Teoremi Informatica Teorica

Zbirciog Ionut Georgian

June 3, 2024

Indice

1 Teoremi Dispensa 2

1.1 Teorema a pag. 5

Per ogni macchina di Turing non deterministica NT esiste una macchina di Turing detreministica T tale che, per ogni possibile input x di NT , l'esito della computazione $NT(x)$ coincide con l'esito della computazione di $T(x)$.

Dimostrazione: Eseguiamo una simulazione della macchina non deterministica NT mediante una macchina deterministica T. La simulazione consiste in una visita in ampiezza[1] dell'albero delle computazioni di NT basata sulla tecnica *coda di rondine con ripetizioni.* Partiamo dallo stato globale $SG(T, x, 0)$ e simuliamo tutte le computazione di lunghezza 1. Se tutte le computazioni terminano in q_R allora T rigetta, se almeno una computazione termina in q_A allora T accetta, altrimenti ricominciamo da capo eseguendo tutte le computazioni di lunghezza 2 e così via.

[1]Perché non in profondità? Non possiamo fare una visità in profondità perché non sappiamo la lunghezza di ciascuna computazione, in quanto potrebbero anche non finire.

2 Teoremi Dispensa 3

2.1 Teorema a pag. 3

Un linguaggio $L \subseteq \Sigma^\star$ è decidibile se e soltanto se L e L^c sono accettabili.

Dimostrazione:

($\Rightarrow$ Se L è decidibile allora esiste una macchina di Turing T deterministica tale che $\forall x \in \Sigma^\star$, $T(x) = q_A \Leftrightarrow x \in L \wedge T(x) = q_R \Leftrightarrow x \in L^c$. Osserviamo dunque che T accetta L.

Da T, deriviamo ora $T^{'}$ aggiungendo le seguenti quintuple:

$$\langle q_A, x, x, q_R^{'}, stop\rangle \wedge \langle q_R, x, x, q_A^{'}, stop\rangle \ \forall x \in \Sigma \cup \square$$

L'esecuzione di $T^{'}$ è simile a quella di T, solo che gli stati di accettazione e rigetto sono stati invertiti, in questo modo se T accetta x allora $T^{'}$ rigetta x, mentre se T rigetta x, $T^{'}$ accetta x, dunque T' accetta L^c.

$\Leftarrow$) Se L e L^c sono accettabili allora esistono due macchine di Turing T_1 e T_2 tali che, $\forall x \in \Sigma^\star \, T_1(x) = q_A \Leftrightarrow x \in L \wedge T_2(x) = q_A \Leftrightarrow x \in L^c$. Non esendo specificato l'esito della computazione nel caso in cui $x \notin L$ e $x \notin L^c$ definiamo la macchina T che, simulando T_1 e T_2 decide L nel seguento modo[2]:

1. Esegui una singola istruzione di T_1 sul nastro 1: se $T_1(x) = q_A$ allora $T(x) = q_A$, altrimenti esegui il passo (2).
2. Esegui una singola istruzione di T_2 sul nastro 2: se $T_2(x) = q_A$ allora $T(x) = q_R$, altrimenti esegui il passo (1).

Se $x \in L$, allora prima o poi, al passo (1), T_1 entrerà nello stato di accettazione, portando T ad accettare. Se $x \in L^c$, allora prima o poi, al passo (1), T_1 entrerà nello stato di accettazione, portando T a rigettare.

2.2 Teorema a pag. 4

Un linguaggio L è decidibile se e soltanto se la funzione χ_L è calcolabile.

Dimostrazione:

($\Rightarrow$ Se L è decidibile allora esiste una macchina di Turing T deterministica di tipo **riconoscitore** tale che $\forall x \in \Sigma^\star$, $T(x) = q_A \Leftrightarrow x \in L \wedge T(x) = q_R \Leftrightarrow x \in L^c$. A partire da T definiamo una macchina di Turing $T^{'}$ di tipo trasduttore a 2 natri, con input $x \in \Sigma^\star$ che opera nel seguente modo:

1. Sul primo nastro simula $T(x)$.
2. Se $T(x)$ termina nello stato q_A allora $T'(x)$ scrive sul nastro di output il valore 1, altrimenti scrive il valore 0 e poi termina.

Osserviamo che poiché L è decidibile il passo (1) termina sempre per ogni input x. Se $x \in L$ allora $T(x) = q_A$ e $T^{'}(x)$ scrive 1 sul nastro di output. Se $x \notin L$ allora $T(x) = q_R$ e $T^{'}(x)$ scrive 0 sul nastro di output. Questo dimostra che χ_L è calcolabile.

$\Leftarrow$) Se χ_L è calcolabile e per costruzione anche totale allora esiste una macchina di Turing T di tipo **trasduttore**, che per ogni $x \in \Sigma^\star$, calcola $\chi_L(x)$. A partire da T definiamo $T^{'}$ di tipo riconoscitore a 2 natri, con input $x \in \Sigma^\star$ che opera nel seguente modo:

1. Sul primo nastro simula $T(x)$ scrivendo il risultato sul secondo nastro.
2. Se sul secondo nastro c'é scritto 1 allora $T^{'}(x) = q_A$, altrimenti nello stato q_R.

Osserviamo che poiché χ_L è calcolabile il passo (1) termina sempre per ogni input x. Se $\chi_L(x) = 1$ allora (1) termina scrivendo 1 sul secondo nastro e $T^{'}(x) = q_A$. Se $\chi_L(x) = 0$ allora (1) termina scrivendo 0 sul secondo nastro e $T^{'}(x) = q_R$. Questo dimostra che L è decidibile.

[2] Osserviamo che non possiamo simulare T_1 e T_2 "blackbox", in quanto non sappiamo se la loro computazione termina o meno.

2.3 Teorema a pag. 5

Se la funzione $f : \Sigma^\star \to \Sigma_1^\star$ è totale e calcolabile allora il linguaggio $L_f \subseteq \Sigma^\star \times \Sigma_1^\star$ è decidibile.

Dimostrazione: Poiché f è calcolabile e totale allora esiste una macchina di Turing trasduttore che calcola $f(x) \forall x \in \Sigma^\star$. A partire da T definiamo una macchina di Turing $T^{'}$ riconoscitore a due nastri con input $\langle x, y \rangle$ dove $x \in \Sigma^\star$ e $y \in \Sigma_1^\star$, che opera nel seguente modo:

1. Sul nastro 1 è scritto l'input $\langle x, y \rangle$.
2. Sul nastro 2 simula $T(x)$, scrivendovi il risultato z.
3. Se $z = y$ allora $T^{'}(x) = q_A$ altrimenti va in q_R.

