Appunti di Linguaggi e Computabilità

Domande e Risposte

1) Conversione da ER a ε – NFA

Sia f: ER → ε – NFA, una funzione che trasforma un'espressione regolare in un ε – NFA. L'automa che viene generato ha un solo stato finale F = {q_f} con q_0 (stato iniziale) senza archi entranti e q_f senza archi uscenti. La funzione viene definita per induzione strutturale:

Caso Base

→ Ø Є ER: nessun arco (q_0 --- q_f);

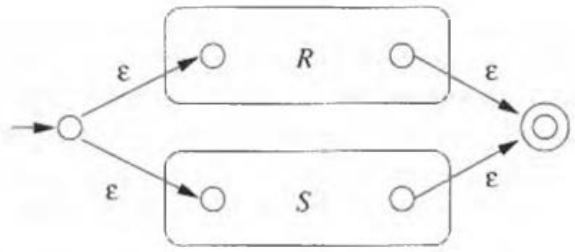
→ ε Є ER: riconosce la ε – mossa (q_0 → ε → q_f);

→ per ogni a Є ∑, con a Є ER: riconosce a (q_0 → a → q_f).

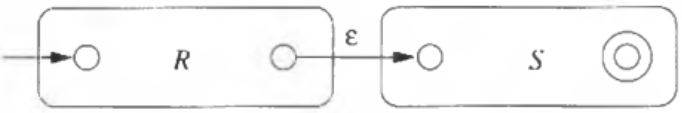
Ipotesi

se E1 e E2 sono espressioni regolari, i rispettivi automi si compongono:

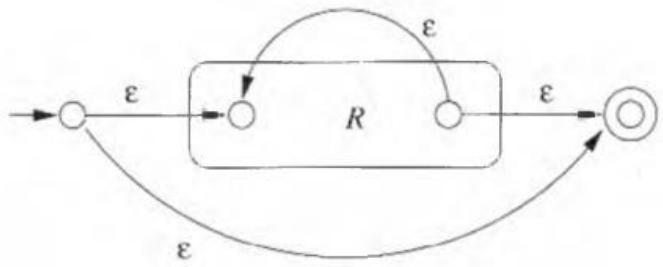
→ E1 + E2 Є ER



→ E1E2 Є ER



→ E1* Є ER



2) Teorema di Kleen

Se L_1 e L_2 sono regolari, allora:

→ L_1L_2 è regolare;

→ L_1 + L_2 è regolare;

→ L_1 U L_2 è regolare;

→ L_1*, L_2* è regolare;

→ L_1 & L_2 è regolare;

3) Complemento di un Linguaggio

Sia ¬L = {w Є Σ*, w non appartiene ad L}.

Se L è regolare, se esiste A: L(A) = L, allora esiste ¬A: L(¬A) = ¬L, quindi anche ¬L è regolare.

Esempio: L, L^R = {w^R: w Є L & w^R è trascritta da destra verso sinistra}.

4) Prodotto di Automi

Siano A_1, A_2 due automi, il prodotto di automi viene definito mediante la quintupla $A_1 \times A_2 = (Q, \Sigma, \delta, q_0, F)$, in cui:

→ $Q = Q_1 \times Q_2$;

→ $q_0 = \langle q_{01}, q_{02} \rangle$;

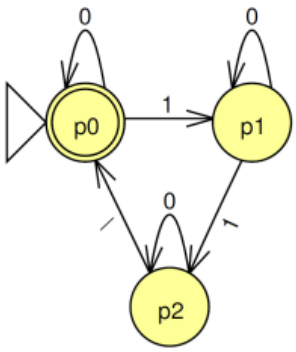
→ $F = F_1 \times F_2$;

→ $\delta(Q, A) = \delta(\langle q_1, q_2 \rangle, a) = \langle \delta_1(q_1,a), \delta_2(q_2,a) \rangle$;

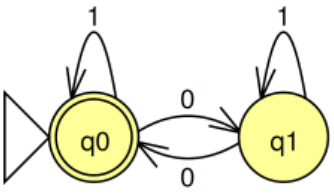
→ $L(A1 \times A2) = \{w \in \Sigma^* \mid \delta\hat{}(\langle q_{01}, q_{02} \rangle \& F \neq \emptyset)$.

