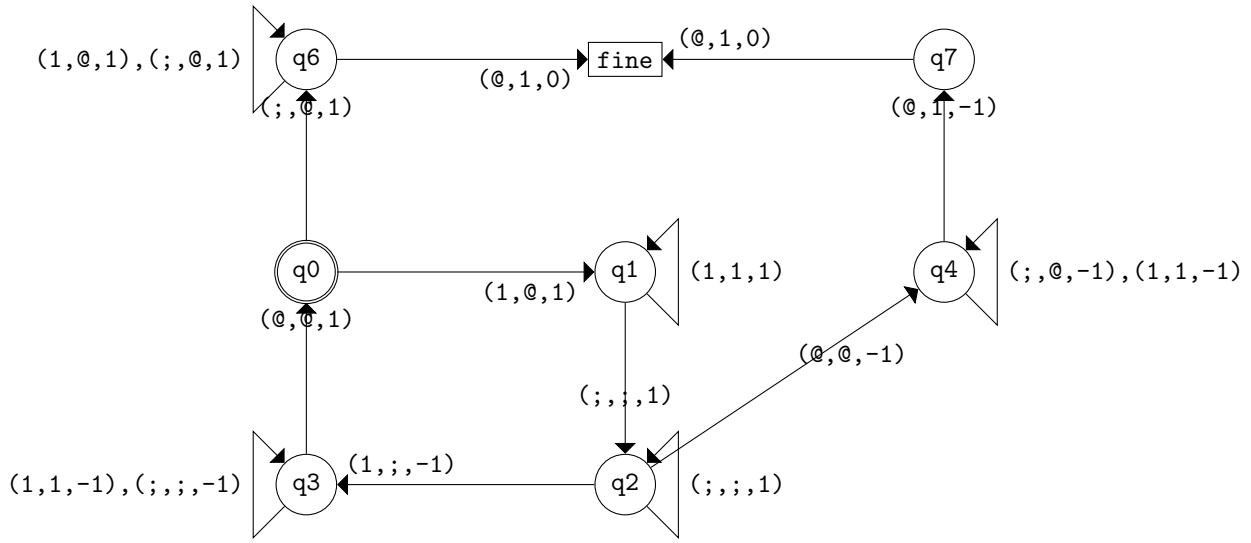
Abbiamo già visto come calcolare la somma di due numeri interi non negativi rappresentati come sequenze non vuote di simboli `1`. Mostriamo ora che anche la funzione di differenza è una funzione calcolabile da una macchina di Turing.

A tale scopo definiamo, anzitutto, la funzione di *differenza propria* nel modo seguente. Dati due numeri interi non negativi m e n,

$$m \ominus n = \begin{cases} m - n & \text{se } m > n, \\ 0 & \text{altrimenti} \end{cases}$$

Definiamo ora una macchina di Turing che, data in input una sequenza di $m + 1$ simboli `1` seguita da un simbolo `;` e da una sequenza di $n + 1$ simboli `1`, termina producendo in output la sequenza di $(m \ominus n) + 1$ simboli `1`. Ad esempio, ricevendo in input la stringa `11111;111`, la macchina produce in output la stringa `111`, mentre,

Figura 2.8: macchina per la differenza propria.



ricevendo in input la stringa `111;11111`, la macchina produce in output `1`, ovvero la rappresentazione unaria del numero 0.

Intuitivamente, la macchina opera nel modo seguente (si veda la Figura 2.8). Sostituisce il simbolo `1` più a sinistra con un `@` e scorre la testina verso destra, ignorando tutti i simboli `1`, fino a trovare il primo simbolo `;`. A questo punto, la macchina continua a scorrere la testina verso destra, ignorando tutti i simboli `;`, fino a incontrare un simbolo `1`, che viene sostituito con il simbolo `;`. Successivamente, la macchina si muove verso sinistra fino a incontrare un `@`, posiziona la testina sul simbolo alla sua destra e ripete l'intero procedimento.

L'iterazione appena descritta termina in uno dei seguenti due casi:

- Cercando un simbolo `1` verso destra, la macchina incontra un `@`: ciò vuol dire che $m > n$. In tal caso, tutti gli $n + 1$ simboli `1` alla destra del simbolo `;` nella stringa di input sono stati sostituiti con il simbolo `;` e $m - n - 1$ simboli `1` si trovano alla sinistra del simbolo `;` più a sinistra. La macchina sostituisce gli $n + 2$ simboli `;` con altrettanti simboli `@` e i due simboli `@` più a sinistra con due simboli `1`. Il risultato è che sul nastro rimangono $m - n + 1$ simboli `1`.

- All'inizio del procedimento, la macchina non trova un simbolo `1` da sostituire con

un `@`: ciò vuol dire che $m \leqslant n$. In tal caso, la macchina sostituisce tutti i simboli `;` e `1` ancora presenti sul nastro con un `@` e termina con la testina posizionata su un unico simbolo `1` (che è la risposta corretta, dato che, essendo $m \leqslant n$, $m \ominus n = 0$).

Esercizi

Esercizio 2.1. Immaginando di rappresentare i numeri interi non negativi come descritto nel testo (ovvero, in modo che la rappresentazione del numero non negativo x sia ottenuta mediante una sequenza di $x + 1$ simboli `1`), si scriva una macchina di Turing che calcola la funzione s, tale che, per ogni x, $s(x) = x + 1$.

Esercizio 2.2. Sempre facendo riferimento alla rappresentazione in unario dei numeri interi non negativi, si scriva una macchina di Turing che calcola la funzione z, tale che, per ogni x, $z(x) = 0$.

Esercizio 2.3. Sempre facendo riferimento alla rappresentazione in unario dei numeri interi non negativi e immaginando di separare argomenti multipli mediante il simbolo `;`, per ogni $i \leqslant 1$, si scriva una macchina di Turing che calcola la funzione u, tale che, per ogni $(n + 1)$-tupla di numeri interi non negativi $i, x_1, \ldots, x_n$ con $n \leqslant i$, $u(i, x_1, \ldots, x_n) = x_i$.

Esercizio 2.4. Definire una macchina di Turing che, data in input una sequenza di $n+1$ simboli `1` con $n \geqslant 0$, produca in output la rappresentazione binaria di n.

Esercizio 2.5. Definire una macchina di Turing che, data in input la rappresentazione binaria di un numero intero non negativo n, produca in output la sequenza di $n + 1$ simboli `1`.

Esercizio 2.6. Definire una macchina di Turing che, data in input la rappresentazione binaria di due numeri interi non negativi n e m, produca in output la rappresentazione binaria del minimo tra n e m.

Esercizio 2.7. Definire una macchina di Turing che, data in input la rappresentazione binaria di due numeri interi non negativi n e m, produca in output la rappresentazione binaria di $n + m$.