Osserviamo che, poiché f è totale e calcolabile il passo (2) termina per ogni input $x \in \Sigma\star$. Se $f(x) = z = y$ allora $T^{'}(x)$ termina in q_A. Se $f(x) = z \neq y$ allora $T^{'}(x)$ termina in q_R. Questo dimostra che L_f è decidibile.

2.4 Teorema a pag. 5

Sia $f : \Sigma^\star \to \Sigma_1^\star$ una funzione. Se il linguaggio $L_f \subseteq \Sigma^\star \times \Sigma_1^\star$ è decidibile allora f è calcolabile[3].

Dimostrazione: Poiché $L_f \subseteq \Sigma^\star \times \Sigma_1^\star$ è decidibile, esiste una macchina di Turing riconoscitore T, tale che $\forall x \in \Sigma^\star$ e $\forall y \in \Sigma_1^\star$, $T(x) = q_A$ se $y = f(x)$ e $T(x) = q_R$ se $y \neq f(x)$. A partire da T definiamo una macchina di Turing trasduttore $T^{'}$ con input $x \in \Sigma^\star$ che opera nel seguente modo:

1. Scrive $i = 0$ sul nastro 1.
2. Enumera tutte le stringhe $y \in \Sigma_1^\star$ di lunghezza pari al valore scritto sul primo nastro, simulando per ciascuna stringa $T(x, y)$.
   (a) Sia y la prima stringa di lunghezza i non ancora enumerata, allora scrive y sul secondo nastro.
   (b) Sul terzo nastro, esegue la computazione $T(x, y)$.
   (c) Se $T(x, y) = q_A$ allora scrive y sul nastro di output eventualmente incrementando i se y era l'ultima stringa, torna al passo (2).

Poiché L_f è decidibile il passo (b) termina per ogni input (x, y). Se x appartiene al dominio di f, allora $\exists y \in \Sigma_1^\star$ tale che $y = f(x)$, e quindi $(x, y) \in L_f$. Allora prima o poi la strigna y verrà scritta sul secondo nastro e $T(x, y) = q_A$. Questo dimostra che f è calcolabile.

2.5 Teorema a pag. 7

Per ogni programma scritto in accordo con il linguaggio di programmazione **PascalMinimo**, esiste un macchina di Turing T di tipo trasduttore che scrive sul nastro di output lo stesso valore fornito in output dal programma.

Dimostrazione omessa

2.6 Teorema a pag. 9

Per ogni macchina di Turing deterministica T di tipo riconoscitore ad un nastro esiste un programma P scritto in accordo alle regole del linguaggio **PascalMinimo** tale che, per ogni stringa x, se $T(x)$ termina nello stato fiale $q_F \in \{q_A, q_R\}$ allora P con input x restituisce q_F in output.

Dimostrazione omessa

[3] Osserviamo che non possiamo invertire del tutto il teorema precendente, dalla decidibilità di L_f possiamo dedurre solo la calcolabilità di f

3 Teoremi Dispensa 5

3.1 Teorema a pag. 2

L'insieme T delle macchine di Turing definite sull'alfabeto $\{0,1\}$ e dotate di un singolo nastro (più l'eventuale nastro di output) è numerabile

Dimostrazione: Per dimostrare tale teorema, dobbiamo trovare una biezione tra l'insieme T e l'insieme $\mathbb{N}$. Tale biezione non è altro che una etichettatura degli elementi dell'insieme con etichette appartenenti ad $\mathbb{N}$, ossia, una numerazione degli elementi dell'insieme. Sia T una macchina di Turing e β_T la sua codifica. Dunque, rappresentiamo T con la parola $\beta_T \in \Sigma^\star$, con $\Sigma = \{0,1,\oplus,\otimes,-,f,s,d\}$ come segue:

$$\beta_T = b(q_0) \;-\; b(q_1) \otimes b(q_{11}) \;-\; b_{11} \;-\; b_{12} \;-\; b(q_{12}) \;-\; m_1 \oplus \cdots \oplus b(q_{h1}) \;-\; b_{h1} \;-\; b_{h2} \;-\; b(q_{h2}) \;-\; m_h$$

Ora, effettuando le seguenti sostituzione in β_T, otteniamo una stringa in $\mathbb{N}$

- "s" con "5"
- "f" con "6"
- "d" con "7"
- "-" con "4"
- "$\otimes$" con "3"
- "$\oplus$" con "2"

Inoltre, dato che la stringa può iniziare con un "0", allora premettiamo il carattere "8" alla stringa ottenuta. La parola in $\{0,1,2,3,4,5,6,7,8\}^\star$ così ottenuta, può, ovviamente, essere considerata come un numero espresso in notazione decimale, ovvero il numero $v(T) \in \mathbb{N}$ associato univocamente a T.

3.2 Teorema a pag. 4 (Halting Problem)

Definiamo il seguente linguaggio L_H in questo modo:

$$L_H = \{(i,x) : i \text{ è la codifica di una } TM \;\wedge\; T_i(x) \text{ termina}\}$$

Il linguaggio L_H è accettabile.

Dimostrazione: Dobbiamo dimostrare che esiste una macchina di Turing T tale che, per ogni input $(i,x) \in \mathbb{N} \times \mathbb{N}$, $T(i,x) = q_A$ se e soltanto se $(i,x) \in L_H$.
Definiamo $U^{'}$ una macchina di Turing universale modificata con input (i,x). Tale macchina opera nel seguente modo:

1. Verifica se i è la codifica di una macchina di Turing. Se non lo è allora $U^{'}(i,x) = q_R$.
2. Simula $U(i,x)$, se termina in q_A o in q_R allora $U^{'}(x) = q_A$.

$U^{'}$ non sa decidere L_H^c, perciò lo accetta solo.

3.3 Teorema a pag. 4 (Halting Problem)

Il linguaggio L_H non è decidibile.