Esempio:

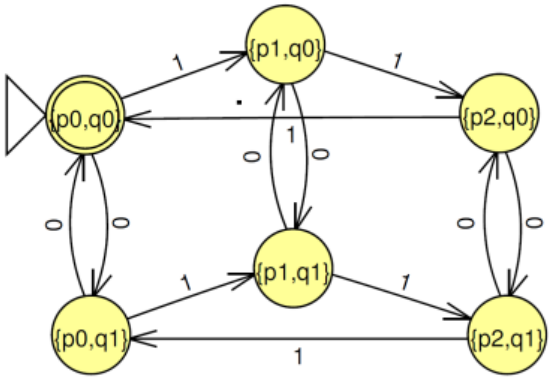
A1 =



A2 =



Il prodotto di automi A1 x A2



5) Da DFA a ER

Dato un DFA, è possibile tradurre tale DFA in una espressione regolare mediante il seguente procedimento:

→ convertire le etichette dei grafi (0, 1 = 0 + 1);

→ eliminare gli stati q_i : q_i != q_0 e q_i != q_f e per eliminare uno stato è necessario concatenare le etichette prestando attenzione alle * sui cicli;

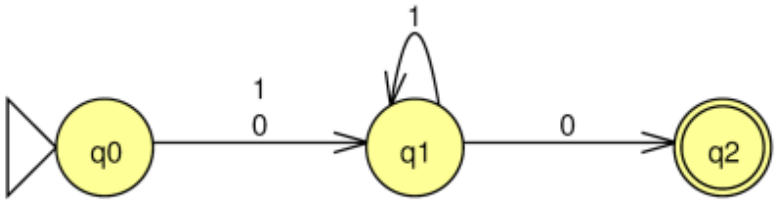
→ per ogni q_i costruire A_i per eliminazione: $Q_i = \{q_0, q_f\}$

→ ottenere quindi n automi con un solo stato iniziale e finale e per ogni i: Ai → Ei;

→ costruire $E_R = E_1 + \dots + E_n$.

Esempio 1

Costruiamo l'espressione regolare che denota il linguaggio accettato dal DFA mostrato in figura



Il primo passaggio da fare è sostituire le etichette delle transizioni dell'automa con delle espressioni regolari.

Ciò descrive la modifica delle etichette:

1) L'etichetta che parte da q0 e va in q1 diventa 0+1.

2) L'etichetta che parte da q1 e va in q1 diventa 1.

3) L'etichetta che parte da q1 e va in q2 diventa 0.

6) Due automi possono accettare lo stesso linguaggio?

Due automi accettano lo stesso linguaggio solo se sono equivalenti.

Le caratteristiche riguardo alla relazione di equivalenza tra gli automi sono:

→ A1 ~ A2 è una relazione di equivalenza riflessiva, simmetrica e transitiva;

→ possibilità di suddividere l'insieme degli automi in partizioni, ovvero classi di equivalenza;

→ per ogni espressione regolare, esiste la classe di automi che riconosce il linguaggio.

L'efficienza di un DFA è misurata sul minor numero possibile di stati. Esiste un unico automa minimo, per ogni classe di equivalenza.

7) Quali sono le proprietà della relazione di equivalenza?

Formulazione matematica

$\forall w \in \Sigma^*, \delta_1\hat{}(q_{01}, w) \in F_1 \rightarrow \delta_2\hat{}(q_{01}, w) \in F_2 \leftarrow \rightarrow L(A_1) = L(A_2)$

Se si considerano anche gli stati iniziali, viene accettato lo stesso linguaggio.

