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 $\implies \exists T : \forall x \in \Sigma^*[o_T(x) = q_A \iff x \in L \land o_T(x) = q_R \iff x \in L^c]$

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

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

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 $\implies \exists T : \forall x \in \Sigma^*[o_T(x) = q_A \iff x \in L \land o_T(x) = q_R \iff x \notin L]$

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

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.
Nel caso i non sia una codifica di alcuna macchina di Turing, allora poiché l'insieme delle quintuple di una qualsiasi macchina di Turing è totale e le computazioni con input che non rispettano le specifiche non terminano, allora $U(i, x)$ non termina.

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 $f : \mathbb{N} \to \mathbb{N}$ 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

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

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

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

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

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

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

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

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

Teorema

Sia $f : \mathbb{N} \to \mathbb{N}$ 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 $f : \mathbb{N} \to \mathbb{N}$ 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 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

$EXPTIME = \bigcup_{k \in \mathbb{N}} DTIME[2^{n^k}]$: 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 $L \in \mathbf{P} \implies \exists \text{ una macchina di Turing deterministica } T \text{ che decide L e una costante } k \in \mathbb{N} \ \land \ \forall x \in \Sigma^*[dtime(T, x) \leq |x|^k)]$.

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 $\implies \exists \text{ una macchina di Turing trasduttore } T_f \text{ e una costante } h \in \mathbb{N} : \forall x \in \Sigma_1^*[\text{ calcola } f(x) \text{ e lo scrive sul nastro di output } \land \ dtime(T_f, x) \leq |x|^h]$.

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