Teoremi Fondamenti

Teoremi Dispensa 3

Teorema

Un $L \subseteq$ 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'alfabero Σ. 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:

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

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 $\implies \exists T_1 : \forall x \in \Sigma^*[o_{T1}(x) = q_A \iff x \in L \wedge o_T(x) \neq q_A \iff x \in L^c]$

Sia L^c accettabile $\implies \exists T_2 : \forall x \in \Sigma^*[o_{T2}(x) = q_A \iff x \in L^c \wedge o_T(x) \neq q_A \iff x \in L]$

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 Σ,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 portà la macchina T nel suo stato di rigetto. Quindi T decide L.

Teorema

Un linguaggio $L \subseteq \Sigma^*$ è decidibile $\iff$ la funzione χ_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 χ_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 χ_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 χ_L è calcolabile.

$(\Longleftarrow)$

Sia χ_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 χ_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 linugaggio $L_f = \{ < x, y >: x \in \Sigma^* \ \wedge \ y = f(x) \} \subseteq \Sigma^* \times \Sigma_1^*$ 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 $x\Sigma^*$ 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 $< x, y >$ con $x \in \Sigma^*$ e $y \in \Sigma_1^*$, opera nella seguente maniera:

1. Sul primo nastro è scritto l'input $< x, y >$
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 $T((x, y))$ 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 $T((x, y))$ 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 $T((x, y))$ 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

Poichè $L_f \subseteq \Sigma^* \times \Sigma_1^*$ 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 le 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 $< x, y >$. Se x appartiene al dominio di f , allora esiste $\bar{y} \in \Sigma_1^*$ tale che $\bar{y} = f(x)$ e, quindi, $< x, \bar{y} > \in L_f$. 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 $< q'_R, y, y, q'_R, fm >$, 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'e 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. Poich´e T^* è ottenuta mediante semplici modifiche della macchina che dovrebbe decidere L_H , ne consegue che L_H non `e decidibile.

Teorema

Sia L_1 un linguaggio non decidibile e sia L_2 un secondo linguaggio tale che $L_1 \leq_m L_2$ allora L_2 non è decidibile.

dim

Indichiamo con f_{12} la funzione che riduce L_1 ad L_2. Se L_2 fosse decidibile, allora potremmo decidere se $x \in L_1$ nella maniera seguente: innanzi tutto, calcoleremmo $f_{12}(x)$ e poi decideremmo se $f_{12}(x) \in L_2$. Poichè $x \in L_1$ se e soltanto se $f_{12}(x) \in L_2$, l'esito della decisione circa $f_{12}(x) \in L_2$ risponderebbe anche al quesito “$x \in L1?$”.

Teorema

Sia L_1 un linguaggio non accettabile e sia L_2 un secondo linguaggio tale che $L_1 \leq_m L_2$ allora L_2 non è accettabile.

dim

Indichiamo con f_{12} la funzione che riduce L_1 ad L_2. Se L_2 fosse accettabile, allora potremmo accettare se $x \in L_1$ nella maniera seguente: innanzi tutto, calcoleremmo $f_{12}(x)$ e poi accetteremmo se $f_{12}(x) \in L_2$. Poichè $x \in L_1$ se e soltanto se $f_{12}(x) \in L_2$, l'esito della decisione circa $f_{12}(x) \in L_2$ risponderebbe anche al quesito “$x \in L1?$”.

Teorema

Sia L_3 un linguaggio decidibile e sia L_4 un secondo linguaggio tale che $L_4 \leq_m L_3$ allora L_4 è decidibile.

dim

Indichiamo con f_{43} la funzione che riduce L_4 ad L_3. Sia $x \in \{0,1\}$; decidiamo se $x \in L_4$ nella maniera seguente:
Innanzi tutto, calcoliamo $f_{43}(x)$ e poi decidiamo se $f_{43}(x) \in L_3$. Poich´e x ∈ L4 se e soltanto se $f_{43}(x) \in L_3$,
l'esito della decisione circa $f_{43}(x) \in L_3$ risponde anche al quesito “$x \in L_4$?”

Teorema

Sia L_3 un linguaggio accettabile e sia L_4 un secondo linguaggio tale che $L_4 \leq_m L_3$; allora L_4 è accettabile

dim

Indichiamo con f_{43} la funzione che riduce L_4 ad L_3. Sia $x \in \{0,1\}$; accettiamo se $x \in L_4$ nella maniera seguente:
Innanzi tutto, calcoliamo $f_{43}(x)$ e poi accettiamo se $f_{43}(x) \in L_3$. Poichè $x \in L_4$ se e soltanto se $f_{43}(x) \in L_3$,
l'esito della decisione circa $f_{43}(x) \in L_3$ risponde anche al quesito “$x \in L_4$?”

