

# Teoremi Fondamenti

## Teoremi Dispensa 3

### Teorema

Un  $L \subseteq \Sigma^*$  un linguaggio è decidibile  $\iff L$  è accettabile e  $L^c$  è accettabile

dim

( $\implies$ )

Sia  $L$  un linguaggio decidibile  $\implies \exists T : \forall x \in \Sigma^* [o_T(x) = q_A \iff x \in L \wedge o_T(x) = q_R \iff x \in L^c]$

Osserviamo immediatamente che siccome  $T$  decide  $L$  lo accetta anche dunque  $L$  è accettabile.

Per dimostrare che anche  $L^c$  lo è, partendo da  $T$  deriviamo una nuova macchina di Turing  $T'$  che accetta  $L^c$ . Essa opera sull'alfabeto  $\Sigma$ . L'insieme dei suoi stati coincide con quello della macchina  $T$ .  $Q_{T'} = Q_T \cup \{q_{AT'}, q_{RT'}\}$  con  $q_{AT'}, q_{RT'} \notin Q_T$ .  $q_{AT'}, q_{RT'}$  sono rispettivamente gli stati di accettazione e rigetto di  $T'$ . L'insieme delle quintuple di  $T$  coincide con quello di  $T'$  tranne che per l'aggiunta di due quintuple nuove:

$\langle q_A, u, u, q_{RT'}, f \rangle, \langle q_R, u, u, q_{AT'}, f \rangle \forall x \in \Sigma \cup \{\square\}$ .

La computazione  $T'(x)$  opera nel seguente modo:

Viene simulata la computazione  $T'(x)$ . Se  $o_T(x) = q_A$  allora viene eseguita la quintupla che porta la macchina  $T'$  nello stato di rigetto  $q_{RT'}$ .

Se  $o_T(x) = q_R$  allora viene eseguita la quintupla che porta la macchina  $T'$  nello stato di accettazione  $q_{AT'}$ .

Dunque  $T'$  accetta  $L^c$ .

( $\impliedby$ )

Sia  $L$  accettabile  $\implies \exists T_1 : \forall x \in \Sigma^* [o_{T_1}(x) = q_A \iff x \in L \wedge o_{T_1}(x) \neq q_A \iff x \in L^c]$

Sia  $L^c$  accettabile  $\implies \exists T_2 : \forall x \in \Sigma^* [o_{T_2}(x) = q_A \iff x \in L^c \wedge o_{T_2}(x) \neq q_A \iff x \in L]$

Dobbiamo dimostrare che  $L$  è decidibile, per farlo deriviamo da  $T_1$  e  $T_2$  una nuova macchina di Turing  $T$  riconoscitore che decide  $L$ .

Essa opera su alfabeto  $\Sigma$ , a due nastri su ognuno dei quali è scritto l'input  $x \in \Sigma^*$ . La computazione  $T(x)$  avviene simulando le computazioni  $T_1(x)$  e  $T_2(x)$  alternando le singole istruzioni delle due computazioni (poiché non abbiamo garanzie che una delle tue computazioni termini) nel seguente modo:

1. Viene eseguita un'istruzione di  $T_1$  utilizzando il primo nastro, se essa termina nello stato di accettazione  $T$  termina nel suo stato di accettazione. Altrimenti viene eseguita la fase 2.
2. Viene eseguita un'istruzione di  $T_2$  utilizzando il secondo nastro, se essa termina nello stato di accettazione  $T$  termina nel suo stato di rigetto. Altrimenti viene eseguita la fase 1.

Osserviamo come se  $x \in L$  allora prima o poi il passo 1 porterà la macchina  $T$  nello stato di accettazione. Se  $x \in L^c$  allora prima o poi il passo 2 porterà la macchina  $T$  nel suo stato di rigetto. Quindi  $T$  decide  $L$ .

### Teorema

Un linguaggio  $L \subseteq \Sigma^*$  è decidibile  $\iff$  la funzione  $\chi_L$  è calcolabile

dim

( $\implies$ )

Sia  $L$  un linguaggio decidibile  $\implies \exists T : \forall x \in \Sigma^* [o_T(x) = q_A \iff x \in L \wedge o_T(x) = q_R \iff x \notin L]$

Dobbiamo dimostrare che se  $L$  è decidibile allora  $\chi_L$  è calcolabile ovvero esiste una macchina di Turing  $T'$  di tipo trasduttore che con input  $x \in \Sigma^*$  calcola il valore  $\chi_L(x)$  e lo scrive sul nastro di output se e soltanto se  $\chi_L(x)$  è definita (in questo caso essendo  $\chi_L$  totale è definita sempre).

Da  $T$  deriviamo una macchina di Turing trasduttore ad un nastro (oltre quello di output) dove viene scritto l'input  $x \in \Sigma^*$ . La computazione  $T'(x)$  avviene nel seguente modo:

1. Viene simulata la computazione  $T(x)$  utilizzando il primo nastro come nastro di lavoro.
2. Nel caso la computazione  $T(x)$  termini nel suo stato di accettazione, sul nastro di output viene scritto 1. Nel caso la computazione  $T(x)$  termini nel suo stato di rigetto, sul nastro di output viene scritto 0.

Poiché  $L$  è decidibile, il primo passo termina sempre. Se  $x \in L$  allora  $T(x)$  termina nello stato di accettazione e nel passo 2 viene scritto 1 sul nastro di output. Se  $x \notin L$  allora  $T(x)$  termina nello stato di rigetto e nel passo 2 viene scritto 0 sul nastro di output. Dunque  $\chi_L$  è calcolabile.

( $\Leftarrow$ )

Sia  $\chi_L$  calcolabile allora esiste una macchina trasduttore  $T'$  che su input  $x \in \Sigma^*$  calcola il valore  $\chi_L(x)$  e lo scrive sul nastro di output.

Da  $T'$  derivo una macchina di Turing riconoscitore  $T$  che decide  $L$ :

Essa utilizza due nastri, sul primo viene scritto l'input  $x \in \Sigma^*$ . La computazione  $T(x)$  avviene nel seguente modo:

1. Viene simulata la computazione  $T'(x)$  utilizzando il primo nastro come nastro di lavoro e viene scritto il valore  $\chi_L(x)$  sul nastro di output.
2. Se sul nastro di output viene scritto 1 la macchina  $T$  termina nello stato di accettazione. Se sul nastro di output viene scritto 0 la macchina  $T$  termina nello stato di rigetto.