Dimostrazione: Supponiamo che L_H sia decidibile. Ovvero L_H è decidibile $\Leftrightarrow$ L_H e L_H^c sono accettabili. Prima abbiamo dimostratro che L_H è accettabile, dunque ci rimane solo L_H^c.

$$L_H^c = \{(i,x) : i \text{ non è la codifica di una } TM \;\vee\; T_i(x) \text{ non termina}\}$$

Allora, se L_H è decidibile deve esistere una macchina di Turing T tale che

$$T(i,x) = \begin{cases} q_A \Leftrightarrow (i,x) \in L_H \\ q_R \Leftrightarrow (i,x) \notin L_H \end{cases}$$

Da T **deriviamo** $T^{'}$ che terminando su ogni input, accetta tutte e sole le coppie $(i,x) \in \mathbb{N} \times \mathbb{N} \setminus L_H$, ossia L_H^c.

$$T^{'}(i,x) = \begin{cases} q_A \Leftrightarrow (i,x) \notin L_H \\ q_R \Leftrightarrow (i,x) \in L_H \end{cases}$$

Da $T^{'}$ **deriviamo** $T^{''}$ in questo modo:

$$T^{''}(i,x) = \begin{cases} q_A \Leftrightarrow T^{'}(i,x) = q_A \\ \text{non termina se } T^{'}(i,x) = q_R \end{cases}$$

Ovvero

$$T^{''}(i,x) = \begin{cases} q_A \Leftrightarrow (i,x) \notin L_H \\ \text{non termina se } (i,x) \in L_H \end{cases}$$

Da $T^{''}$ **deriviamo** T^* in questo modo:

$$T^*(i) = T^{''}(i,i) = \begin{cases} q_A \Leftrightarrow (i,i) \notin L_H \\ \text{non termina se } (i,i) \in L_H \end{cases}$$

$$L_H = \{(i,x) : i \text{ è la codifica di una } TM \ \wedge \ T_i(x) \text{ termina}\}$$
$$L_H^c = \{(i,x) : i \text{ non è la codifica di una } TM \ \vee \ T_i(x) \text{ non termina}\}$$

Se T esiste $\Rightarrow$ T^* esiste $\Rightarrow$ $\exists k \in \mathbb{N}$ tale che $T^* = T_k$ (k è la codifica numerica di T^*).
Se $\mathbf{T_k(k)} = \mathbf{T^*(k)}$ accettasse, allora $T^{'}(k,k)$ dovrebbe accettare anch'essa. Ma se $T^{'}(k,k)$ accetta, allora $(k,k) \notin L_H$, ossia, per definizione di L_H, $T_k(k)$ non termina poiché $(k,k) \notin L_H$. Allora $T^*(k)$ non può accettare e, dunque, necessariamente non termina. Ma, se $T^*(k)$ non termina, allora $T^{'}(k,k)$ rigetta e, quindi, $(k,k) \in L_H$. Dunque, per definizione di L_H, $T_k(k)$ termina. Quindi, in entrambi le ipotesi, $T_k(k)$ termina o non termina, portando ad una contraddizione. Allora T^* non può esistere $\Rightarrow$ $T^{''}$ non può esistere, $\Rightarrow$ $T^{'}$ non può esistere e di conseguenza T. Quindi se T non esiste, L_H non è decidibile.

3.4 Teorema a pag. 6

Se $L_1 e L_2$ sono due linguaggi accettabili, allora $L_1 \cup L_2$ è un linguaggio accettabile.
Se $L_1 e L_2$ sono due linguaggi decidibili, allora $L_1 \cup L_2$ è un linguaggio decidibile.

Dimostrazione:

3.5 Teorema a pag. 6

Se $L_1 e L_2$ sono due linguaggi accettabili, allora $L_1 \cap L_2$ è un linguaggio accettabile.
Se $L_1 e L_2$ sono due linguaggi decidibili, allora $L_1 \cap L_2$ è un linguaggio decidibile.

Dimostrazione:

4 Teoremi Dispensa 6

4.1 Teorema a pag. 3

Sia T una macchina di Turing deterministica, definita su un alfabeto $\Sigma \setminus \square$ e un insieme di stati Q, e sia $x \in \Sigma^\star$ tale che $T(x)$ termina, allora:

$$dspace(T,x) \leq dtime(T,x) \leq dspace(T,x)|Q|(|\Sigma|+1)^{dspace(T,x)}$$

Dimostrazione:

1. $dspace(T,x) \leq dtime(T,x)$
   Banalmente, una computazione deterministica che termina in k passi non può utilizzare più di k celle del nastro.

2. $dtime(T,x) \leq dspace(T,x)|Q|(|\Sigma|+1)^{dspace(T,x)}$

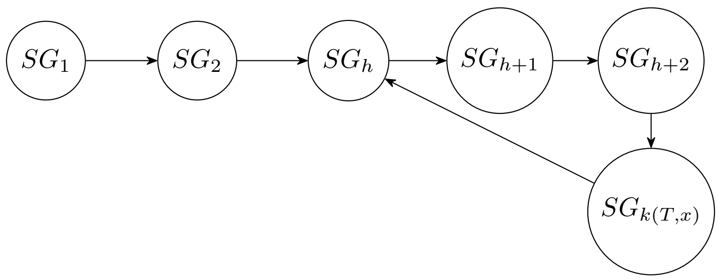
   (a) $dspace(T,x)|Q|(|\Sigma|+1)^{dspace(T,x)}$: è il numero di stati globali possibili di T nel caso in cui non più di $dspace(T,x)$ celle del nastro vengano utilizzate dalla computazione $T(x)$.

   (b) $(|\Sigma|+1)^{dspace(T,x)}$: sono tutte le possibili parole di $dspace(T,x)$ simboli di $\Sigma \cup \{\square\}$, ossia tutte le possibili configurazioni delle $dspace(T,x)$ celle utilizzate.

   Siano, dunque, T una macchina deterministica e $x \in \Sigma^\star$ tali che $T(x)$ termina in k passi utilizzando $dspace(T,x)$ celle del nastro. Poiché $T(x)$ termina in k passi, essa è una successione di stati globali

   $$SG_0(x), SG_2(x), \ldots, SG_k(x)$$

   tali che $SG_0(x)$ è lo stato globale iniziale e per ogni $0 \leq i \leq k-1$ esiste una transizione $SG_i(x) \rightarrow SG_{i+1}(x)$, e $SG_k(x)$ è lo stato globale finale.
   Sia $k(T,x) = dspace(T,x)|Q|(|\Sigma|+1)^{dspace(T,x)}$. Se $T(x)$ durasse più di $k(T,x)$ passi (senza mai uscire dalle $dspace(T,x)$ celle), allora sarebbe una successione di stati globali contenente almeno due volte uno stesso stato globale, SG_h.