Le proprietà della relazione di equivalenza sono:

→ riflessiva, uno stato è equivalente a se stesso;

→ simmetrica, per definizione ←→;

→ transitiva: se p ~ q e q ~ s allora p ~ s.

Dimostrazione: Supponiamo che p /~ s per assurdo, esiste $w \in \Sigma$: $\delta\hat{}(q, w) \in F$ & $\delta\hat{}(s, w) /\in F$ o viceversa. La relazione p ~ q, w è accetata da p, $\delta(q, w) \in F$ poiché in relazione, ma è assurdo che q ~ s, poiché w non è in s.

8) Automa minimo: definizione

Dato $A = (Q, \Sigma, \delta, q_0, F)$, si dice $\min(A) = (Q/\sim, \Sigma, \delta\min, [q_0]_\sim, F/\sim)$ se:

→ $\delta\min([q]_\sim, a) = [\delta(q, a)]/\sim$;

→ min(A) è unico, a meno di isomorfismi.

9) Verifica di equivalenza

Per poter verificare più facilmente l'equivalenza tra due stai è necessario dover verificare quando due stati non sono equivalenti:

→ $p \not\sim q$: esiste $w \in \Sigma^*$: $\delta\hat{}(p, w) \in F$ & $\delta\hat{}(q, w) /\in F$ o viceversa [distinguibilità];

→ p ~ q ←→ not (p !~ q).

La relazione $p \not\sim q$ ha le seguenti proprietà:

→ simmetrica: $p \not\sim q \leftarrow\rightarrow q \not\sim p$

→ non riflessiva, perché uno stato è non distinguibile da se stesso;

→ transitiva non deducibile: $p \not\sim q, q \not\sim s \rightarrow p \not\sim s$

10) Algoritmo Riempi – Tabella

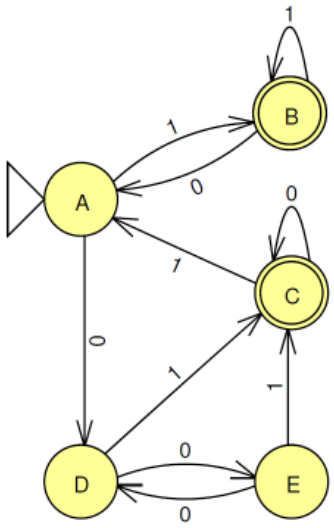
L'algoritmo riempi tabella è definito per induzione:

→ base: $p \in F, q /\in F \rightarrow p \not\sim q$, poiché $\delta(p, \varepsilon) \in F, \delta(q, \varepsilon) /\in F$ (riempire le caselle finali/non finali);

→ passo: se $r \not\sim s$ & esiste $a \in \Sigma$: $\delta(p, a) = r$ & $\delta(q, a) = s \rightarrow p \not\sim q$

La tabella è tagliata alla diagonale perché !~ è una relazione simmetrica, non riflessiva e non transitiva.

Esempio:



Step 1

Scegli una o più alternative:

- [ ] a. B e C
- [x] b. C e D
- [x] c. A e B
- [ ] d. A ed E
- [ ] e. D ed E
- [x] f. C ed E
- [x] g. B ed E
- [x] h. A e C
- [ ] i. A e D
- [x] j. B e D

Step 2

Creiamo la tabella di distinguibilità:

B	x	-	-	-
C	x		-	-
D		x	x	-
E		x	x	
	A	B	C	D

Prendiamo, ad esempio, gli stati A e C (distinguibili o meno?). Quale è il simbolo di input che porta in questi due stati partendo da due stati che non sappiamo ancora se siano o meno distinguibili?

Risposta: 0

Step 3

La nostra tabella di distinguibilità ora sarà:

B	x	-	-	-
C	x	x	-	-
D		x	x	-
E		x	x	
	A	B	C	D

Prendiamo, ad esempio, gli stati B e C. Quale è il simbolo di input che porta in questi due stati partendo da due stati che non sappiamo ancora se siano o meno distinguibili?