Teorema Dispensa 6

Misure di complessità

$\forall T$ deterministica (riconoscitore o trasduttore) che opera su alfabeto Σ $\forall x \in \Sigma^*$ 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 definzione, $dtime(T,x)$ 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 , utilizzamo 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 Σ (non contentente $\square$), 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 Σ 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 determinstica 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 piu 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 Tf 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 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 contente 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

- $DTIME[f(n)] = \{\, L \subseteq \Sigma^* :$
Σ è un alfabeto finito e L è un linguaggio deciso da una macchina di Turing T deterministica tale che $\forall x \in \Sigma^*[dtime(T, x) \in O(f(|\, x|))]\,\}$

- $NTIME[f(n)] = \{\, L \subseteq \Sigma^* :$
Σ è un alfabeto finito e L è un linguaggio accettato da una macchina di Turing NT non deterministica tale che $\forall x \in L[ntime(NT, x) \in O(f(|\, x|))]\,\}$

- $DSPACE[f(n)] = \{\, L \subseteq \Sigma^* :$
Σ è un alfabeto finito e L è un linguaggio deciso da una macchina di Turing T deterministica tale che $\forall x \in \Sigma^*[dspace(T, x) \in O(f(|\, x|))]\,\}$

- $NSPACE[f(n)] = \{\, L \subseteq \Sigma^* :$
Σ è un alfabeto finito e L è un linguaggio accettato da una macchina di Turing NT non deterministica tale che $\forall x \in L[nspace(NT, x) \in O(f(|\, x|))]\,\}$

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

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

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

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

Teorema

Sia $f : \mathbb{N} \to \mathbb{N}$ totale e calcolabile.
$DTIME[f(n)] \subseteq NTIME[f(n)]$
$DSPACE[f(n)] \subseteq NSPACE[f(n)]$

dim

Basta osservare che una macchina di Turing deterministica, non è altro che una macchina non deterministica con grado di non determinismo pari a 1. E se un linguaggio è deciso in k passi allora è anche accettato in k passi.

Teorema

Sia $f : \mathbb{N} \to \mathbb{N}$ totale e calcolabile.
$DTIME[f(n)] \subseteq DSPACE[f(n)]$
$NTIME[f(n)] \subseteq NSPACE[f(n)]$

dim

Sia $L \subseteq \Sigma^*$ e $L \in DTIME[f(n)] \implies \exists T$ macchina di Turing deterministica $, k \in \mathbb{N} : \forall x \in \Sigma^*[o_T(x) = q_A \iff x \in L \wedge\ o_T(x) = q_R \iff x \notin L \wedge\ dtime(T, x) \in O(f(|x|))]$.

Siccome $dspace(T,x) \leq dtime(T,x)$, abbiamo che $dspace(T,x) \in O(f(|x|))$.
Dunque $L \in DSPACE[f(n)]$

Teorema

Sia $f : \mathbb{N} \rightarrow \mathbb{N}$ totale e calcolabile.
$DSPACE[f(n)] \subseteq EXPTIME[f(n)]$
$NSPACE[f(n)] \subseteq NEXPTIME[f(n)]$

dim

Sia $L \subseteq \Sigma^*$ e $L \in DSPACE[f(n)] \implies \exists T \text{ macchina di Turing deterministica }, k \in \mathbb{N} : \forall x \in \Sigma^*[o_T(x) = q_A \iff x \in L \land\ o_T(x) = q_R \iff x \notin L \land\ dspace(T,x) \in O(f(|x|))]$.
Siccome $dtime(T,x) \leq dspace(T,x)|Q|(|\Sigma|+1)^{dspace(T,x)}$, ù
abbiamo che
$dtime(T,x) \leq |Q|2^{log(dspace(T,x))+log((|\Sigma|+1))dspace(T,x)}$.
dunque $dtime(T,x) \leq |Q|2^{O(f(|x|))}$
dunque $dtime(T,x) \in O(2^{O(f(|x|))})$

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

Teorema

Sia $f : \mathbb{N} \rightarrow \mathbb{N}$ totale e calcolabile.
$coDTIME[f(n)] = DTIME[f(n)]$
$coDSPACE[f(n)] = DSPACE[f(n)]$

dim

Sia $L \subseteq \Sigma^*$ e $L \in coDTIME[f(n)]$ allora $L^c \in DTIME[f(n)]$ dunque esiste una macchina di Turing T' che decide L^c e $\forall x \in \Sigma^*[dtime(T',x) \in O(f(|x|))]$. Da T' derivo una nuova macchina di Turing T invertendo però gli stati finali di T'. Dunque se $T'(x)$ accetta $T(x)$ rigetta. Osserviamo che $dtime(T',x) = dtime(T,x)$ dunque $DTIME[f(n)] \subseteq coDTIME[f(n)]$. La dimostrazione che $coDTIME[f(n)] \subseteq DTIME[f(n)]$