   Ma T è deterministica, allora, a partire da SG_h è possibile eseguire un'unica quintupla (verso SG_{h+1}) ed essa viene eseguita tutte le volte in cui $T(x)$ si trova in SG_h. Quindi, entrambe le volte, avviene una transizione verso lo stesso stao globale SG_{h+1}, in questo modo $T(x)$ va in loop e non termina che è contro l'ipotesi che termina.

4.2 Teorema a pag. 4

Sia $f : \mathbb{N} \to \mathbb{N}$ una funzione totale e calcolabile.

Se $L \subseteq \Sigma^\star$ è accettato da una macchina di Turing non deterministica NT tale che, per ogni $x \in L$, $ntime(NT, x) \leq f(|x|)$ allora L è decidibile.

Se $L \subseteq \Sigma^\star$ è accettato da una macchina di Turing non deterministica NT tale che, per ogni $x \in L$, $nspace(NT, x) \leq f(|x|)$ allora L è decidibile.

Dimostrazione: Poiché f è totale e calcolabile, esiste una macchina di Turing T_f trasduttore tale che, per ogni $n \in \mathbb{N}$, $T_f(n)$ termina con il valore $f(n)$ scritto sul nastro di output. Assumiamo che l'input e l'output siano codificati in unario. Sia $L \subseteq \Sigma^\star$ un linguaggio accettato da una macchina di Turing NT tale che $\forall x \in L,\ ntime(NT, x) \leq f(|x|)$. Deriviamo ora da NT e T_f una nuova macchina non deterministica $NT^{'}$ a tre nastri.

N_1: viene scritto in unario l'input $x \in \Sigma^\star$.

N_2: viene scritta la lunghezza di x in unario.

N_3: viene utitlizzato come clock, ovvero viene scritto $f(|x|)$.

La computazione $NT^{'}$ consiste di 3 fasi:

FASE 1: $NT^{'}(x)$ scrive $|x|$ su N_2 in unario. Una volta letto $\square$ su N_1, le testine di N_1 e N_2 vengono riposizionate sul carattere più a sinistra.

FASE 2: Simula $T_f(|x|)$, usando N_2 come nastro di input e N_3 come nastro di output. Essa termina scrivendo il valore di $f(|x|)$ su N_3 e riavvolge la testina.

FASE 3: Simula i primi $f(|x|)$ passi della computazione, utilizzando N_1 come input e nastro di lavoro e N_3 come clock, ovvero come contatore del numero di istruzioni eseguite. Fino a quando viene letto 1 su N_3 viene eseguita un'istruzione di $NT(x)$ e la testina di N_3 viene spostata a destra. Se $NT(x)$ raggiunge q_A o q_R, $NT^{'}(x)$ termina nel medesimo stato. Se viene letto $\square$ su N_3, $NT^{'}(x)$ termina in q_R.

Poiché f è calcolabile e totale e poiché la simulazione della computazione $NT(x)$ nella terza fase viene forzatamente terminata, se non ha terminato entro $f(|x|)$ passi, tutte le computazioni di $NT^{'}$ terminano.

- Se $x \in L$ allora poiché NT accetta x in $f(|x|)$ passi, nella terza fase termina in q_A prima che venga letto $\square$.
- Se $x \notin L$ allora o $NT(x)$ termina in q_R durante la terza fase e di conseguenza anche $NT^{'}(x)$ termina in q_R, oppure viene letto $\square$, ovvero $NT(x)$ non ha accettato x in $f(|x|)$ passi e dunque $NT^{'}(x)$ rigetta.

Questo dimostra che $NT^{'}$ decide L e dunque che L è decidibile.

4.3 Teorema a pag. 10

Per ogni funzione totale calcolabile $f : \mathbb{N} \to \mathbb{N}$

$$DTIME[f(n)] \subseteq NTIME[f(n)] \ \wedge\ DSPACE[f(n)] \subseteq NSPACE[f(n)]$$

Dimostrazione: Una macchina di Turing deterministica è una particolare macchina di Turing non deterministica avente grado di non determinismo pari a 1, inoltre ogni parola decisa in k passi e anche accettata in k passi (ogni parola decisa in k celle e anche accettata in k celle).

4.4 Teorema a pag. 10

Per ogni funzione totale calcolabile $f : \mathbb{N} \to \mathbb{N}$

$$DTIME[f(n)] \subseteq DSPACE[f(n)] \ \wedge \ NTIME[f(n)] \subseteq NSPACE[f(n)]$$

Dimostrazione: Segue dal teorema:

$$dspace(T, x) \leq dtime(T, x) \ldots$$

Sia $L \subseteq \Sigma^\star$ tale che $L \in DTIME[f(n)]$. Allora esiste T che decide L e tale che $\forall x \in \Sigma^\star,\ dtime(T, x) \in O(f(|x|))$. Poiché $dspace(T, x) \leq dtime(T, x)$ e $dspace(T, x) \leq dtime(T, x) \in O(f(|x|))$, allora $dspace(T, x) \in O(f(|x|))$ e dunque $L \in DSPACE(f(|x|))$.
Analogamente per $NTIME[f(n)] \subseteq NSPACE[f(n)]$.