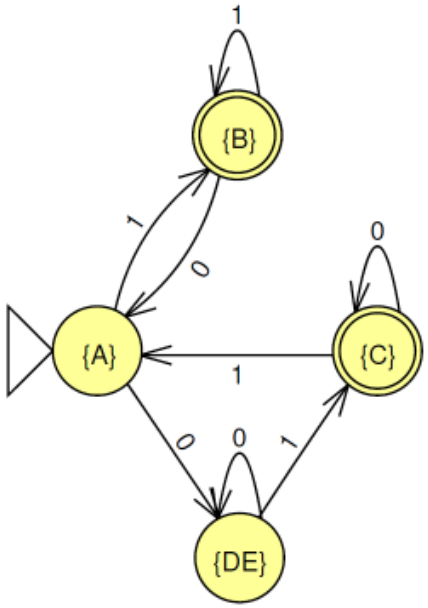
Risposta: 1

Step 4

Tabella

B	x	-	-	-
C	x	x	-	-
D	x	x	x	-
E	x	x	x	
	A	B	C	D

Automa minimo



11) Quando due automi accettano lo stesso linguaggio?

Per stabilire se due automi A_1 e A_2 accettano lo stesso linguaggio, cioè se $L(A_1) = L(A_2)$, ci sono due tecniche:

→ se $min(A_1)$ è isomorfa a $min(A_2)$ → $L(A_1) = L(A_2)$;

→ se $L(A_1) = L(A_2)$ sse $q_{01} \sim q_{02}$, per ogni $w \in \Sigma^*$ $\delta_1(q_{01}, w) \in A_1 \leftrightarrow \delta_2(q_{02}, w) \in F_2$. Non funziona con gli NFA, vanno trasformati in DFA.

12) Pumping Lemma

Il Pumping Lemma è una proprietà che viene utilizzata per dimostrare che determinati linguaggi non sono regolari. Sia L un linguaggio regolare, dunque esiste una costante n (dipendente da L): per ogni $w \in L$ con $|w| >= n$, w può essere scomposta come $w = xyz$ tali che:

→ $y != \varepsilon$;

→ $|xy| <= n$;

→ per ogni $k >= 0$, anche $xy^kz \in L$.

Esempio:

Dimostrare che il linguaggio $L = \{w = a^ncb^m \mid m = n*2\}$ non è regolare.

Dimostrazione:
Suppongo per assurdo L regolare, allora per il Pumping Lemma esiste un numero k tale che per ogni w che appartiene a L, con $|w| \geq k$, posso scomporre w = xyz tale che: $|xy| \leq k$, $y \neq \varepsilon$ e per ogni $i \geq 0$ la stringa xy^iz appartiene a L.
Presa $w = a^k cb^{k*2}$ la stringa xy deve essere un prefisso di a^k (perchè $|xy| \leq k$) e $y = a^h$ per un qualche $h \geq 1$. Se i = 0, allora $xz = a^{k-h} cb^{k*2}$ deve appartenere a L: questo contraddice l'ipotesi perché se $h \geq 1$ allora $(k - h) * 2 \neq k*2$.

13) Automi a Pila (PDA)

Si dice automa a pila (PDA), un ε – NFA con una pila associata. Formalmente un automa a pila è una settupla $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, tale che:
→ Q: insieme finito e non vuoto di stati;
→ ∑: alfabeto dei simboli in input;
→ Γ: alfabeto dei simboli di stack;
→ $q_0 \in Q$: stato iniziale;
→ $Z_0 \in \Gamma$: simbolo inizialmente presente nello stack;
→ F sottoinsieme Q: insieme degli stati finali.
→ δ: $Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \rightarrow 2^{Q \times \Gamma^*}$ tale che:
$\delta(q, a, b) = \{(p, b)\}$ mantengo la b
$\delta(q, a, b) = \{(p, cb)\}$ push di c
$\delta(q, a, b) = \{(p, \varepsilon)\}$ tolgo la b
Essendo non deterministico ci possono essere delle scelte.