Siccome  $\chi_L$  è totale, il passo 1 termina sempre. Se  $\chi_L(x) = 1$  allora la computazione  $T(x)$  termina nello stato di accettazione e dunque  $x \in L$ . Se  $\chi_L(x) = 0$  allora la computazione  $T(x)$  termina nello stato di rigetto e dunque  $x \notin L$ . Dunque  $L$  è decidibile

## Teorema

Se la funzione  $f: \Sigma^* \rightarrow \Sigma_1^*$  è totale è calcolabile allora il linguaggio  $L_f = \{ \langle x, y \rangle : x \in \Sigma^* \wedge y = f(x) \} \subseteq \Sigma^* \times \Sigma_1^*$  è decidibile

dim

Poiché  $f$  è totale e calcolabile allora esiste una macchina di Turing  $T_f$  ad un nastro (oltre quello di output), che su con input  $x \in \Sigma^*$  calcola il valore  $f(x)$  e lo scrive sul nastro di output. Da  $T_f$  deriviamo una nuova macchina di Turing  $T$  riconoscitore a 2 nastri. Sul primo di questi verrà scritto l'input  $\langle x, y \rangle$  con  $x \in \Sigma^*$  e  $y \in \Sigma_1^*$ , opera nella seguente maniera:

1. Sul primo nastro è scritto l'input  $\langle x, y \rangle$
2. Viene simulata la computazione  $T_f(x)$  calcolando il valore  $z = f(x)$  e scrivendo  $z$  sul secondo nastro
3. Esegue un confronto fra  $z$  e  $y$ . Se  $z = y$  allora la computazione  $T(\langle x, y \rangle)$  termina nello stato di accettazione, altrimenti nello stato di rigetto.

Siccome  $f$  è una funzione totale il passo 2. termina sempre per ogni input  $x$ . Se al passo 2. viene scritto sul secondo nastro il valore  $y = f(x)$  al passo 3. la computazione  $T(\langle x, y \rangle)$  termina nello stato di accettazione. Se invece  $f(x) = z \neq y$  allora il passo 2. termina scrivendo  $z$  sul secondo nastro e al passo 3. la computazione  $T(\langle x, y \rangle)$  termina nello stato di rigetto.  $L$  è decidibile

## Teorema

Sia  $f: \Sigma^* \rightarrow \Sigma_1^*$  una funzione. Se il linguaggio  $L_f \subseteq \Sigma^* \times \Sigma_1^*$  è decidibile allora  $f$  è calcolabile.

dim

Poiché  $L_f \subseteq \Sigma^* \times \Sigma_1^*$  è decidibile, esiste una macchina di Turing di tipo riconoscitore  $T$ , con stato di accettazione  $q_A$  e stato di rigetto  $q_R$ , tale che, per ogni  $x \in \Sigma^*$  e per ogni  $y \in \Sigma_1^*$ :

$$o_T(x, y) = \begin{cases} q_A & \text{se } y = f(x) \\ q_R & \text{altrimenti} \end{cases}$$

Senza perdita di generalità, supponiamo che  $T$  utilizzi un unico nastro. A partire da  $T$ , definiamo una macchina di Turing di tipo trasduttore  $T_f$  a 4 nastri, che, con input  $x \in \Sigma^*$  sul primo nastro, opera nella maniera seguente:

1. Scrive il valore  $i = 0$  sul primo nastro
2. Enumera tutte le stringhe  $y \in \Sigma_1^*$  la cui lunghezza è pari al valore scritto sul primo nastro, simulando per ciascuna di esse la computazione  $T(x, y)$ ; in altri termini, opera come segue:
  - 2.1. sia  $y$  la prima stringa di lunghezza  $i$  non ancora enumerata; allora, scrive  $y$  sul secondo nastro;
  - 2.2 sul terzo nastro, esegue la computazione  $T(x, y)$ ;
  - 2.3 se  $T(x, y)$  termina nello stato  $q_A$  allora scrive sul nastro di output la stringa  $y$  e termina, altrimenti, eventualmente incrementando il valore  $i$  scritto sul primo nastro se  $y$  era l'ultima stringa di lunghezza  $i$ , torna al passo 2.

Osserviamo innanzi tutto che, poichè  $L_f$  è decidibile, il passo 2.1 sopra termina per ogni input  $\langle x, y \rangle$ . Se  $x$  appartiene al dominio di  $f$ , allora esiste  $\bar{y} \in \Sigma_1^*$  tale che  $\bar{y} = f(x)$  e, quindi,  $\langle x, \bar{y} \rangle \in L_f$ . Allora, in questo caso, prima o poi (ma, comunque, in tempo finito) la stringa  $\bar{y}$  verrà scritta sul secondo nastro e la computazione  $T(x, y)$  terminerà nello stato di accettazione e, quindi, al passo 2.3  $T_f(x)$  scriverà  $\bar{y}$  sul nastro output terminerà. Questo dimostra che  $f$  è calcolabile.

Notiamo esplicitamente che, nella dimostrazione del Teorema, se  $x$  non appartiene al dominio di  $f$ , allora nessuna stringa  $y$  generata al passo 2.2 consente a  $T(x, y)$  di terminare nello stato di accettazione e, quindi, la computazione  $T_f(x)$  non termina. Pertanto, l'ipotesi di decidibilità di  $L_f$  non consente di affermare che  $f$  sia totale.

## Teoremi Dispensa 5

### Halting Problem

$L_H = \{ (i, x) \in \mathbb{N} \times \mathbb{N} : i \text{ è una codifica di una macchina di Turing } \wedge T(i) \text{ termina} \}$

### Teorema

$L_H$  è accettabile

dim

$L_H$  è accettabile  $\implies \exists T : \forall i, x \in \mathbb{N} \times \mathbb{N} [o_T(i, x) = q_A \iff (i, x) \in L_H]$

Mostriamo che  $T$  è una modifica della macchina universale  $U$  che lavora con 4 nastri. Sul primo nastro verrà scritta la codifica della macchina di Turing  $i$  mentre sul secondo nastro  $x \in \{0, 1\}^*$ . La macchina di Turing  $T$  inizia la sua computazione verificando che la codifica  $i$  scritta sul nastro 1 di input sia effettivamente la codifica di una macchina di Turing, così non fosse terminerebbe nello stato di rigetto. Poi  $T$  simula la computazione di  $U$  e se  $U$  termina (che sia nello stato di accettazione o di rigetto) allora  $T$  termina nello stato di accettazione. Quindi  $T(i, x)$  accetta le sole coppie  $(i, x)$  che appartengono a  $L_H$ . Dunque  $L_H$  è accettabile.

Siccome nel caso  $x \notin L_H^c$  non possiamo dire nulla,  $T$  non decide  $L_H^c$ .

### Teorema

$L_H$  non è decidibile

dim

Supponiamo che  $L_H$  sia decidibile, allora esiste una macchina di Turing  $T$  tale che

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

Da  $T$  possiamo, allora, semplicemente complementando gli stati di accettazione e di rigetto di  $T$ , derivare una nuova macchina  $T'$  che accetta tutte e sole le coppie  $(i, x) \in \mathbb{N} \times \mathbb{N} - L_H$ , ossia,

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

A partire da  $T'$  deriviamo, poi, una terza macchina  $T^*$  che, invece che su una coppia di interi, opera su un singolo input  $i \in \mathbb{N}$ . Inoltre,  $T^*(i)$  accetta se  $T'(i, i)$  accetta, mentre non termina se  $T'(i, i)$  rigetta. Questo è possibile apportando a  $T'$  le seguenti modifiche:

- sostituiamo lo stato  $q_R$  con un nuovo stato non finale  $q'_R$  in tutte le quintuple di  $T'$  che terminano nello stato  $q_R$ ;
- aggiungiamo alle quintuple di  $T'$  la quintupla  $\langle q'_R, y, y, q'_R, fm \rangle$ , per ogni  $y \in \{0, 1\}$ . Allora:

$$T^*(i) = \begin{cases} \text{non termina} & T'(i, i) \text{ rigetta} \\ q_A & \text{se } T'(i, i) \text{ accetta} \end{cases}$$

Poichè  $\mathcal{T}$  è un insieme numerabile e  $T^* \in \mathcal{T}$ , allora deve esistere  $k \in \mathbb{N}$  tale che  $T^* = T_k$ .