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 determinisitico $O(f(n))$

dim

Sia $f : \mathbb{N} \rightarrow \mathbb{N}$ una funzione time-constructible allora esiste una macchina di Turing T_f trasduttore che su input $n \in N$ 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 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 $f : \mathbb{N} \to \mathbb{N}$ 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 $\square$.
2. 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 $hf(x)$.
3. 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 determinsitiche 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 determinsitico $O(f(n))$
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 $h(f(|x|))$ 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:

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

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 determinisitico polinomiale

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

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

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

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

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

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

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

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

Corollario

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

Teorema

Siano C e C' due classi di complessità tali che $C' \subseteq C$. Se C' è chiusa rispetto alla π-riducibilità, allora per ogni linguaggio L che sia C-completo rispetto a tale π-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 $L \in \mathbf{P} \implies \exists \text{ una macchina di Turing deterministica } T \text{ che decide L e una costante } k \in \mathbb{N} \wedge \forall x \in \Sigma^*[dtime(T,x) \leq |x|^k)]$.

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

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 determinisitico 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 poiche 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 $L_1 \in NP$.
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 $\mathbf{NP}$-completo, allora $L \in \mathbf{NP}$ e dunque $L^c \in \mathbf{coNP}$.
Inoltre siccome L è $\mathbf{NP}$-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 $f : \Sigma' \to \Sigma$ 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 $\mathbf{coNP}$-completo

$(\impliedby)$

Se L è $\mathbf{coNP}$-completo, allora $L \in \mathbf{coNP}$ e dunque $L^c \in \mathbf{NP}$.
Inoltre siccome L è $\mathbf{coNP}$-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 $f : \Sigma' \to \Sigma$ 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 $\mathbf{NP}$-completo

Teorema

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

dim

Siccome L è $\mathbf{NP}$-completo allora:

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

$(\subseteq)$

Poiché $L \in coNP$ allora per ogni $L_1 \in \mathbf{NP}$, $L_1 \leq L$, ma $\mathbf{coNP}$ 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 $L_1 \in NP$ si ha che $L_1 \leq L$ e $L \in \mathbf{coNP}$. Dunque per la chisura di $\mathbf{coNP}$, $L_1 \in \mathbf{coNP}$. Quindi $\mathbf{NP} \subseteq \mathbf{coNP}$

$(\supseteq)$

Poiché $L \in coNP$ 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 $\mathbf{NP}$ 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 $L' \in \mathbf{coNP}, L' \leq L^c e L^c \in \mathbf{NP}$. Dunque per la chiusura di $\mathbf{NP}$, $L' \in \mathbf{NP}$ quindi $\mathbf{coNP} \subseteq \mathbf{NP}$.

Teoremi Dispensa 7

Teorema

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

dim

Poichè $\chi(I_\Gamma) \in \mathbf{P}$, 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_Γ ed un intero k tali che, per ogni $x \in L_\Gamma(\chi)$, NT_Γ accetta x e $ntime(NT_\Gamma, x) \in O(|x|^k)$.

Combinando T e NT_Γ, 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 $\mathbf{NP}$, e dunque $L_{\Gamma^c}(\chi) \in \mathbf{coNP}$.

Teorema Dispensa 9

Teorema

Un linguaggio $L \subseteq \Sigma^*$ è in $\mathbf{NP}$ se e soltanto se esistono una macchina di Turing determinisitica 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

dim

$(\implies)$

Sia $L \subseteq \Sigma^*$ un linguaggio in $\mathbf{NP}$: per definizione di $\mathbf{NP}$, esistono una macchina di Turing non determinisitica NT e un intero $h \in \mathbb{N}$ tali che NT accetta L e, per ogni $x \in L[ntime(NT, x) \leq |x|^h]$. 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 $p_1, ..., p_{|x|^k}$ che, eseguite a partire dallo stato globale di NT in cui l'input $x = x_1x_2x_3...x_n$ è scritto sul nastro, lo stato interno è lo stato inziiale q_0 di NT e la testina è posizionata sulla cella contenente x_1, porta ad uno stato globale di accettazione. Allora indichiamo con $p_i =< q_{i_1}, s_{i_1}, s_{i_2}, q_{i_2}, m_i >$ la i-esima quintupla della sequenza, per ogni $i = 1, ..., n^k$, deve essere $q1_1 = q_0, q_{(n^k)_2} = q_A$ e, per ogni $2 \leq i \leq n^k$, $q_{i_1} = q_{(i-1)_2}$. Poniamo allora:

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

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
$y(x) = q_{1_1}, s_{1_1}, s_{1_2}, q_{1_2}, m_1 - q_{2_1}, s_{2_1}, s_{2_2}, q_{2_2}, m_2 - ... - q_{n^k_1}, s_{n^k_1}, s_{n^k_2}, q_{n^k_2}, m_{(n^k)}$: 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 $\langle q_{1_1}, s_{1_1}, s_{1_2}, q1_2, m_1 \rangle$ 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.