4.5 Teorema a pag. 11

Per ogni funzione totale calcolabile $f : \mathbb{N} \to \mathbb{N}$

$$DSPACE[f(n)] \subseteq DTIME[2^{O(f(n))}] \ \wedge \ NSPACE[f(n)] \subseteq NTIME[2^{O(f(n))}]$$

Dimostrazione: Segue dal teorema:

$$\ldots dtime(T, x) \leq dspace(T, x)|Q|(|\Sigma| + 1)^{dspace(T,x)}$$

Sia $L \subseteq \Sigma^\star$ tale che $L \in DSPACE[f(n)]$. Allora, esiste una machina di Turing T deterministica T che decide L e tale che, per ogni $x \in \Sigma^\star$, $dspace(T, x) \in O(f(|x|))$. Poiché:
Supponiamo che $\Sigma = \{0, 1\}$

$$\begin{aligned} dtime(T, x) &\leq dspace(T, x)|Q|(|\Sigma| + 1)^{dspace(T,x)} \\ &= dspace(T, x)|Q|3^{dspace(T,x)} \\ &= 2^{\log(dspace(T,x))}\ |Q|\ 2^{\log(3) dspace(T,x)} \\ &= |Q|2^{\log(dspace(T,x))\ +\ \log(3) dspace(T,x)} \\ &\leq |Q|2^{(1+\log(3)) dspace(T,x)} \end{aligned}$$

Allora $dtime(T, x) \in O(2^{O(f(|x|))})$ e dunque $L \in DTIME[2^{O(f(|x|))}]$

4.6 Teorema a pag. 11

Per ogni funzione totale calcolabile $f : \mathbb{N} \to \mathbb{N}$

$$DTIME[f(n)] = coDTIME[f(n)] \ \wedge \ DSPACE[f(n)] = coDSPACE[f(n)]$$

Dimostrazione: $\forall L \in DTIME[f(n)]$, esiste T che decide L e $dtime(T, x) \in O(f(|x|))$. Da T deriviamo T^c con input $x \in \Sigma^\star$ e $Q_f = \{q_A^c, q_R^c\}$ che decide L^c nel seguente modo:

FASE 1: Simula $T(x)$

FASE 2:
- Se $T(x) = q_A$, allora $T^c(x) = q_R$
- Se $T(x) = q_R$, allora $T^c(x) = q_A$

Dunque $L^c \in DTIME[f(n)]$.
Analogamente possiamo dimostrare che un qualsiasi linguaggio $L \in coDTIME[f(n)]$. Di conseguenza che $DTIME[f(n)] = coDTIME[f(n)]$. Dimostrazione analoga per $DSPACE[f(n)] = coDSPACE[f(n)]$.

4.7 Teorema a pag. 14

Sia $f : \mathbb{N} \rightarrow \mathbb{N}$ una funzione time-constructible (space-constructible.).

Allora, per ogni $L \in NTIME[f(n)]$, si ha che L è decidibile in tempo non deterministico in $O(f(n))$.

Allora, per ogni $L \in NSPACE[f(n)]$, si ha che L è decidibile in spazio non deterministico in $O(f(n))$.

Dimostrazione: Sia $f : \mathbb{N} \rightarrow \mathbb{N}$ una funzione time-constructible. Allora esiste una macchina di Turing di tipo trasduttore T_f che, avendo scritto sul nastro di input/lavoro $n \in \mathbb{N}$ in unario, in $O(f(n))$ passi scrive sul nastro di output il valore di $f(n)$ in unario. Sia $L \in NTIME[f(n)]$. Allora esiste una NT che accetta L e tale che, per ogni $x \in L$, $ntime(NT, x) \in O(f(|x|))$. Definiamo ora da T_f e NT, la macchina $NT^{'}$ con input $x \in L$ che opera nel seguente modo:

FASE 1: Scrive su N_2, $|x|$ in unario.

FASE 2: Simula $T_f(x)$ e scrive il risulato in unari su N_3.

FASE 3: Finché legge 1 su N_3 esegue una istruzione di $NT(x)$ su N_1.

- Se termina in q_A, allora $NT^{'}(x) = q_A$.
- Altrimenti sposta la testina a destra su N_3

FASE 4: Se legge $\square$ su N_3 allora rigetta.

- La fase 1 termina in $O(|x|)$ passi.
- la fase 2 termina in $O(f(|x|))$ passi, in quanto f è time-constructible.
- La fase 3 termina in $O(f(|x|))$ passi, in quanto $\forall x \in \Sigma^\star$, $ntime(NT^{'}, x) \in O(f(|x|))$.

Dunque, $NT^{'}(x)$ decide L, $\forall x \in \Sigma^\star$, e $ntime(NT^{'}, x) \in O(f(|x|))$.

4.8 Teorema a pag. 14

Per ogni funzione $f : \mathbb{N} \rightarrow \mathbb{N}$ time-constructible,

$$NTIME[f(n)] \subseteq DTIME[2^{O(f(n)}]$$

Dimostrazione: Sia $L \in \Sigma^\star$ tale che $L \in NTIME[f(n)]$, allora esistono una macchina di Turing NT che accetta L e una costante h tali che $\forall x \in L$, $ntime(NT, x) \leq hf(|x|)$. Poiché f è time-constructible, esiste una macchina di Turing deterministica T_f con inpun la rappresentazione in unario di $n \in \mathbb{N}$, che calcola il valore di $f(n)$ in unario in tempo $O(f(n))$. Indichiamo con k il grado di non determinismo di NT e definiamo T deterministica che simuli NT con input $x \in \Sigma^\star$ che opera nel seguente modo:

FASE 1: Scrive su N_2, $|x|$ in unario.

FASE 2: Simula $T_f(|x|)$ e scrive su N_3 l'output $hf(|x|)$ in unario.

FASE 3: Per ogni computazione deterministica $\alpha(x)$ in $NT(x)$:

- Finché legge 1 su N_3 esegue un'istruzione lungo $\alpha(x)$.
- Se $\alpha(x) = q_A$ allora $T(x) = q_A$.
- Altrimenti sposta la testina a destra su N_3.
- Se legge $\square$ si sposta sul primo 1 a sinistra e passa alla prossima computazione deterministica $\alpha(x)$ se esiste.

FASE 4: Rigetta

- La fase 1 termina in $O(f(|x|))$ passi.
- La fase 2 termina in $O(hf(|x|)) = O(f(|x|))$ passi.
- La fase 3 termina in $O(f(|x|)k^{hf(x)})$ poiché esegue ogni computazione deterministica di $NT(x)$

Dunque,

$$dtime(T, x) \in O(f(|x|)k^{hf(x)}) \subseteq O(2^{O(f(|x|))})$$

4.9 Teorema a pag. 20

Siano C e $C^{'}$ due classi di complessità tali che $C^{'} \subseteq C$. Se $C^{'}$ è chiusa rispetto ad una π-riduzione allora, per ogni linguaggio L che sia C-completo rispetto a tale π-riduzione, $L \in C^{'}$ se e solo se $C = C^{'}$.

Dimostrazione:

$\Leftarrow$) Se $C = C^{'}$ allora $L \in C^{'}$.