14) Descrizioni istantanee: definizione

Una configurazione è una tripla ID(q, w, γ) dove:
→ q è lo stato attuale;
→ $w \in \Sigma^*$ è l'input residuo;
→ $\gamma \in \Gamma^*$ è il contenuto attuale della pila.
Sia $P \in PDA$, supponiamo che $(p, a) \in \delta(Q, a, X)$, allora per ogni $w \in \Sigma^*$ & per ogni $\beta \in \Gamma^*$ vale che $(q, aw, X\beta) \vdash_P (p, w, a\beta)$. Formalmente:
→ $\vdash_P$ definisce una mossa del PDA;
→ $\vdash$ è una relazione binaria tra coppie di configurazione;
→ $\vdash^*$ per induzione:
a) base: $I \vdash^* I$ per ogni ID I;
b) passo: $I \vdash^* J$ se esiste ID K tale che $I \vdash K$ & $K \vdash^* J$

15) Teoremi sulle descrizioni istantanee

Teorema 1: se P Є PDA e $(q, x, \alpha) \vdash^* (p, y, \beta)$ allora per ogni w Є Σ* e per ogni γ Є Γ* vale che $(q, xw, \alpha\gamma) \vdash^* (p, yw, \beta x)$.

Teorema 2: se P Є PDA e $(q, xw, \alpha) \vdash^* (p, yw, \beta)$ allora vale $(q, x, \alpha) \vdash^* (p, y, \beta)$.

16) Prefix – free: definizione

Sia L un linguaggio, L è prefix – free se non esistono x e y Є L tali che:

→ x != y;

→ x è prefissa di y.

Esempio 1

$L = \{wcw^R : w \in \Sigma^* = \{0,1\}\}$

w = 01c10 → tolto 10 si ha 01c Є/ L (proprietà verificata)

w = 000c000 → tolto lo 0 si ha 000c00 Є/ L (proprietà verificata)

Esempio 2

{0}* = {ε, 0, 00, 000, …} non è prefix – free. Tolto 0 si ha che w appartiene nuovamente a {0}*.

17) Quali sono i linguaggi accettati dai PDA?

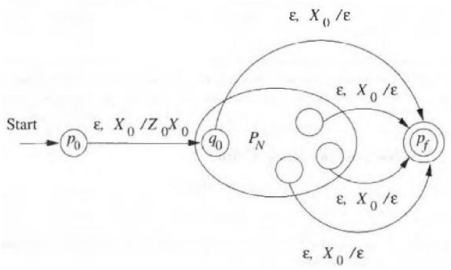
I linguaggi accettati dai PDA sono i linguaggi generati dalle context free grammar (CFG). Esistono due tipologie di accettazione:

→ accettazione per stato finale: sia P Є PDA, il linguaggio accettato da P per stato finale è $L(P) = \{w \in \Sigma^* : (q_0, w, Z_0) \vdash^* (q, \varepsilon, \alpha)\}$ con q Є F e α Є Γ*;

→ accettazione per pila vuota: sia P Є PDA, il linguaggio accettato da P per pila vuota è $N(P) = \{w \in \Sigma^* : (q_0, w, Z_0) \vdash^* (q, \varepsilon, \varepsilon)\}$ con q Є Q. P non avrà stati finali F. I due automi sono equivalenti e "trasformabili".