$(\Longleftarrow)$

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 `e 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 Cook-Levin

Sia $\Gamma \in \mathbf{NP}$ e sia $L_\Gamma \subseteq \{0, 1\}^*$, il linguaggio che contiene la codifica delle istanze si di Γ.

Siccome $\Gamma \in \mathbf{NP} \implies \exists NT, h \text{ costante } : h \in \mathbb{N} \wedge \forall x \in \Sigma^*[o_{NT}(x) = q_A \iff x \in L_\Gamma \wedge o_{NT}(x) \neq q_A \iff x \notin L_\Gamma] \wedge \forall x \in L_\Gamma[ntime(NT, x) \leq |x|^h]$

L'affermazione $x \in L_\Gamma$ è equivalente alla seguente affermazione:
$\gamma(x) =$ "x (e solo x) è scritto sul nastro di input, **e** la testina è posizionata sulla cella contenente il primo carattere di x **e** la macchina si trova nel suo stato iniziale **e** esiste una sequenza di al più $|x|^h$ quintuple che eseguite in successione porta la macchina nel suo stato di accettazione q_A".

Dunque $x \in L_\Gamma$ se e soltanto se $\gamma(x)$ è vera.

Dobbiamo definire $\gamma(x)$ sottoforma di espressione booleana $E(x)$, ove $\gamma(x)$ è vera se e soltanto se $E(x)$ è soddisfacibile.

$E(x)$ deve descrivere dunque una computazione di NT che ha inizio con x (e solo x) scritto sul nastro di input.
Una computazione di NT non è altro che una successione di stati globali.

Dunque per costruire $E(x)$ è necessario introdurre le variabili booleane che descrivono per ogni passo t $(0 \leq t \leq |x|^h)$ della computazione lo stato globale in cui si trovrebbe la computazione di NT al passo t:

- L'insieme N che contiene le variabili booleane che permettono di descrivere i caratteri contenuti in ogni cella del nastro ad ogni *passo* della computazione
- L'insieme R che contiene le variabili booleane che permettono di descrivere la cella su cui è posizionata la testina della macchina ad ogni *passo* della computazione
- L'insieme M che contiene le variabili booleane che permettono di descrivere lo stato che assume la macchina NT ad oggni *passo* della computazione.

Insieme M

Sia $Q = \{ q_0, q_1, q_2, ..., q_k \}$ l'insieme degli stati della macchina NT, con q_0 stato iniziale, $q_1 = q_A, q_2 = q_R$.

Per ogni passo t della computazione ($0 \leq t \leq |x|^h$) e per ogni $i \in \{ 1, ..., k \} M$ contiene la variabile booleana M_i^t:

$M = \{ M_i^t : 0 \leq t \leq | x|^h \wedge 0 \leq i \leq k \}$.

Assegnando il valore vero alla variabile M_i^t indichiamo che al passo t della computazione la macchina si trova nello stato q_i.

Affinché le variabili booleane M_i^t descrivano correttamente lo stato della macchina dobbiamo imporre che siano coerenti, ovvero la macchina ad ogni passo t della computazione può trovarsi in uno e un solo stato, quindi dobbiamo prendere in considerazione le sole assegnazioni di verità che per $0 \leq t \leq |x|^h$ ad una sola delle $M_0^t, ..., M_k^t$ sia assegnato il valore **vero**.

Per fare ciò introduciamo $|x|^h + 1$ espressioni $E_M^0, ..., E_M^{|x|^h}$.
Dove E_M^t rappresenta la seguente affermazione:
"La macchina NT si trova in uno e un solo dei suoi stati interni"

$E_M^t = [M_0^t \wedge \neg M_1^t \wedge ... \wedge \neg M_k^t] \vee ... \vee [M_k^t \wedge \neg M_0^t \wedge ... \wedge M_{k-1}^t]$

E questa espressione è vera se e soltanto se viene assegnato valore vero ad una ed una sola variabile booleana M_i^t con $0 \leq t \leq |x^h| \ e \ 0 \leq i \leq k$.

Insieme R

Siccome la computazione $NT(x)$ accetta in al più $|x|^h$ passi allora utilizza al più $|x|^h$ celle del nastro.