Ci chiediamo, ora: quale è l'esito della computazione  $T_k(k)$ ?

Se  $T_k(k) = T^*(k)$  accettasse, allora  $T'(k, k)$  dovrebbe accettare anch'essa. Ma se  $T'(k, k)$  accetta, allora  $(k, k) \in L_H$ , ossia,  $T_k(k)$  non termina. Allora,  $T^*(k)$  non può accettare e, dunque, necessariamente non termina.

Ma, se  $T^*(k)$  non termina, allora  $T'(k, k)$  rigetta e, quindi,  $(k, k) \in LH$ . Dunque, per definizione,  $T_k(k)$  termina. Quindi, entrambe le ipotesi,  $T_k(k)$  termina o  $T_k(k)$  non termina, portano ad una contraddizione. Allora, la macchina  $T^*$  non può esistere. Poiché  $T^*$  è ottenuta mediante semplici modifiche della macchina che dovrebbe decidere  $L_H$ , ne consegue che  $L_H$  non è decidibile.

## Teorema Dispensa 6

### Misure di complessità

$\forall T$  deterministica (riconoscitore o trasduttore) che opera su alfabeto  $\Sigma \forall x \in \Sigma^*$  definiamo le due seguenti funzioni:

1.  $dtime(T, x)$  = Numero di istruzioni eseguite dalla computazione  $T(x)$
2.  $dspace(T, x)$  = Numero di celle utilizzate dalla computazione  $T(x)$

#### dim (1.)

Per dimostrare che  $dtime(T, x)$  si una misura di complessità, dobbiamo dimostrare che essa rispetti gli assiomi di Blum, ovvero:

1. La funzione  $f$  (misura di complessità) è definita solo per le computazioni che terminano
2. La funzione  $f$  deve essere calcolabile.

Dimostriamo:

1. Per definizione,  $dtime(T, x)$  è definita per  $\forall T$  macchina di Turing deterministica e  $\forall x \in \Sigma^*$  se e soltanto se  $T(x)$  termina
2. Per dimostrare che  $dtime(T, x)$  è calcolabile, utilizziamo una  $U_{dtime}$  della macchina universale  $U$  a 5 nastri. Dove sul primo nastro vi sarà la codifica della macchina di Turing  $T$ , sul secondo nastro l'input  $x \in \Sigma^*$ , sul terzo nastro vi sarà scritto, ad ogni istante della computazione che simula  $T(x)$  lo stato attuale della macchina  $T$ . Sul quarto nastro invece vi sarà scritto lo stato di accettazione della macchina  $T$ . Una volta che la macchina  $U_{dtime}$  avrà eseguito un'istruzione di  $T$  e dopo essersi preparata ad eseguire l'istruzione successiva, scriverà un 1 sul nastro 5 e muove la testina su tale nastro di una posizione a destra. Una volta che la computazione  $U_{dtime}$  è terminata (se essa termina), sul quinto nastro vi sarà scritto in unario il numero di passi eseguiti dalla computazione  $T(x)$ , dunque  $dtime(T, x)$  è calcolabile.

### Teorema

Sia  $T$  una macchina di Turing deterministica, che opera su un alfabeto  $\Sigma$  (non contenente  $\square$ ), sia  $Q$  l'insieme dei suoi stati. Sia  $x \in \Sigma^*$  per cui  $T(x)$  termina, allora:

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

#### dim

Dimostriamo anzitutto che  $dspace(T, x) \leq dtime(T, x)$  poi dimostreremo che  $dtime(T, x) \leq dspace(T, x) |Q| (|\Sigma| + 1)^{dspace(T, x)}$ .

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

#### 1. $dspace(T, x) \leq dtime(T, x)$ :

Se la computazione  $T(x)$  utilizza  $dspace(T, x)$  celle, quelle dovrà almeno leggerle tutte. Per leggere una singola cella la computazione esegue un'istruzione.

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

Osserviamo come il valore di  $dspace(T, x) |Q| (|\Sigma| + 1)^{dspace(T, x)}$  corrisponda al numero di possibili stati globali di  $T$  nel caso la computazione  $T(x)$  non usi più di  $dspace(T, x)$  celle del nastro. Infatti poiché ogni cella può contenere un simbolo di  $\Sigma$  o  $\square$  il numero di possibili configurazioni del nastro è  $(|\Sigma| + 1)^{dspace(T, x)}$ , la testina può trovarsi in una delle  $dspace(T, x)$  celle, e la macchina può essere in uno dei qualsiasi  $|Q|$  stati interni. Per semplificare scritture successive assumo che  $k(T, x) = dspace(T, x) |Q| (|\Sigma| + 1)^{dspace(T, x)}$ .

Una computazione deterministica non è altro che una successione di stati globali, tale che si passa da uno stato globale ad un altro eseguendo una quintupla.

Dunque se  $T(x)$  durasse più di  $K(T, x)$  passi allora sarebbe una successione di stati globali contenente uno stato globale almeno due volte.

Ma la computazione è deterministica quindi a partire da uno stato globale  $SG_h$  è possibile eseguire un'unica quintupla, che verrebbe rieseguita ogni volta che la computazione  $T(x)$  si trovi in quello stato globale,  $T(x)$  sarebbe in loop contro l'ipotesi che termini.

## Teorema

Sia  $f : \mathbb{N} \rightarrow \mathbb{N}$  una funzione totale è calcolabile.

Sia  $L \subseteq \Sigma^*$  un linguaggio accettato da una macchina di Turing non deterministica  $NT$  tale che  $\forall x \in \Sigma^* [ntime(NT, x) \leq f(|x|)]$  allora  $L$  è decidibile.

### dim

Sia  $f$  una funzione calcolabile allora esiste una macchina di Turing  $T_f$  che con input  $x \in \mathbb{N}$  calcola il valore  $f(n)$  e lo scrive sul nastro di output se e soltanto se  $f(n)$  è definita (in questo caso sempre, siccome da ipotesi  $f$  è una funzione totale).