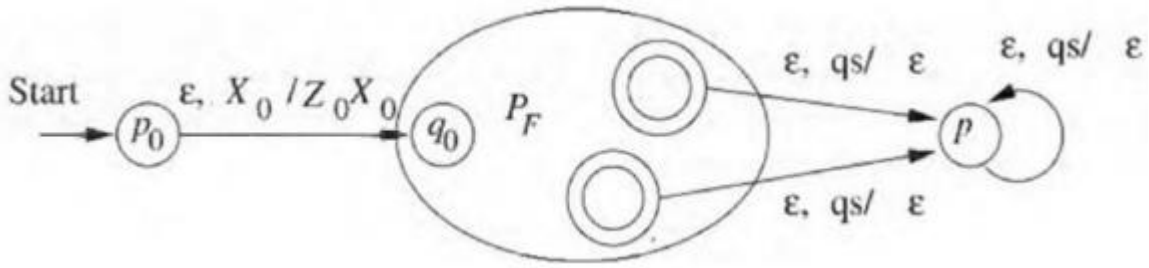
18) Come è possibile trasformare un PDA con accettazione pila vuota a stato finale?

Se $L = N(P_N)$ per un PDA $P_N = (Q, \Sigma, \Gamma, \delta_N, q_0, Z_0)$ allora esiste un PDA PF tale che L = L(PF) con $PF = (Q \cup \{P_0, P_F\}, \Sigma, \Gamma \cup \{x_0\}, \delta_F, P_0, x_0, \{P_F\})$. Si ha un δ_F è uguale alla δ_N, in più ha il caso iniziale e tutti i passi finali.



19) Come è possibile trasformare un PDA con accettazione a stato finale a pila vuota?

Se L = L(PF) per un PDA PF = (Q, Σ, Γ, δ_F, q_0, Z_0, F) allora esiste un PDA PN tale che L = N(PN) con PN = (Q U {P_0, P}, Σ, Γ U {x_0}, δ_F, P_0, x_0).



Se L = N(P) per un DPDA pila vuota, allora L ha una CFG non ambigua.
Se L = L(P) per un DPDA stato finito, allora L ha una CFG non ambigua.
Nota: il simbolo x_0 non appartenente a Γ viene utilizzato sia come simbolo iniziale del PDA con accettazione a stato finale che come segnale che indica che lo stack di PN è stato svuotato. Quando PF vede x_0 alla sommità del proprio stack, sa che PN vuoterebbe lo stack.

20) Come è possibile convertire una Context – Free Grammar in un PDA?

Data G = (V, T, Q, S), si costruisce il PDA P = ({q}, T, V U T, δ, q, S) in cui:
→ VA Є V, δ(q, ε, A) = {(q, β) : A → β è una produzione di G}
→ Va Є T, δ(q, a, a) = {(q, ε)}

21) Definizione di PDA Deterministico

Un PDA = (Q, Σ, Γ, δ, q_0, Z_0, F) è deterministico (DPDA) se:
1) |δ(q, a, x)| <= 1, Vq Є Q, Va Є Σ U {ε}, Vx Є Γ;
2) |δ(q, a, x)| != 0 per qualche a Є Σ, allora |δ(q, ε, x)| = 0.

22) Accettazione Regolari

Se L è regolare allora esiste un PDA non deterministico tale che L = L(P_F).
Inoltre esiste un PDA non determinstico tale che L = N(PN) e un PDA è un ε – NFA che utilizza una pila (infatti accetta i regolari).

23) Teorema sull'accettazione dei regolari

Se L è regolare allora esiste un DPDA PF tale che L = L(PF).
Attenzione: non è detto che esista un DPDA PN tale che L = N(PN). Quindi per pila vuota fanno meno che per stato finale.

24) Teorema 1 e Teorema 2

Teorema 1: Se L = N(P) per un PDA a pila vuota, allora ha una CFG non ambigua.

Teorema 2: Se L = L(P) per un DPDA a stato finale, allora L ha una CFG non ambigua. Non tutti e soli, non è una caratterizzazione.

25) Perchè i DPDA che accettano per pila vuota possono accettare solo linguaggi che sono prefix free (che hanno la PP)?