($\Rightarrow$ Sia $L \in C^{'}$. Poiché L è C-completo rispetto alla π-riducibilità, allora per ogni $L_0 \in C$, $L_0 \leq_\pi L$. Poiché $C^{'}$ è chiusa rispetto alla π-riduzione, ovvero per ogni altro linguaggio $L^{'}$, $L^{'} \leq_\pi L$, allora $L^{'} \in C^{'}$, dunque per ogni linguaggio $L_0 \in C$, $L_0 \leq_\pi L$, allora $L_0 \in C^{'}$. Quindi $C = C^{'}$.

4.10 Teorema a pag. 21

La classe **P** è chiusa rispetto alla riducibilità polinomiale.

Dimostrazione: Sia $L \in \mathbf{P}$, allora esistono una macchina di Turing T deterministica e $k \in \mathbb{N}$ tale che T decide L e per ogni $x \in \Sigma^\star$, $dtime(T, x) \in O(|x|^k)$.
Sia $L^{'}$: $L^{'} \leq_p L$, allora esiste una funzione $f : \Sigma_1^\star \to \Sigma_2^\star$ in **FP** che riduce $L^{'}$ a L, con T_f trasduttore tale che per ogni $x \in \Sigma_1^\star$, $T_f(x) \in L \Leftrightarrow x \in L^{'} \wedge dtime(T_f, x) \in O(|x|^c)$.
Da T e T_f definiamo $T^{'}$ con input x che opera nel seguente modo:

FASE 1: Simula $T_f(x)$ e scrive l'output y su N_2.

FASE 2: Simula $T(y)$:

- Se termina in q_A allora $T^{'}$ accetta.
- Se termina in q_R allora $T^{'}$ rigetta.

Dato che f è una riduzione da $L^{'}$ a L, quindi $f(x) \in L \Leftrightarrow x \in L^{'}$, quindi $T^{'}$ termina per ogni input $x \in \Sigma^\star$ e accetta $\Leftrightarrow T(f(x))$ accetta, ossia $\Leftrightarrow$ $f(x) \in L$.
Resta da mostrare che $T^{'}(x)$ opera in tempo polinomiale in $|x|$. La simulazione di $T_f(x)$ richiede $dtime(T_f, x) \leq |x|^c$ e la simulazione di $T(f(x))$ richiede $dtime(T, f(x)) \leq |f(x)|^k$.

$$\begin{aligned} dtime(T^{'}, x) &\leq |x|^c + f(|x|)^{k4} \\ &\leq |x|^c + |x|^{ck} \\ &\Rightarrow dtime(T^{'}, x) \in O(|x|^{ck}) \end{aligned}$$

Poiché c e k sono costanti, allora risulta che **P** è chiusa rispetto la riducibilità polinomiale dato che $L^{'} \in \mathbf{P}$.

4.11 Teorema a pag. 21

Le classi **NP, PSPACE, EXPTIME, NEXPTIME**, sono chiuse rispetto alla riducibilità polinomiale.

Dimostrazione:

NP

PSPACE

EXPTIME

NEXPTIME

4.12 Corollario a pag. 21

Se $\mathbf{P} \neq \mathbf{NP}$ allora, per ogni linguaggio **NP**-completo L, $L \notin \mathbf{P}$.

Dimostrazione: Supponiamo che L sia un linguaggio **NP**-completo e che $L \in \mathbf{P}$. Poiché L è **NP**-completo, per ogni linguaggio $L^{'} \in \mathbf{NP}$, $L^{'} \leq L$, ma se $L \in \mathbf{P}$, poiché **P** è chiusa rispetto alla riduzione $\leq$, questo implica che, per ogni $L^{'} \in \mathbf{NP}$, $L^{'} \in \mathbf{P}$. Ossia $\mathbf{P} = \mathbf{NP}$, contraddicendo l'ipotesi.

4.13 Teorema a pag. 23

Se $\mathbf{NP} \neq \mathbf{coNP}$, allora $\mathbf{P} \neq \mathbf{NP}$.

Dimostrazione $(A \to B \Leftrightarrow \neg B \to \neg A)$**:** Se $\mathbf{P} = \mathbf{NP}$, allora $\mathbf{NP} = \mathbf{coP}$ poiché $\mathbf{P} = \mathbf{coP}$. Ma se $\mathbf{P} = \mathbf{NP}$ $\wedge$ $\mathbf{P} = \mathbf{coP}$, allora $\mathbf{coP} = \mathbf{coNP}$. Dunque:

$$\underline{\mathbf{NP}} = \mathbf{P} = \mathbf{coP} = \underline{\mathbf{coNP}}$$

4.14 Teorema a pag. 23

La classe **coNP** è chiusa rispetto alla riducibilità polinomiale.

Dimostrazione: Poiché **NP** è chiusa rispetto alla riducibilità polinomiale

Per ogni $L_2 \in \mathbf{NP}$ e per ogni L_1, se $L_1 \leq L_2$, allora $L_1 \in \mathbf{NP}$.

$\Rightarrow$ Per ogni $L_2^c \in \mathbf{coNP}$ e per ogni L_1^c, se $L_1^c \leq L_2^c$, allora $L_1^c \in \mathbf{coNP}$

4.15 Teorema a pag. 23

Un linguaggio L è **NP**-completo se e soltanto se L^c è **coNP**-completo.

Dimostrazione:

($\Rightarrow$ Sia L un linguaggio **NP**-completo, allora per definizione di completezza

1. $L \in \mathbf{NP}$ allora $L^c \in \mathbf{coNP}$.
2. $\forall L_0 \in \mathbf{NP},\ L_0 \leq L$.

Sia L_1 un qualsiasi linguaggio in **coNP**, allora $L_1^c \in \mathbf{NP}$. Poiché L è **NP**-completo, $L_1^c \leq L$.

$$\begin{aligned}
&\Rightarrow \mathit{Esiste}\ f \in \mathbf{FP} : \forall x \in \Sigma^\star [x \in L_1^c \Leftrightarrow f(x) \in L] \\
&\Rightarrow \mathit{Esiste}\ f \in \mathbf{FP} : \forall x \in \Sigma^\star [x \notin L_1^c \Leftrightarrow f(x) \notin L] \\
&\Rightarrow \mathit{Esiste}\ f \in \mathbf{FP} : \forall x \in \Sigma^\star [x \in L_1 \Leftrightarrow f(x) \in L^c]
\end{aligned}$$

In conclusione, $\forall L_1 \in \mathbf{coNP} :\ L_1 \leq L^c$, e quindi L^c è **coNPC**.