Sia  $L \subseteq \Sigma^*$  un linguaggio accettato da una macchina di Turing non deterministica  $NT$  tale che  $\forall x \in \Sigma^* [ntime(NT, x) \leq f(|x|)]$

Da  $NT$  e  $T_f$  deriviamo una nuova macchina di Turing non deterministica  $NT'$  che utilizza tre nastri, sul primo verrà scritto l'input  $x \in \Sigma^*$  ed opera nel seguente modo:

1. Nella prima fase viene letta ogni cella del primo nastro e ogni volta che viene letto un carattere diverso da  $\square$  viene scritto un uno sul secondo nastro e vengono fatte muovere le testine di entrambi i nastri di una posizione a destra. Una volta che viene letto  $\square$  sul primo nastro vengono spostate le testine di entrambi i nastri sulla cella più a sinistra che contiene un carattere diverso da  $\square$ . Al termine della prima fase vi sarà scritto sul secondo nastro il valore di  $|x|$  in unario.
2. Nella seconda fase viene simulata la computazione  $T_f(|x|)$  utilizzando il secondo nastro come nastro di lavoro. Verrà calcolato il valore di  $f(|x|)$  e verrà scritto sul terzo nastro. Verrà poi spostata la testina sul terzo nastro sulla cella più a sinistra contenente un carattere diverso da  $\square$ .
3. Nella terza fase viene simulata la computazione di  $NT(x)$  utilizzando il primo nastro come nastro di lavoro e il terzo come contatore del numero di istruzioni eseguite. Ogni volta che viene letto 1 sul terzo nastro viene eseguita una quintupla di  $NT$ . Una volta che è stata eseguita la quintupla viene fatta spostare la testina sul terzo nastro di una posizione a destra. Se la computazione  $NT(x)$  termina nello stato di rigetto o di accettazione, la macchina  $NT'$  termina nel medesimo stato. Nel caso sul terzo nastro venga letto il carattere  $\square$  la macchina  $NT'$  termina nello stato di rigetto.

Osserviamo che poiché la funzione  $f$  è totale la seconda fase termina sempre e poiché la computazione  $NT(x)$  viene forzatamente terminata nel caso venga letto  $\square$  sul terzo nastro, anche la terza fase termina sempre. Tutte le computazioni  $NT'$  terminano sempre.

- Se  $x \in L$ , allora la computazione  $NT(x)$  termina in  $f(|x|)$  passi, dunque la terza fase termina nello stato di accettazione.
- Se  $x \notin L$  allora o la computazione  $NT(x)$  termina nello stato di rigetto in  $f(|x|)$  passi oppure la computazione  $NT(x)$  non termina prima che venga letto  $\square$  sul terzo nastro.

Questo dimostra che  $NT'$  decide  $L$ . Dunque  $L$  è decidibile.

## Classi complessità

- $DTIME[f(n)] = \{ L \subseteq \Sigma^* : \Sigma \text{ è un alfabeto finito e } L \text{ è un linguaggio deciso da una macchina di Turing } T \text{ deterministica tale che } \forall x \in \Sigma^* [dtime(T, x) \in O(f(|x|))] \}$
- $NTIME[f(n)] = \{ L \subseteq \Sigma^* : \Sigma \text{ è un alfabeto finito e } L \text{ è un linguaggio accettato da una macchina di Turing } NT \text{ non deterministica tale che } \forall x \in \Sigma^* [ntime(NT, x) \in O(f(|x|))] \}$
- $DSPACE[f(n)] = \{ L \subseteq \Sigma^* : \Sigma \text{ è un alfabeto finito e } L \text{ è un linguaggio deciso da una macchina di Turing } T \text{ deterministica tale che } \forall x \in \Sigma^* [dtime(T, x) \in O(f(|x|))] \}$
- $NSPACE[f(n)] = \{ L \subseteq \Sigma^* : \Sigma \text{ è un alfabeto finito e } L \text{ è un linguaggio accettato da una macchina di Turing } NT \text{ non deterministica tale che } \forall x \in \Sigma^* [nspace(NT, x) \in O(f(|x|))] \}$
- $coDTIME[f(n)] = \{ L \subseteq \Sigma^* : L^c \text{ è un linguaggio deciso da una macchina di Turing } T \text{ deterministica tale che } \forall x \in \Sigma^* [dtime(T, x) \in O(f(|x|))] \}$
- $coNTIME[f(n)] = \{ L \subseteq \Sigma^* : L^c \text{ è un linguaggio accettato da una macchina di Turing } NT \text{ non deterministica tale che } \forall x \in \Sigma^* [ntime(NT, x) \in O(f(|x|))] \}$

- $coDSPACE[f(n)] = \{ L \subseteq \Sigma^* : L^c \text{ è un linguaggio deciso da una macchina di Turing } T \text{ deterministica tale che } \forall x \in \Sigma^* [dtime(T, x) \in O(f(|x|))] \}$
- $coNSPACE[f(n)] = \{ L \subseteq \Sigma^* : L^c \text{ è un linguaggio accettato da una macchina di Turing } NT \text{ non deterministica tale che } \forall x \in \Sigma^* [nspace(NT, x) \in O(f(|x|))] \}$

## Teorema

Sia  $f : \mathbb{N} \rightarrow \mathbb{N}$  una funzione time-constructible

$\forall L \in NTIME[f(n)]$  si ha che  $L$  è decidibile in tempo non deterministico  $O(f(n))$

**dim**

Sia  $f : \mathbb{N} \rightarrow \mathbb{N}$  una funzione time-constructible allora esiste una macchina di Turing  $T_f$  trasduttore che su input  $n \in \mathbb{N}$  sul nastro di input codificato in unario, calcola il valore  $f(n)$  in  $O(f(n))$  passi e lo scrive sul nastro di output.

Sia  $L$  un linguaggio e  $L \in NTIME[f(n)]$  allora esiste una macchina di Turing  $NT$  che accetta  $L$  e  $\forall x \in \Sigma^* [ntime(NT, x) \in O(f(|x|))]$ . Da  $T_f$  e  $NT$  deriviamo una nuova macchina di Turing non deterministica riconoscitore  $NT'$  a tre nastri, sul primo dei quali vi è l'input  $x \in \Sigma^*$ . Che opera in 3 fasi:

1. Nella prima fase viene scritto il valore di  $|x|$  in unario sul secondo nastro. Viene letta ogni cella del primo nastro e ogni volta che viene letto un carattere diverso da  $\square$  viene scritto un uno sul secondo nastro e vengono fatte muovere le testine di entrambi i nastri di una posizione a destra. Una volta che viene letto  $\square$  sul primo nastro vengono spostate le testine di entrambi i nastri sulla cella più a sinistra che contiene un carattere diverso da  $\square$ . Al termine della prima fase vi sarà scritto sul secondo nastro il valore di  $|x|$  in unario.
2. Nella seconda fase viene simulata la computazione  $T_f(|x|)$  utilizzando il secondo nastro come nastro di lavoro. Verrà calcolato il valore di  $f(|x|)$  e verrà scritto sul terzo nastro. Verrà poi spostata la testina sul terzo nastro sulla cella più a sinistra contenente un carattere diverso da  $\square$ .
3. Nella terza fase viene simulata la computazione di  $NT(x)$  utilizzando il primo nastro come nastro di lavoro e il terzo come contatore del numero di istruzioni eseguite. Ogni volta che viene letto 1 sul terzo nastro viene eseguita una quintupla di  $NT$ . Una volta che è stata eseguita la quintupla viene fatta spostare la testina sul terzo nastro di una posizione a destra. Se la computazione  $NT(x)$  termina nello stato di rigetto o di accettazione, la macchina  $NT'$  termina nel medesimo stato. Nel caso sul terzo nastro venga letto il carattere  $\square$  la macchina  $NT'$  termina nello stato di rigetto.

Siccome  $f$  è una funzione time constructible, la seconda fase termina sempre, e siccome viene forzatamente terminata anche la terza fase, tutte le computazioni di  $NT'$  terminano sempre.

La prima fase termina in  $O(|x|)$  passi.