Per ogni passo t della computazione ($0 \leq t \leq |x|^h$) e per ogni $0 \leq i \leq |x|^h R$ contiene la variabile booleana R_i^t:

$R = \{ R_i^t : 0 \leq t \leq | x|^h \wedge 0 \leq i \leq k \}$.

Assegnando il valore vero alla variabile R_i^t indichiamo che al passo t della computazione la testina della macchina si trova sulla cella i.

Affinché le variabili booleane R_i^t descrivano correttamente la cella su cui è posizionata la testina della macchina dobbiamo imporre che siano coerenti, ovvero la testina della macchina ad ogni passo t della computazione può trovarsi su una e un sola cella del nastro, quindi dobbiamo prendere in considerazione le sole assegnazioni di verità che per $0 \leq t \leq |x|^h$ ad una sola delle $R_1^t, ..., R_{|x|^h}^t$ sia assegnato il valore **vero**.

Per fare ciò introduciamo $|x|^h + 1$ espressioni $E_R^0, ..., E_R^{|x|^h}$.
Dove E_R^t rappresenta la seguente affermazione:
"La testina della macchina NT si trova su una e un sola cella del nastro"

$E_R^t = [R_0^t \wedge \neg R_1^t \wedge ... \wedge \neg R_k^t] \vee ... \vee [R_{|x|^h}^t \wedge \neg R_0^t \wedge ... \wedge R_{|x|^h-1}^t]$

E questa espressione è vera se e soltanto se viene assegnato valore vero ad una ed una sola variabile booleana R_i^t con $0 \leq t \leq |x^h| \ e \ 0 \leq i \leq k$.

Insieme N

Siccome la computazione $NT(x)$ accetta in al più $|x|^h$ passi allora utilizza al più $|x|^h$ celle del nastro.
Ed $L_\Gamma \subseteq \{ 0, 1 \}^*$ allora in ogni cella del nastro può trovarsi il carattere 0, 1 o $\square$

Per ogni passo t della computazione ($0 \leq t \leq |x|^h$) e per ogni $0 \leq i \leq |x|^h$ e per ogni $j \in \{ 0, 1, \square \} N$ contiene la variabile booleana N_{ij}^t:

$N = \{ N_{ij}^t : 0 \leq t \leq | x|^h \wedge 0 \leq i \leq k \wedge j \in \{ 1, 0, \square \} \}$.

Assegnando il valore vero alla variabile N_{ij}^t indichiamo che al passo t della computazione nella cella i si trova il carattere j.

Affinché le variabili booleane N^t_{ij} descrivano correttamente i caratteri contenuti in ciascuna cella del nastro al passo t della computazione dobbiamo imporre che siano coerenti, ovvero in ogni cella del nastro ad ogni passo t della computazione può trovarsi uno e un solo carattere, quindi dobbiamo prendere in considerazione le sole assegnazioni di verità che per $0 \leq t \leq |x|^h$ e per $1 \leq i \leq |x|^h$ ad una sola delle $N^t_{i1}, N^t_{i0}, N^t_{i\square}$ sia assegnato il valore **vero**.

Per fare ciò introduciamo $|x|^h + 1$ espressioni $E^0_N, ..., E^{|x|^h}_N$.
Dove E^t_N rappresenta la seguente affermazione:
"Al passo t della computazione su ogni cella del nastro vi è uno e un solo carattere compreso tra 0,1,☐" .
Ma prima dobbiamo introdurre una nuova espressione E^{ti}_j che corrisponde alla seguente affermazione: "al passo t della computazione, nella cella i vi è scritto uno e un solo carattere $j \in \{0, 1, \square\}$".

$$E^{ti}_j = [N^t_{i0} \wedge \neg N^t_{i1} \wedge \neg N^t_{i\square}] \vee [N^t_{i1} \wedge \neg N^t_{i0} \wedge \neg N^t_{i\square}] \vee [N^t_{i\square} \wedge \neg N^t_{i0} \wedge \neg N^t_{i1}]$$

Questa espressione è vera se e soltanto se viene assegnato vero ad una ed una sola delle variabili booleane $N^t_{i0}, N^t_{i1}, N^t_{i\square}$.
Dunque ora possiamo definire E^t_N

$$E^t_N = E^{t1}_j \wedge ... \wedge E^{t|x|^h}_j$$

E^t_N assume il valore vero se e soltanto se al passo t della computazione nella cella i è scritto uno e un solo carattere j compreso tra 0,1,☐.

Rappresentare un generico stato globale

Attraverso le variabili booleane introdotte precedentemente ora siamo in grado di descrivere un generico stato globale. Gli stati globali di NT al passo t sono descritti dalle assegnazioni di verità che rendono soddisfacibile la seguente espressione $S^t = E^t_N \wedge E^t_M \wedge E^t_R$

Infatti un'assegnazione di verità che rende soddisfacibile S^t rappresenta la seguente affermazione:
"La macchina NT si trova in uno e un solo stato al passo t, la testina si trova su una e una sola cella del nastro al passo t e su ciascuna cella del nastro è scritto uno e un solo carattere al passo t della computazione".