$\Leftarrow$) Sia L^c un linguaggio **coNP**-completo, allora per definizione di completezza

1. $L^c \in \mathbf{coNP}$ allora $L \in \mathbf{NP}$.
2. $\forall L_0^c \in \mathbf{coNP},\ L_0^c \leq L^c$.

Sia L_1 un qualsiasi linguaggio in **NP**, allora $L_1^c \in \mathbf{coNP}$. Poiché L^c è **coNP**-completo, $L_1^c \leq L^c$.

$$\begin{aligned}
&\Rightarrow \mathit{Esiste}\ f \in \mathbf{FP} : \forall x \in \Sigma^\star [x \in L_1^c \Leftrightarrow f(x) \in L^c] \\
&\Rightarrow \mathit{Esiste}\ f \in \mathbf{FP} : \forall x \in \Sigma^\star [x \notin L_1^c \Leftrightarrow f(x) \notin L^c] \\
&\Rightarrow \mathit{Esiste}\ f \in \mathbf{FP} : \forall x \in \Sigma^\star [x \in L_1 \Leftrightarrow f(x) \in L]
\end{aligned}$$

In conclusione, $\forall L_1 \in \mathbf{NP} :\ L_1 \leq L$, e quindi L è **NPC**.

4.16 Teorema a pag. 24

Se esiste un linguaggio L **NP**-completo tale che $L \in \mathbf{coNP}$, allora $\mathbf{NP} = \mathbf{coNP}$.

Dimostrazione: L è **NP**-completo, allora per definizione di completezza

1. $L \in \mathbf{NP}$.
2. $\forall L_1 \in \mathbf{NP},\ L_1 \leq L$.

$\subseteq$ Se $L \in \mathbf{coNP}$ allora $\forall L_1 \in \mathbf{NP},\ L_1 \leq L$, ma **coNP** è chiusa ripsetto alla riducibilità polinomiale ovvero $[L_2 \in \mathbf{coNP},\ L_1 \leq L_2,\ \Rightarrow\ L_1 \in \mathbf{coNP}]$, allora, per ogni $L_1 \in \mathbf{NP}$ si ha che $L_1 \leq L$, e $L \in \mathbf{coNP}$. Dunque, per la chisura di **coNP**, $L_1 \in \mathbf{coNP}$, quindi $\mathbf{NP} \subseteq \mathbf{coNP}$.

$\supseteq$ Poiché $L \in \mathbf{coNP}$, allora $L^c \in \mathbf{NP}$, ma poiché L è **NP**-completo, allora L^c è **coNP**-completo, quindi $\forall L^{'} \in \mathbf{coNP},\ L^{'} \leq L^c$. Ma **NP** è chiusa rispetto alla riducibilità polinomiale, ovvero $[L_2 \in \mathbf{NP},\ L_1 \leq L_2,\ \Rightarrow\ L_1 \in \mathbf{NP}]$, allora, per ogni $L^{'} \in \mathbf{coNP},\ L^{'} \leq L^c$ e $L^c \in \mathbf{NP}$. Dunque, per la chiusura di **NP**, $L^{'} \in \mathbf{NP}$, quindi $\mathbf{coNP} \subseteq \mathbf{NP}$.

In conclusione, se L è **NP**-completo $\wedge$ $L \in \mathbf{coNP}$, allora $\mathbf{NP} = \mathbf{coNP}$.

5 Teoremi Dispensa 9

5.1 Teorema a pag. 8

Un linguaggio $L \subseteq \Sigma^\star$ è in **NP** se è soltanto se[5] esistono una macchina di Turing deterministica T che opera in tempo polinomiale e due costanti $k, h \in \mathbb{N}$ tali che, per ogni
$x \in \Sigma^\star,\ x \in L \Leftrightarrow \exists y_x \in \{0,1\}^\star :\ |y_x| \leq |x|^k \wedge T(x, y_x) = q_A \wedge dtime(T, x, y_x) \leq |x|^h$

Dimostrazione:

($\Rightarrow$ Sia $L \subseteq \Sigma^\star$ un linguaggio in **NP**, allora esistono una macchina di Turing non deterministica NT e un intero $h \in \mathbb{N}$ tale che NT accetta L e, per ogni, $x \in L,\ ntime(NT, x) \leq |x|^h$.

Questo significa che $\forall x \in L$ esiste una sequenza di quintuple che eseguite a partire dallo stato globale inziale SG_0 porta ad uno stato globale di accettazione.

Allora, $p_i = \langle q_{i1},\ s_{i1},\ s_{i2},\ q_{i2},\ m_i \rangle$ è la $i-esima$ quintupla della sequenza $y(x)$, definita come segue:

$$y(x) =\ q_{11},\ s_{11},\ s_{12},\ q_{12},\ m_1\ -\ q_{21},\ s_{21},\ s_{22},\ q_{22},\ m_2\ -\ \dots\ q_{(n^k)1},\ s_{(n^k)1},\ s_{(n^k)2},\ q_{(n^k)2},\ m_{(n^k)}$$

è la sequenza di quintuple che rappresentano una computazione deterministica accettante.[6]

Definiamo ora una macchina deterministica $\overline{T}$ che ha il ruolo di verificare la sequenza di quintuple y_x scelta dal genio. Dunque $\overline{T}$ è detto **verificatore** ed opera nel seguente modo:

1. $\overline{T}$ verifica che y sia nella forma descritta sopra, se non è così, rigetta.
2. $\overline{T}$ verifica che, per ogni $1 \leq i \leq n^k,\ \langle q_{i1},\ s_{i1},\ s_{i2},\ q_{i2},\ m_i \rangle \in P$, se non è così, allora rigetta.
3. $\overline{T}$ verifica che $q_{11} = q_0$ e $q_{(n^k)2} = q_A$, se così non è, rigetta.
4. $\overline{T}$ verifica che, per ogni $2 \leq i \leq n^k,\ q_{i1} = q_{(i-1)2}$, se così non è, rigetta.
5. $\overline{T}$ simula la computazione di $NT(x)$ descritta da y. Verifica se ogni quintupla può eseguita, se sì la esegue, altrimenti rigetta.
6. $\overline{T}$ accetta.