Siccome  $f$  è una funzione time constructible la seconda fase termina in  $O(f(|x|))$  passi.

Siccome  $ntime(NT, x) \in O(f(|x|))$ , la terza fase termina in  $O(f(|x|))$  passi.

Dunque  $NT'$  decide  $L$  e  $\forall x \in \Sigma^* [ntime(NT', x) \in O(f(|x|))]$ .

## Teorema

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

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

**dim**

Sia  $f$  una funzione time-constructible, allora esiste una macchina di Turing trasduttore  $T_f$  che con input  $n \in \mathbb{N}$ , in  $O(f(n))$  passi, calcola  $f(n)$  e scrive il suo valore sul nastro di output

Sia  $L \subseteq \Sigma^*$  e  $L \in NTIME[f(n)]$  allora esiste una macchina di Turing non deterministica  $NT$  e una costante  $h$  per cui  $NT$  accetta  $L$  e per ogni  $x \in \Sigma^*$   $ntime(NT, x) \leq hf(|x|)$ .

Da  $NT$  e  $T_f$  deriviamo una macchina di Turing deterministica  $T$  a tre nastri, sul primo dei quali vi sarà l'input  $x$ .  $T$  simula in successione tutte le computazioni deterministiche di  $NT(x)$  di lunghezza  $hf(|x|)$ . In modo più dettagliato opera nel seguente modo:

1. Usando come nastro di lavoro il primo nastro, ogni volta che incontra un valore diverso da  $\square$ , scrive un 1 sul secondo nastro e sposta la testina sul secondo nastro di una posizione a destra. Una volta che sul primo nastro viene letto  $\square$  vengono fatte spostare entrambe le testine sulla cella con carattere più a sinistra diverso da  $\square$ .

- Viene simulata la computazione  $T_f(|x|)$ : utilizzando come nastro di lavoro il secondo nastro, viene calcolato il valore di  $f(|x|)$  e viene scritto sul terzo nastro, in unario. E infine viene concatenato  $h$  volte il contenuto del terzo nastro, cos' da avere il valore in unario di  $hf(x)$ .
- Vengono simulate tutte le computazioni deterministiche di  $NT(x)$  di lunghezza  $hf(|x|)$ , utilizzando la posizione della testina del terzo nastro come contatore.

Ora se  $x \in L$  siccome  $ntime(NT, x) \leq hf(|x|)$ , allora o in  $hf(|x|)$  passi la computazione  $NT(x)$  termina nello stato di accettazione oppure  $x \notin L$ . Dunque se dopo aver simulato tutte le computazioni deterministiche di  $NT(x)$  di lunghezza  $hf(|x|)$ ,  $T$  non ha terminato nello stato di accettazione, allora può correttamente entrare nello stato di rigetto. Dunque  $T$  decide  $L$ .

Dimostriamo ora che lo fa in tempo non deterministico  $O(f(n))$

Indichiamo con  $k$  il grado di non determinismo di  $NT$ , il numero di computazioni deterministiche di  $NT(x)$  è  $k^{hf(|x|)}$ . Ogni computazione deterministica di  $NT(|x|)$  di lunghezza  $hf(|x|)$  viene eseguita in  $O(f(|x|))$  passi. Poiché il passo 2. richiede  $O(f(|x|))$  passi e il passo 1.  $O(|x|)$  passi abbiamo che:

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

$$\text{Dunque } L \in DTIME[2^{O(f(|x|))}]$$

## Specifiche classi di complessità

$P = \bigcup_{k \in \mathbb{N}} DTIME[n^k]$ : è la classe dei linguaggi decisi in tempo deterministico polinomiale

$NP = \bigcup_{k \in \mathbb{N}} NTIME[n^k]$ : è la classe dei linguaggi accettati in tempo non deterministico polinomiale

$PSPACE = \bigcup_{k \in \mathbb{N}} DSPACE[n^k]$ : è la classe dei linguaggi decisi in spazio deterministico polinomiale

$NPSpace = \bigcup_{k \in \mathbb{N}} NSPACE[n^k]$ : è la classe dei linguaggi accettati in spazio non deterministico polinomiale

$EXPTIME = \bigcup_{k \in \mathbb{N}} DTIME[2^n]$ : è la classe dei linguaggi decisi in tempo deterministico esponenziale, ove l'esponente che descrive la funzione limite è un polinomio;

$NEXPTIME = \bigcup_{k \in \mathbb{N}} NTIME[2^n]$ : è la classe dei linguaggi accettati in tempo non deterministico esponenziale, ove l'esponente che descrive la funzione limite è un polinomio;

$$FP = \bigcup_{k \in \mathbb{N}} \{ f : \Sigma_1^* \rightarrow \Sigma_2^* : \exists \text{ una macchina di Turing trasduttore } T_f \text{ che calcola } f \text{ e } \forall x \in \Sigma_1^* [dtime(T_f, x) \in O(n^k)] \}$$

$$coP = \{ L \subseteq \Sigma^* : \Sigma \text{ è un alfabeto finito e } L^c \in P \}$$

$$coNP = \{ L \subseteq \Sigma^* : \Sigma \text{ è un alfabeto finito e } L^c \in NP \}$$

## Corollario

$$coP = P$$

## Teorema

Siano  $C$  e  $C'$  due classi di complessità tali che  $C' \subseteq C$ . Se  $C'$  è chiusa rispetto alla  $\pi$ -riducibilità, allora per ogni linguaggio  $L$  che sia  $C$ -completo rispetto a tale  $\pi$ -riduzione,  $L \in C' \iff C = C'$

**dim**

$$(\implies)$$

Partiamo dall'ipotesi che  $L \in C'$ . Siccome  $L$  è  $C$  completo, allora appartiene alla classe  $C$  e inoltre per ogni  $L' \in C$  abbiamo che  $L' \leq_\pi L$ . Siccome  $C'$  è chiusa rispetto la riducibilità polinomiale abbiamo che per ogni coppia di linguaggi  $L_1$  e  $L_2$  per cui  $L_1 \leq_\pi L_2$  e  $L_2 \in C$  allora  $L_1 \in C$ . Dunque nel nostro caso implica che per ogni  $L' \in C$ ,  $L' \in C'$ . Dunque  $C = C'$

$$(\impliedby)$$

Facile, se  $C = C'$  allora  $L \in C'$

## Teorema

La classe di complessità  $P$  è chiusa rispetto la riducibilità polinomiale

**dim**

Sia un linguaggio  $L \subseteq \Sigma_2^*$  e  $L \in \mathbf{P} \implies \exists$  una macchina di Turing deterministica  $T$  che decide  $L$  e una costante  $k \in \mathbb{N} \wedge \forall x \in \Sigma^*[dtime(T, x) \leq |x|^k]$ .

Sia  $L' \subseteq \Sigma_1^*$  e  $L' \leq_p L$  allora esiste una funzione totale e calcolabile  $f: \Sigma_1^* \rightarrow \Sigma_2^*$  e  $f \in FP$  allora  $\forall x \in \Sigma_1^*[x \in L' \iff f(x) \in L]$ . Siccome  $f$  è una funzione calcolabile  $\implies \exists$  una macchina di Turing trasduttore  $T_f$  e una costante  $h \in \mathbb{N}: \forall x \in \Sigma_1^*[ \text{calcola } f(x) \text{ e lo scrive sul nastro di output} \wedge dtime(T_f, x) \leq |x|^h ]$ .