Rappresentare una computazione $NT(x)$

Adesso che possiamo rappresentare un generico stato globale attraverso un espressione booleane dobbiamo rappresentare una computazione accettante di NT in $|x|^h$ passi, ossia:

"Esiste un sequenza di al più $|x|^h$ quintuple, che eseguite in successione porta la macchina NT al suo stato di accettazione"

Possiamo dire che $NT(x)$ è una computazione accettante in $|x|^h$ passi se:

- al passo 0 viene eseguita una quintupla
- al passo 1 viene eseguita una quintupla o la macchina NT è in q_A
  ...
- al passo $|x|^h - 1$ viene eseguita una quintupla o la macchina NT è in q_A
- al passo $|x|^h$ la macchina NT è in q_A

Dobbiamo dunque rappresentare sotto espressione booleana la seguente affermazione:
"NT è in q_A o viene eseguita una quintupla".

Sia $\langle q_{i1}, s_{i1}, s_{i2}, q_{i2}, m\rangle$ con $m = 1$ se il movimento della testina viene effettuato a destra, $m = -1$ a sinistra e $m = 0$ se rimane ferma.

Possiamo descrivere attraverso espressione booleana la seguente affermazione: "La quintupla $\langle q_{i1}, s_{i1}, s_{i2}, q_{i2}, m\rangle$ viene eseguita al passo t mentre la testina si trova sulla cella u" nel seguente modo:

$$G^t(u, \langle q_{i1}, s_{i1}, s_{i2}, q_{i2}, m\rangle) = M^t_{i1} \wedge R^t_u \wedge N^t_{us_{i1}} \wedge N^{t+1}_{us_{i2}} \wedge M^{t+1}_{i2} \wedge R^t_{u+m}$$

La seguente espressione booleana rappresenta la seguente affermazione: "Al passi t della computazione la testina è posizionata sulla cella u, la macchina è nello stato q_{i1} e nella cella u viene letto il carattere s_{i1} e al passo $t + 1$ nella cella u vi sarà il carattere s_{i2} la macchina si troverà nello stato q_{i2} e la testina sarà posizionata sulla cella $u + m2$".

Ma la testina potrebbe trovarsi su una delle qualsiasi $|x|^h$ celle del nastro dunque

$$G^t(\langle q_{i1}, s_{i1}, s_{i2}, q_{i2}, m\rangle) = (1, \langle q_{i1}, s_{i1}, s_{i2}, q_{i2}, m\rangle) \vee (2, \langle q_{i1}, s_{i1}, s_{i2}, q_{i2}, m\rangle) \vee ... \vee (|x|^h, \langle q_{i1}, s_{i1}, s_{i2}, q_{i2}, m\rangle)$$

Al passo t la macchina è nello stato q_{i1}, la testina è posizionata su una qualsiasi cella u (con $1 \leq u \leq |x|^h$) e legge il simbolo s_{i1}, e al passo $t+1$ la macchina è nello stato q_{i2}, nella cella u è stato scritto s_{i2} e la testina è stata spostata sulla cella $u+m$

Se l'insieme delle quintuple di NT_Γ è $\{ \langle q_{11}, s_{11}, s_{12}, q_{12}, m_1 \rangle, \langle q_{21}, s_{21}, s_{22}, q_{22}, m_2 \rangle, \dots, \langle q_{h1}, s_{h1}, s_{h2}, q_{h2}, m_h \rangle \}$

allora per esprimere la seguente espressione: "Viene eseguita una quintupla di NT", in forma booleana scriviamo la seguente espressione

$$G^t = G^t(\langle q_{11}, s_{11}, s_{12}, q_{12}, m_1 \rangle) \vee G^t(\langle q_{21}, s_{21}, s_{22}, q_{22}, m_2 \rangle) \vee ... \vee G^t(\langle q_{h1}, s_{h1}, s_{h2}, q_{h2}, m_h \rangle)$$

Quindi per rappresentare in forma booleana l'affermazione:
"NT è in q_A o esegue una quintupla" ,ricordando che q_1 corrisponde allo stato di accettazione di NT, possiamo usare la seguente espressione:

$$G^t \vee M_1^T$$

Rappresentare la configurazione iniziale della macchina

Rappresentiamo ora la configurazione iniziale della macchina ovvero rappresentiamo la seguente affermazione:
"x (e solo x) è scritto sul nastro di input, la macchina si trova nel suo stato inziale e la testina è posizionata sulla cella contenente il primo carattere di x" attraverso un espressione booleana:

Assegniamo vero alla variabile M_0^0, vero alla variabile R_1^0.