Il seguente linguaggio L = $\{b^n \mid n \geq 0\}$ non ha la PP. Quindi non può essere accettato per pila vuota. Supponiamo per assurdo che esiste un DPDA che accetta per pila vuota in grado di riconoscere il linguaggio L. Si assume quindi che $w \in L$ e $wv \in L$. Dal momento che w è accettato da M, esiste un percorso computazionale da (q_0, w, S_0) a $(q_k, \varepsilon, \varepsilon)$. Dal momento che M è deterministico, l'unica possibile computazione per l'input wv è $(q_0, w, S_0) \Rightarrow^* (q_k, v, \varepsilon)$. Ma a quel punto M ha svuotato lo stack e non può andare oltre (qk, v, ε) e quindi non accetta vw. Si ha una contraddizione e quindi i DPDA per pila vuota possono accettare solo linguaggi che sono prefix free.

26) Macchina di Turing: definizione formale

Si dice macchina di Turing la settupla $MdT = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$, dove:

→ Q: insieme finito di stati;
→ Σ: alfabeto finito di simboli di input;
→ Γ: alfabeto finito di simboli del nastro;
→ δ: $Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ è la funzione di transizione (deterministico);
→ q_0 appartiene a Q, ovvero lo stato finale;
→ B appartiene a Σ, Γ simbolo di blank;
→ F iniseme degli stati finali
→ L(M) = {w | w accettata da M}

27) Ricorsivamente Enumerabile

Si dice RE se esiste una Macchina di Turing M tale che L = L(M). Sono linguaggi di tipo 0.

28) Descrizioni Istantanee delle MdT

Sia ID una descrizione istantanea della seguente forma:
$\langle q_n, Bx_1x_2\ldots x_{i-1} | x_i x_{i+1} \ldots x_n B \rangle$
tale descrizione istantanea ha le seguenti proprietà:

→ deve essere che Q & Γ = insieme vuoto;
→ |→ relazione binaria tra ID, ovvero I |→ J, se δ lo prevede;
→ |→* definito per induzione:

a) base: I |→* I per ogni ID I;
b) passo: I |→* J se esiste ID K tale che I |→ K & K |→* J

29) Linguaggio accettato dalla Macchina di Turing

Sia M una MdT, allora il linguaggio accettato dalla macchina è
L(M) = {w appartente a Σ* | <q0, w> |→* apβ, con p appartenente a F & α, β appartenenti a Γ *}.

30) Estensioni delle Macchina di Turing

Una macchina di Turing può essere:
→ multinastro: controllo finito con numero finito di nastri, ognuno con una testina indipendente con k fissato. Può essere simulata da una MdT a singolo nastro con un partizionamento simulato;
→ non deterministiche: più rami di computazione e può essere simulata da una macchina di Turing deterministica, sul quale nastro si accodano i nodi dell'albero di computazione così come sono visitati (il nastro è visto come coda). La visita è per livelli e non per profondità.

31) Differenza tra problemi di decisione e non di decisione (P e NP)

Un problema si dice di decisione (decidibile) se è risolvibile con una DTM in un numero di passi polinomiali e dato in input un problema, è in grado di rispondere "si" se è risolvibile, "no" altrimenti. Un problema si dice non di decisione (indecidibile) se è risolvibile con una NTM e P è sottoinsieme di NP. Non è possibile dimostrare che P != NP.

32) Restrizioni delle MdT

Ogni linguaggio accettato da Mdt M2, è accettato anche da una Mdt M1, con i seguenti vincoli:
→ la testina di M1 non va mai a sinistra della posizione iniziale;
→ M1 non scrive mai un B, ma può usare B'.

33) Macchina Multi – Stack e teoremi relativi

Si dice macchina multi – stack, una macchina visibile come un DPDA + k pile. Essa è in grado di accettare linguaggi riconosciuti dalle MdT ma il nastro viene spezzato in due pile.
Teorema 8.14: Ogni linguaggio ricorsivamente enumerabile è accettato da una macchina a 3 contatori e se L è RE, allora è accettato da un DPDA a due stack.
Teorema 8.15: Ogni linguaggio ricorsivamente enumerabile è accettato da una macchina a 2 contatori.