Dalla macchina  $T$  e  $T_f$  deriviamo una nuova macchina di Turing riconoscitore  $T'$  che decide  $L'$ .

$T'$  utilizza 2 nastri. Il primo dei quali contiene l'input  $x \in \Sigma_1^*$ . La macchina  $T$  opera nel seguente modo:

1. Viene simulata la computazione  $T_f(x)$ , utilizzando il primo nastro, come nastro di lavoro e scrive il valore  $f(x)$  sul secondo nastro.
2. Viene simulata la computazione  $T(f(x))$ . Se  $T(f(x))$  termina nello stato di rigetto allora  $T'(x)$  termina nello stato di rigetto. Se  $T(f(x))$  termina nello stato di accettazione, allora  $T'(x)$  termina nello stato di accettazione.

$f$  è una riduzione da  $L'$  a  $L$ , quindi  $f(x) \in L \iff x \in L'$ . In conclusione  $T'$  termina per ogni input e  $T'(x)$  accetta se e soltanto se  $x \in L'$ . Dunque  $T'$  decide  $L'$

Dobbiamo ora mostrare che  $T$  opera in tempo deterministico polinomiale in  $|x|$ . La simulazione  $T_f(x)$  richiede  $dtime(T_f, x) \leq |x|^h$  passi e la simulazione  $T(f(x))$  termina in  $dtime(T, f(x)) \leq |f(x)|^k$  passi. Dunque  $dtime(T', x) \leq |x|^h + |f(x)|^k$ . Dobbiamo capire quanto è grande  $|f(x)|^h$ : Siccome  $dtime(T_f, x) \leq |x|^k$  e  $T_f$  deve almeno scrivere il suo output  $f(x)$ , allora  $|f(x)| \leq |x|^h$  (Altrimenti  $T_f$  non riuscirebbe a scriverla sul suo nastro di output in  $|x|^k$  passi). Dunque

$$dtime(T', x) \leq |x|^h + |f(x)|^k \leq |x|^h + (|x|^h)^k = |x|^h + |x|^{hk}$$

E poiché  $h$  e  $k$  sono costanti,  $L' \in \mathbf{P}$

**Corollario**

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

**dim**

Supponiamo che  $L \in \mathbf{P}$  e che sia  $\mathbf{NP}$ -completo. Allora per ogni linguaggio  $L' \in \mathbf{NP}$ ,  $L' \leq L$ , ma se  $L \in \mathbf{P}$ , siccome la classe  $\mathbf{P}$  è chiusa rispetto la riducibilità polinomiale questo implica che per ogni  $L'$  in  $\mathbf{NP}$ ,  $L' \in \mathbf{P}$ , dunque  $\mathbf{P} = \mathbf{NP}$ , contraddicendo l'ipotesi iniziale.

**Teorema**

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

**dim**

Supponiamo che  $\mathbf{P} = \mathbf{NP}$ , allora siccome  $\mathbf{coP} = \mathbf{coNP}$ . In virtù del precedente corollario,  $\mathbf{P} = \mathbf{coP}$ , allora:

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

**Teorema**

La classe  $\mathbf{coNP}$  è chiusa rispetto la riducibilità polinomiale

**dim**

Poiché  $\mathbf{NP}$  è chiusa rispetto la riducibilità polinomiale dunque per ogni coppia di linguaggi  $L_1$  e  $L_2$  tale che  $L_1 \leq_p L_2$  e  $L_2 \in \mathbf{NP}$  allora  $L_1 \in \mathbf{NP}$ . Dunque per ogni coppia  $L_1^c$  e  $L_2^c$  tale che  $L_1^c \leq_p L_2^c$  e  $L_2^c \in \mathbf{coNP}$  allora  $L_1^c \in \mathbf{coNP}$ .

**Teorema**

Un linguaggio  $L$  è  $\mathbf{NP}$ -completo se e soltanto se  $L^c$  è  $\mathbf{coNP}$ -completo

**dim**

( $\implies$ )



Se  $L$  è **NP**-completo, allora  $L \in \mathbf{NP}$  e dunque  $L^c \in \mathbf{coNP}$ .

Inoltre siccome  $L$  è **NP**-completo, abbiamo che per ogni linguaggio  $L' \in \mathbf{NP}$  tale che  $L' \leq L$ . Questo significa che per ogni  $L' \in \mathbf{NP}$  esiste una funzione  $f: \Sigma' \rightarrow \Sigma$  e  $f \in \mathbf{FP}$  per cui  $\forall x \in \Sigma' [x \in L' \iff f(x) \in L]$ .

Ma questo significa che per ogni  $\forall x \in \Sigma' [x \notin L \iff f(x) \notin L]$ .

Ossia  $\forall x \in \Sigma' [x \in L'^c \iff f(x) \in L^c]$ : Quindi  $L^c$  è **coNP**-completo

( $\Leftarrow$ )

Se  $L$  è **coNP**-completo, allora  $L \in \mathbf{coNP}$  e dunque  $L^c \in \mathbf{NP}$ .

Inoltre siccome  $L$  è **coNP**-completo, abbiamo che per ogni linguaggio  $L' \in \mathbf{coNP}$  tale che  $L' \leq L$ . Questo significa che per ogni  $L' \in \mathbf{coNP}$  esiste una funzione  $f: \Sigma' \rightarrow \Sigma$  e  $f \in \mathbf{FP}$  per cui  $\forall x \in \Sigma' [x \in L' \iff f(x) \in L]$ .

Ma questo significa che per ogni  $\forall x \in \Sigma' [x \notin L \iff f(x) \notin L]$ .

Ossia  $\forall x \in \Sigma' [x \in L'^c \iff f(x) \in L^c]$ : Quindi  $L^c$  è **NP**-completo

## Teorema

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

dim

Siccome  $L$  è **NP**-completo allora:

1.  $L \in \mathbf{NP}$
2.  $\forall L' \in \mathbf{NP} [L' \leq L]$

( $\subseteq$ )

Poiché  $L \in \mathbf{coNP}$  allora per ogni  $L_1 \in \mathbf{NP}$ ,  $L_1 \leq L$ , ma **coNP** è chiusa rispetto la riducibilità polinomiale ovvero  $L_2 \in \mathbf{coNP}$  e  $L_1 \leq L_2 \implies L_1 \in \mathbf{coNP}$ , allora per ogni  $L_1 \in \mathbf{NP}$  si ha che  $L_1 \leq L$  e  $L \in \mathbf{coNP}$ . Dunque per la chiusura di **coNP**,  $L_1 \in \mathbf{coNP}$ . Quindi **NP**  $\subseteq$  **coNP**

( $\supseteq$ )

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

## Teoremi Dispensa 7

### Teorema

Sia  $\Gamma = \langle I_\Gamma, S_\Gamma, \pi_\Gamma \rangle$  un problema decisionale e sia  $\chi: I_\Gamma \rightarrow \Sigma^*$  una sua codifica ragionevole. Se  $\chi(I_\Gamma) \in \mathbf{P}$ , allora  $L_\Gamma(\chi) \in \mathbf{NP} \implies L_{\Gamma^c}(\chi) \in \mathbf{coNP}$

dim

Poiché  $\chi(I_\Gamma) \in \mathbf{P}$ , allora esistono una macchina di Turing deterministica  $T$  ed un intero  $h$  tali che, per ogni  $x \in \Sigma^*$ ,  $T$  decide se  $x \in \chi(I_\Gamma)$  e  $dtime(T, x) \in O(|x|^h)$ .