Sia x (e solo x) scritta sul nastro di input e sia $x = x_1...x_n$ allora per ogni $i = 1, ..., n$ assegno alla variabile $N_{ix_i}^t$ il valore vero e per ogni $i = n+1, ..., p(n)$ assegno alla variabile $N_{i\square}^t$ il valore vero.

Quindi lo stato globale iniziale della computazione $NT(x)$ è completamente descritto dalla seguente espressione:

$$H = M_0^0 \wedge R_1^0 \wedge N_{1x_1}^0 \wedge N_{2x_2}^0 \wedge \dots \wedge N_{nx_n}^0 \wedge N_{n+1\square}^0 \wedge N_{n+2\square}^0 \wedge \dots \wedge N_{p(n)\square}^0$$

Dunque se H è vera allora esiste un assegnazione di verità in M, N, R che descrive la configurazione iniziale della macchina ovvero x (e solo x) scritto sul nastro di input, la testina posizionata sulla cella contenente il primo carattere di x e la macchina che si trova nel suo stato iniziale

Se S^t è vera allora vengono assegnati alle variabili che descrivono lo stato globale al passo t valori di verità consistenti

Se G^t è vera allora vengono assegnati valori di verità che corrispondo all'esecuzione di una quintupla di NT al passo t

Ricordando che M_1^t è la variabile che descrive se al passo t la macchina è in q_A allora siccome una computazione NT accettante in $|x|^h$ passi se:

- al passo 0 viene eseguita una quintupla
- al passo 1 viene eseguita una quintupla o la macchina NT è in q_A
  ...
- al passo $|x|^h - 1$ viene eseguita una quintupla o la macchina NT è in q_A
- al passo $|x|^h$ la macchina NT è in q_A

Allora $E(x) = H \wedge S^0 \wedge (M_1^0 \vee G^0) \wedge ... \wedge S^{|x|^h - 1} \wedge (M_1^{|x|^h - 1} \vee G^{|x|^h - 1}) \wedge S^{|x|^h} \wedge M_1^{|x|^h}$.

$x \in L_\Gamma \implies E(x)$ è soddisfacibile

Se $x \in L_\Gamma$ allora esiste una computazione $NT(x)$ che in $|x|^h$ passi termina nello stato di accettazione. Ossia esiste una successione di stati globali $SG_0, ..., SG_u$ con $u \leq |x|^h$ e una successione di quintuple $\langle q_{t1}, s_{t1}, s_{t1}, q_{t2}, m_t \rangle$ con $0 \leq t \leq u-1$, tali che:

- SG_0 è lo stato globale iniziale di NT, ove x è scritto sul nastro di input, la macchina si trova sul suo stato iniziale e la testina è poisizionata sulla cella contenente il primo carattere di x
- per $0 \leq t \leq u-1$, lo stato interno di SG_t è q_{t1} il simbolo letto dalla testina è s_{t1} e SG_{t+1} è lo stato corrispondente all'esecuzione della t-esima quintupla della sequenza a partire da SG_t
- Lo stato interno di SG_u è q_A

a partire da ciò, costruiamo un'assegnazione di verità a che soddisfa $E(x)$.

1. Partiamo da SG_0:
   poniamo $a(M_0^0) = vero$ e $a(R_1^0) = vero$ e per ogni $0 \leq j \leq |x|$ se il j-esimo bit di x è 1 poniamo $a(N_{j1}^0) = vero$, se 0 poniamo $a(N_{j0}^0) = vero$ inoltre per ogni $|x| + 1 \leq j \leq p(|x|)$ poniamo $a(N_{j\square}^0) = vero$.

a assegna falso a tutte le variabili in M^0, R^0, N^0, pertanto a soddisfa H e S^0

2. Continuiamo con $SG_1, ..., SG_u$, definiamo $a(M_i^t), a(R_i^t), a(N_{ij}^t)$ usando SG_t come abbiamo fatto per il punto 1. Questo garantisce che, per ogni $t = 1, \ldots, u$:
esiste un solo i per cui $a(M_i^t) = vero$ ed esiste un solo i per cui $a(R_i^t) = vero$ e per ogni $0 \leq j \leq |x|^h$ esiste un solo i per cui $a(N_{ji}^t) = vero$.
e quindi per ogni $t = 1, \ldots, u$, a soddisfa S^t
3. Per ogni $0 \leq t \leq u - 1$ può essere eseguita la t-esima quintupla della sequenza quindi a soddisfa G^t
4. Lo stato globale SG_u è nel suo stato interno q_A. Dunque poniamo $a(M_1^u) = vero$. Dunque a soddisfa l'espressione $(M^u \vee G^u)$, anche se non viene eseguita alcuna quintupla al passo u.
5. Per $t > u$ e per ogni $i = 0, ..., k$ poniamo $a(M_i^t) = a(M_i^u)\ e\ a(R_i^t) = a(R_i^u)\ e\ a(N_{i0}^t) = a(N_{i0}^u), a(N_{i1}^t) = a(N_{i1}^t), a(N_{i\square}^t) = a(N_{i\square}^t)$.
Quindi a soddisfa $S^t\ e\ M_1^t$