Dunque, se $x \in L$, allora $y(x)$ è la codifica di $NT(x)$ accettante che è costituita da al più $|x|^k$ passi. Dunque, se $x \in L$, allora $|y(x) \in O(|x|^k)|$ e quindi $\overline{T}$ opera in tempo polinomiale in $|x|$.

Se $x \in L,\ y_x$ prende il nome di **certificato** per x. Dunque $x \in L \Leftrightarrow\ \exists y(x) \in [\Sigma \cup Q \cup \{-, s, f, d\}^\star]$ tale che $\overline{T}(x, y_x)$ accetta.

$\Leftarrow$) Sia $L \subseteq \Sigma^\star$ un linguaggio per il quale esistono una macchina di Turing deterministica T che opera in tempo polinomiale e una costante $k \in \mathbb{N}$ tali che, $\forall x \in \Sigma^\star,\ x \in L \Leftrightarrow \exists y_x \in \{0,1\}^\star$, tale che $|y_x| \leq |x|^k \wedge T(x, y_x)$ accetta.

Dobbiamo dimostrare che $\exists NT$ e $a \in \mathbb{N}$ tale che, $\forall x \in L,\ NT(x)$ accetta e $ntime(NT, x) \in O(|x|^a)$. NT opera come segue:

FASE 1: NT sceglie non determisticamente una parola binaria y di lunghezza $|y| \leq |x|^k$.

FASE 2: NT invoca $T(x, y)$ e, se $T(x, y)$ accetta entro $O(|x|^h)$ passi allora accetta.

NOTA: $|x|^k$ è time-constructible, ovvero essite una macchina T_f trasduttore che calcola $|x|^k$ e $dtime(T_f, x) \leq |x|^k$.

- Se $x \in L$ allora $\exists y_x \in \{0,1\}^\star :\ |y_x| \leq |x|^k \wedge T(x, y_x) = q_A$, allora esiste una sequenza di scelte che genera y_x, allora nella $FASE\ 2$, $T(x, y_x)$ accetta entro $|x|^h$ passi, allora $NT(x)$ corrisponde alla sequenza di scelte che ha generato y_x e accetta. Questo dimostra che se $x \in L$, allora $NT(x)$ accetta.
- Se $x \notin L$ allora non esiste alcuna $y_x \in \{0,1\}$, non esiste alcuna $y(x)$ tale che $|y(x)| \leq |x|^k \wedge T(x, y_x) = q_A$. Dunque, qualunque sia la sequenza di scelte per generare y, $T(x, y)$ non accetta. Quindi se $x \notin L$, allora $NT(x)$ non accetta.

Questo dimostra che $L \in$ **NP**

[5] Questo è da dimostrare

[6] Adesso il Genio non ci da più una quintupla per volta, ma una sequenza di quintuple che però, devono essere verificate.

```
procedure NT-FASE-1( x ∈ Σ* )
    B ← T_f(|x|)
    i ← 1
    while i ≤ B do begin
        scegli y[i] nell'insieme {0, 1}
        i ← i + 1
    end
    y ← y[1] ⊕ y[2] ⊕ ··· ⊕ y[B]
    return y
procedure NT-FASE-2( x ∈ Σ* )
    y_x ← NT − FASE − 1(x)                ▷ O(|x|^k)
    q ← T(x, y_x)                         ▷ O(|x|^h)
    return q
```

5.2 Teorema a pag. 14

Sia $\Gamma_1 \in \mathbf{NP} \land \exists\ \Gamma_2 \in \mathbf{NPC} \land \Gamma_2 \leq \Gamma_1$, allora Γ_1 è **NPC**.

Dimostrazione:

Sia Γ_2 un problema **NPC** tale che $\Gamma_2 \leq \Gamma_1$, allora $\exists\ f_{21} :\ I_{\Gamma_2} \to I_{\Gamma_1}$ tale che $f_{21} \in \mathbf{FP}$ e per ogni $x \in I_{\Gamma_2},\ [x \in \Gamma_1 \Leftrightarrow f_{21}(x) \in \Gamma_1]$, ovvero

$$\exists\ T_{21}, k :\ \forall x \in I_{\Gamma_2},\ [x \in \Gamma_1 \Leftrightarrow T_{21}(x) \in \Gamma_1 \land\ dtime(T_{21}, x) \leq |x|^k]$$

Poiché Γ_2 è **NPC**, per ogni problema $\Gamma_3 \in \mathbf{NP}$ si ha che $\Gamma_3 \leq \Gamma_2$, e dunque esiste una funzione $f_{32} :\ \Gamma_3 \Rightarrow \Gamma_2$ tale che $f_{32} \in \mathbf{FP}$ e per ogni $z \in I_{\Gamma_3},\ [z \in \Gamma_3 \Leftrightarrow f_{32}(z) \in \Gamma_2]$, ovvero

$$\exists T_{32}, h : \forall z \in I_{\Gamma_3},\ [z \in \Gamma_2 \Leftrightarrow T_{32}(z) \in \Gamma_2 \land\ dtime(T_{32}, z) \leq |x|^h]$$

Da T_{21} e T_{32} definiamo T_{31} con input $z \in I_{\Gamma_3}$ che opera nel seguente modo:

FASE 1: Simula $T_{32}(z) = x$

FASE 2: Simula $T_{21}(x) = y$

FASE 3: Scrivi sul nastro di output y

Sia $z \in I_{\Gamma_3}$, allora $z \in \Gamma_3 \Leftrightarrow f_{32}(z) \in \Gamma_2$ e inoltre, $f_{32}(z) \in \Gamma_2 \Leftrightarrow f_{21}(f_{32}(z)) \in \Gamma 1$. Se indichiamo con f_{31} la composizione delle funzionie f_{32} e f_{21}, questo dimostra che f_{31} è una riduzione da Γ_3 a Γ_1.
Resta da dimostrare che la macchina T_{32} opera in tempo polinomiale.

$$\forall z \in I_{\Gamma_3} :\ [dtime(T_{31}, z) = dtime(T_{32}, z) + dtime(T_{21}, x) \leq |z|^h + |x|^k \leq |z|^{hk}]$$

Questo dimostra che Γ_1 è **NPC**.