Se  $L_\Gamma(\chi) \in \mathbf{NP}$ , allora esistono una macchina di Turing non deterministica  $NT_\Gamma$  ed un intero  $k$  tali che, per ogni  $x \in L_\Gamma(\chi)$ ,  $NT_\Gamma$  accetta  $x$  e  $ntime(NT_\Gamma, x) \in O(|x|^k)$ .

Combinando  $T$  e  $NT_\Gamma$ , deriviamo una nuova macchina non deterministica  $NT'$  che, con input  $x \in \Sigma^*$ , opera in due fasi, come di seguito descritto.

1. Simula la computazione  $T(x)$ : se  $T(x)$  termina nello stato di rigetto, allora  $NT'$  termina nello stato di accettazione, altrimenti ha inizio la 2.
2. Simula la computazione  $NT_\Gamma(x)$ .

Quindi,  $NT'(x)$  accetta se e soltanto se  $x \notin \chi(I_\Gamma)$  oppure  $x \in L_\Gamma(\chi)$ , ossia, se e soltanto se  $x$  appartiene al linguaggio complemento di  $L_{\Gamma^c}(\chi)$ . Inoltre, è semplice verificare che  $ntime(NT', x) \in O(|x|^{\max\{h, k\}})$ .

In conclusione, il linguaggio complemento di  $L_{\Gamma^c}(\chi)$  è in **NP**, e dunque  $L_{\Gamma^c}(\chi) \in \mathbf{coNP}$ .

# Teorema Dispensa 9

## Teorema

Un linguaggio  $L \subseteq \Sigma^*$  è in **NP** se e soltanto se esistono una macchina di Turing deterministica  $T$  che opera in tempo polinomiale e una costante  $k \in \mathbb{N}$  tali che per ogni  $x \in \Sigma^*$ ,

$$x \in L \iff \exists y_x \in \{0, 1\}^* : |y_x| \leq |x|^k \wedge T(x, y_x) \text{ accetta}$$

dim

( $\implies$ )

Sia  $L \subseteq \Sigma^*$  un linguaggio in **NP**: per definizione di **NP**, esistono una macchina di Turing non deterministica  $NT$  e un intero  $h \in \mathbb{N}$  tali che  $NT$  accetta  $L$  e, per ogni  $x \in L$ ,  $\text{time}(NT, x) \leq |x|^h$ . Questo significa che, per ogni  $x \in L$  esiste una computazione deterministica di  $NT(x)$  che termina nello stato di accettazione, ossia esiste una sequenza di quintuple  $p_1, \dots, p_{|x|^k}$  che, eseguite a partire dallo stato globale di  $NT$  in cui l'input  $x = x_1 x_2 x_3 \dots x_n$  è scritto sul nastro, lo stato interno è lo stato iniziale  $q_0$  di  $NT$  e la testina è posizionata sulla cella contenente  $x_1$ , porta ad uno stato globale di accettazione. Allora indichiamo con  $p_i = \langle q_{i_1}, s_{i_1}, s_{i_2}, q_{i_2}, m_i \rangle$  la  $i$ -esima quintupla della sequenza, per ogni  $i = 1, \dots, n^k$ , deve essere  $q_{1_1} = q_0, q_{(n^k)_2} = q_A$  e, per ogni  $2 \leq i \leq n^k, q_{i_1} = q_{(i-1)_2}$ . Poniamo allora:

$$y(x) = q_{1_1}, s_{1_1}, s_{1_2}, q_{1_2}, m_1 - q_{2_1}, s_{2_1}, s_{2_2}, q_{2_2}, m_2 - \dots - q_{n^k_1}, s_{n^k_1}, s_{n^k_2}, q_{n^k_2}, m_{(n^k)}$$

ossia  $y(x)$  è la parola nell'alfabeto  $\Sigma \cup Q \cup \{-, s, f, d\}$  ottenuta concatenando le parole che corrispondono alla sequenza accettante di quintuple. Sulla base della precedente osservazione, definiamo, ora, una macchina di Turing deterministica a due nastri (a testine indipendenti)  $\bar{T}$  che corrisponde alla macchina  $NT$ : essa possiede, codificata nelle sue quintuple, la descrizione dell'insieme  $P$  delle quintuple di  $NT$ . La computazione  $\bar{T}(x, y)$ , con input  $x \in \Sigma^*$  scritto sul primo nastro e  $y \in (\Sigma \cup Q \cup \{-, s, f, d\})^*$  scritto sul secondo nastro, procede come di seguito descritto.

1.  $\bar{T}$  verifica che  $y$  sia nella forma  $y(x) = q_{1_1}, s_{1_1}, s_{1_2}, q_{1_2}, m_1 - q_{2_1}, s_{2_1}, s_{2_2}, q_{2_2}, m_2 - \dots - q_{n^k_1}, s_{n^k_1}, s_{n^k_2}, q_{n^k_2}, m_{(n^k)}$ : se così non fosse, rigetta.
2.  $\bar{T}$  verifica che, per ogni  $1 \leq i \leq n^k, \langle q_{i_1}, s_{i_1}, s_{i_2}, q_{i_2}, m_i \rangle \in P$ : se così non è rigetta
3.  $\bar{T}$  verifica che  $q_{1_1} = q_0$  e  $q_{(n^k)_2} = q_A$ : se così non è, rigetta.
4.  $\bar{T}$  verifica che, per ogni  $2 \leq i \leq n^k, q_{i_1} = q_{(i-1)_2}$ : se così non è, rigetta.
5.  $\bar{T}$  simula la computazione di  $NT(x)$  descritta da  $y$ :

5.1 Con la testina posizionata sulla cella contenente  $x_1$ , verifica se la quintupla  $\langle q_{1_1}, s_{1_1}, s_{1_2}, q_{1_2}, m_1 \rangle$  può essere eseguita, ossia, se  $s_{1_1} = x_1$  e, se è così, eseguila modificando (eventualmente) il contenuto della cella del primo nastro su cui è posizionata la testina e spostando (eventualmente) la testina sul primo nastro, altrimenti rigetta;

5.2 per ogni  $2 \leq i \leq n^k$ , verifica se la quintupla  $q_{i_1}, s_{i_1}, s_{i_2}, q_{i_2}, m_i$  può essere eseguita, ossia, se il simbolo letto dalla testina è  $s_{i_1}$  e, se è così, la esegue modificando (eventualmente) il contenuto della cella del primo nastro su cui è posizionata la testina e spostando (eventualmente) la testina sul primo nastro, altrimenti rigetta.

6.  $\bar{T}$  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  $\bar{T}$  opera in tempo polinomiale in  $|x|$ .