Dunque a soddisfa $E(x)$

$E(x)$ è soddisfacibile $\implies x \in L_\Gamma$

Supponiamo ora che esista un'assegnazione di verità a per le variabili in N, M, R che soddisfa $E(x)$.
Ossia a soddisfa $H, S^{|x|^h} e M^{|x|^h}$
e per $t = 0, ..., |x|^h - 1$ a soddisfa S^t ovvero descrive in modo consistente lo stato globale in cui si trova la computazione al passo t.
Dunque a descrive una successione di stati globali $SG_0, ..., SG_{|x|^h-1}$.

poichè a soddisfa H, allora $a(S^0)$ descrive lo stato globale iniziale dove x è scritto sul nastro di input, la macchina si trova nel suo stato iniziale, e la testina si trova sulla cella contenente il primo carattere di x.

Siccome a soddisfa per $t = 0, ..., |x|^h - 1$ $(G^t \vee M_1^t)$ allora o viene eseguita una quintupla al passo t e dunque $a(G^t) = vero$ o ,al passo t,la macchina e nel suo stato di accettazione e dunque $a(M_1^{|x|^h}) = vero$

Notiamo bene che è impossibile che $a(G^t) = vero$ e $a(M_1^{|x|^h}) = vero$ contemporaneamente perchè se la macchina si trova nel suo stato di accettazione non può più eseguire alcuna quintupla perché la computazione è terminata.

Dunque per $t = 0, ..., |x|^h - 1$ se $a(G^t) = vero$ allora viene eseguita una quintupla che fa passare dallo stato SG_t allo stato globale SG_{t+1}.

D'altra parte a soddisfa $M_1^{|x|^h}$ dunque esiste un indice h tale $a(M_1^h) = vero$ e per ogni $t > h$ si ha che $a(M_1^t) = vero$.

Dunque sia $u = 0, ..., |x|^h$ il primo intero per cui $a(M_1^u) = vero$, allora per $t = 0, ..., u - 1$ $a(G^t) = vero$ e quindi viene eseguita una quintupla che fa passare da SG_t a SG_{t+1}.

Dunque la successione di stati globali $\langle SG_1, ..., SG_u \rangle$ corrisponde a una computazione accettante di $NT(x)$.
Dunque $x \in L_\Gamma$

Teorema

Sia Γ_0 un problema in $\mathbf{NP}$. Se esiste un problema $\mathbf{NP}$-completo riducibile a Γ_0, allora Γ_0 è $\mathbf{NP}$-completo

dim

Sia Γ_1 un problema $\mathbf{NP}$-completo tale che $\Gamma_1 \leq \Gamma_0$. Poiché $\Gamma_1 \leq \Gamma_0$, esiste una funzione $f_{10} : I_{\Gamma_1} \to I_{\Gamma_0}$ tale che $f_{10} \in \mathbf{FP}$ e per ogni $x \in I_{\Gamma_1}[x \in \Gamma_1 \iff f_{10}(\Gamma_0) \in \Gamma_0]$

Poiché Γ_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 $f_{21} : I_{\Gamma_2} \to I_{\Gamma_1} : f_{21} \in \mathbf{NP} \ \wedge \ \forall x \in I_{\Gamma_2}[x \in \Gamma_2 \iff f_{21}(x) \in \Gamma_1]$.

Mostriamo ora che la composizione f_{21} e f_{10} è una riduzione polinomiale da Γ_2 a Γ_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 Γ_2 a Γ_0.

Poichè $f_{21} \in \mathbf{FP}$, 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, $|f21(x)| \leq |x|^k$.
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 $dtime(T10, x) \leq |x|^h$. 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 $x \in I_{\Gamma_2}$`e scritto sul nastro di lavoro e, a questo punto:

1. Viene eseguita la computazione $T_{21}(x)$ scrivendo il suo output $f_{21}(x)$ sul nastro di lavoro
2. 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 Γ_2 è un qualunque problema in $\mathbf{NP}$, questo prova che ogni problema in $\mathbf{NP}$ è riducibile polinomialmente a Γ_0. Dall'appartenenza di Γ_0 a $\mathbf{NP}$ segue che Γ_0 è $\mathbf{NP}$-completo.