34) Codifica della MdT (come stringa binaria)

Data $MdT = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ dove:

→ Q: insieme finito di stati;

→ Σ: alfabeto finito di simboli di input;

→ $\Gamma = \{x_1, x_2, x_3\}$ con $x_1 = 0$, $x_2 = 1$, $x_3 = B$;

→ indichiamo L = D1, R = D2;

→ δ: $Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ è la funzione di transizione (deterministico) che viene codificata nel seguente modo:

1 – Sia $\delta(q_i, x_j) = (q_k, x_l, D_m)$ che diventa la quintupla $(q_i, x_j, q_k, x_l, D_m)$.

2 – Tale quintupla $(q_i, x_j, q_k, x_l, D_m)$ viene convertita in (i, j, k, l, m).

3 – La codifica sarà quindi $0^i10^j10^k10^l10^m$. Ogni caso della delta può essere codificato.

→ La codifica di δ con i casi $c_1, c_2, \ldots, c_n = c_1 11 c_2 \ldots 11 c_n$ con 1 come separatore.

→ Se cod(M) = w, se $w = w_i$, allora $M_i = M$ e $L(M_i) = w_i$.

35) Linguaggio di Diagonalizzazione (Ld)

Il linguaggio diagonale Ld è l'insieme delle stringhe binarie wi tali che wi non appartengono a L(Mi) ovvero la macchina con lo stesso indice. Ciò significa che la macchina non accetta la sua codifica binaria ed è non ricorsivamente enumerabile (non esiste una macchina di Turing che accetta tale linguaggio). Per dimostrare che tale linguaggio è non ricorsivamente enumerabile si effettua il complemento della diagonale principale della matrice che descrive la funzione caratteristica del linguaggio per conoscere le stringhe appartenenti al linguaggio diagonale e, procedendo in questa maniera, si ottiene il vettore ottenuto non compare in nessuna riga della matrice. Si ha quindi un assurdo nella dimostrazione.

36) Descrivere il linguaggio ricorsivo

Il linguaggio ricorsivo è un sottoinsieme dei linguaggi ricorsivamente enumerabili che data in ingresso una stringa, la macchina risponde sì se tale stringa viene accettata o no se tale stringa non viene accettata. In ogni caso la macchina termina sempre.

37) Descrivere il linguaggio universale

Il linguaggio universale Lu è l'insieme delle stringhe binarie che codificano coppie (M, w), dove M è una MdT con alfabeto binario in input e w è un input da essa accettato.

Esso è un linguaggio:

→ RE in quanto esiste una MdT (macchina universale) che lo accetta.
→ NON Ricorsivo. Se lo fosse, lo sarebbe anche il suo complemento. Da $\overline{Lu}$ si potrebbe costruire una macchina che accetti Ld (riduzione da $\overline{Lu}$ a Ld), ma ciò è assurdo in quanto sappiamo che Ld non è RE. L'ipotesi che Lu sia Ricorsivo è quindi assurda, quindi Lu risulta solo RE.
Lu si relaziona al problema della terminazione in 2 modi:
1. Sia Lu che il problema della terminazione sono indecidibili.
Si può infatti dimostrare che, apportando semplici modifiche alla macchina che accetta l'halting problem, se si supponesse che termina sempre si arriverebbe a un assurdo (T' inverte accetto/non accetto, T'' in caso di non accettazione non termina.
2. L'originale MdT accetta per arresto, non per stato finale.
Posso definire H(M) l'insieme delle stringhe w tali che M si arresta con input w. Considerando il linguaggio contenente le coppie t.c. $w \in H(M)$, so che anche per questo linguaggio deve esistere una MdT che prende in input la coppia e simula il comportamento di M su w.

38) Teoremi sui linguaggi

Una proprietà dei linguaggi RE è un insieme di linguaggi RE, cioè un insieme di macchine di Turing e una proprietà è banale se è l'insieme è vuoto o se è tutto RE. Ogni proprietà non banale dei linguaggi RE è indecidibile.