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

( $\impliedby$ )

sia  $L \subseteq \Sigma^*$  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, per ogni  $x \in \Sigma^*, x \in L \iff \exists y_x \in \{0, 1\}^* : |y_x| \leq |x|^k \wedge T(x, y_x)$  accetta.

Senza perdita di generalità, assumiamo che  $T$  disponga di un solo nastro sul quale, inizialmente, sono scritte le due parole  $x$  e  $y$  separate da un carattere '2'.

Definiamo la seguente macchina di Turing non deterministica  $NT$  che opera in due fasi, con input  $x$ :

- durante la prima fase genera una parola  $y \in \{0, 1\}^*$  di lunghezza al più  $|x|^k$  scrivendola sul nastro, alla destra di  $x$  e separato da esso da un carattere '2';

- durante la seconda fase simula la computazione  $T(x, y)$ .

Si osservi, innanzi tutto, che  $NT$  opera in tempo polinomiale in  $|x|$ : infatti, la prima fase, in cui viene generata  $y$ , richiede al più  $O(|x|^k)$  passi e la seconda fase richiede tempo proporzionale a  $dtime(T, (x, y))$  e quindi, poichè  $dtime(T, (x, y))$  è un polinomio in  $|x|$  e  $|y|$  e  $|y| \leq |x|^k$ , polinomiale in  $|x|$ .

Sia  $x \in \Sigma^*$ . Se  $x \in L$ , allora, per ipotesi, esiste una parola  $y_x \in \{0, 1\}^*$  di lunghezza  $|y_x| \leq |x|^k$  tale che  $T(x, y_x)$  accetta.

Allora, durante la prima fase della computazione di  $NT(x)$  esiste una sequenza di scelte che genera  $y_x$  che, nella seconda fase, induce  $NT$  ad accettare. Quindi,  $NT$  accetta  $x$ . Di contro, se  $NT$  accetta  $x$  allora esiste una parola  $y_x \in \{0, 1\}^*$  di lunghezza  $|y_x| \leq |x|^k$ , generata durante la prima fase, che induce la seconda fase ad accettare: ma poichè la seconda fase di  $NT$  non è altro che una simulazione di  $T$ , questo significa  $T(x, y_x)$  accetta. Quindi,  $x \in L$ .

In conclusione,  $x \in L$  se e soltanto se  $NT(x)$  accetta, e, poichè  $NT$  è una macchina di Turing non deterministica che opera in tempo polinomiale, questo prova che  $L \in \mathbf{NP}$ .

## Teorema

Sia  $\Gamma_0$  un problema in  $\mathbf{NP}$ . Se esiste un problema  $\mathbf{NP}$ -completo riducibile a  $\Gamma_0$ , allora  $\Gamma_0$  è  $\mathbf{NP}$ -completo

### dim

Sia  $\Gamma_1$  un problema  $\mathbf{NP}$ -completo tale che  $\Gamma_1 \leq \Gamma_0$ . Poichè  $\Gamma_1 \leq \Gamma_0$ , esiste una funzione  $f_{10} : I_{\Gamma_1} \rightarrow I_{\Gamma_0}$  tale che  $f_{10} \in \mathbf{FP}$  e per ogni  $x \in I_{\Gamma_1}$   $x \in \Gamma_1 \iff f_{10}(x) \in \Gamma_0$

Poichè  $\Gamma_1$  è  $\mathbf{NP}$ -completo, per ogni problema  $\Gamma_2 \in \mathbf{NP}$ , si ha che  $\Gamma_2 \leq \Gamma_1$  e dunque esiste una funzione  $f_{21} : I_{\Gamma_2} \rightarrow I_{\Gamma_1}$  :  $f_{21} \in \mathbf{NP} \wedge \forall x \in I_{\Gamma_2} [x \in \Gamma_2 \iff f_{21}(x) \in \Gamma_1]$ .

Mostriamo ora che la composizione  $f_{21}$  e  $f_{10}$  è una riduzione polinomiale da  $\Gamma_2$  a  $\Gamma_0$ :

Sia  $x \in I_{\Gamma_2}$  : allora,  $x \in \Gamma_2$  se e soltanto se  $f_{21}(x) \in \Gamma_1$  e, inoltre,  $f_{21}(x) \in \Gamma_1$  se e soltanto se  $f_{10}(f_{21}(x)) \in \Gamma_0$ . Se indichiamo con  $f_{20}$  la composizione delle funzioni  $f_{21}$  e  $f_{10}$ , questo dimostra che  $f_{20}$  è una riduzione da  $\Gamma_2$  a  $\Gamma_0$ .

Poichè  $f_{21} \in \mathbf{NP}$ , esistono una macchina di Turing di tipo trasduttore  $T_{21}$  e un intero  $k \in \mathbb{N}$  tali che, per ogni  $x \in I_{\Gamma_2}$ ,  $T_{21}(x)$  calcola  $f_{21}(x)$  e  $dtime(T_{21}, x) \leq |x|^k$ . Osserviamo che, poichè la computazione  $T_{21}(x)$  deve anche scrivere il risultato  $f_{21}(x)$  sul nastro di output, questo implica che  $|f_{21}(x)| \leq dtime(T_{21}, x)$ , ossia,  $|f_{21}(x)| \leq |x|^k$ .

Analogamente, poichè  $f_{10} \in \mathbf{FP}$ , esistono una macchina di Turing di tipo trasduttore  $T_{10}$  e un intero  $h \in \mathbb{N}$  tali che, per ogni  $x \in I_{\Gamma_1}$ ,  $T_{10}(x)$  calcola  $f_{10}(x)$  e  $dtime(T_{10}, x) \leq |x|^h$ . Allora, possiamo definire la seguente macchina di Turing di tipo trasduttore  $T_{20}$  che calcola  $f_{20}$ : quando la computazione  $T_{20}(x)$  ha inizio, l'input  $x \in I_{\Gamma_2}$  è scritto sul nastro di lavoro e, a questo punto:

- Viene eseguita la computazione  $T_{21}(x)$  scrivendo il suo output  $f_{21}(x)$  sul nastro di lavoro
- Utilizzando il risultato della computazione  $T_{21}(x)$ , viene eseguita la computazione  $T_{10}(f_{21}(x))$  ed il suo output viene scritto sul nastro di output

Infine, in virtù del fatto che  $|f_{21}(x)| \leq |x|^k$ , per ogni  $x \in \Sigma_2$ ,

$$dtime(T_{20}, x) \leq |x|^k + |f_{21}(x)|h \leq |x|^k + |x|^{kh} \leq 2|x|^{kh} \leq |x|^{kh} + 1.$$

Essendo  $h$  e  $k$  due valori costanti (indipendenti dall'input  $x$ ), questo dimostra che  $f_{20} \in \mathbf{FP}$ .

Quindi, abbiamo dimostrato che  $\Gamma_2 \leq \Gamma_0$ , e, poichè  $\Gamma_2$  è un qualunque problema in  $\mathbf{NP}$ , questo prova che ogni problema in  $\mathbf{NP}$  è riducibile polinomialmente a  $\Gamma_0$ . Dall'appartenenza di  $\Gamma_0$  a  $\mathbf{NP}$  segue che  $\Gamma_0$  è  $\mathbf{NP}$ -completo.