



POLITECNICO DI MILANO
DIPARTIMENTO DI ELETTRONICA,
INFORMAZIONE E BIOINGEGNERIA

API

ALGORITMI E PRINCIPI DELL'INFORMATICA

BLOCCO APPUNTI

PRIMA PARTE
INFORMATICA TEORICA

Corso del prof. Dino Mandrioli • Assistente esercitatore Achille Frigeri

Leonardo Turchi

A.A. 2014/2015

Enunciati utili

API Introduzione e note

domenica 5 ottobre 2014 17:22

ENUNCIATI VARI DEL TEOREMA DI RICE

NOTA: QUESTI APPUNTI SONO DA
CONSIDERARSI "COME SONO".
L'AUTORE DECLINA OGNI RESPONSABILITA'
CIRCA EVENTUALI ERRORI O MANCANZE CHE
ESSI POTREBBERO CONTENERE.
AA 2014/2015

1)

Teorema di Rice Sia data una qualunque enumerazione delle funzioni calcolabili e sia F un qualunque insieme di funzioni calcolabili; l'insieme $S = \{x \mid f_x \in F\}$ è decidibile se e solo se F (e quindi S) è vuoto o F coincide con l'insieme di tutte le funzioni calcolabili ($S = \mathbb{N}$).

Dim. Supponiamo che S sia ricorsivo e, per assurdo non sia vuoto né F né il complemento di F . Allora, esistono due indici i e j tali che i è il minimo indice in S e j è il minimo indice nel complemento di S . Consideriamo la seguente funzione:

$$c_S(x) = \begin{cases} i & \text{se } x \notin S \\ j & \text{altrimenti} \end{cases}$$

2)

12.4 Teorema. (Rice) Sia $A \subseteq \mathbb{N}$ un insieme estensionale diverso da \emptyset e da \mathbb{N} . Allora A è indecidibile.

Dimostrazione. Sia P_e un programma per la funzione sempre indefinita. Distinguiamo due casi seconda che $e \in A$ o $e \notin A$.

Primo caso. Supponiamo $e \notin A$. Scegliamo $a \in A$ e sia $g(x) = \phi_a(x)$. Definiamo

$$f(x, y) = \begin{cases} g(y) & \text{se } \phi_x(x) \downarrow \\ \uparrow & \text{altrimenti} \end{cases}$$

Per il teorema s-m-n (e il teorema della funzione universale) esiste una funzione calcolabile totale $k: \mathbb{N} \rightarrow \mathbb{N}$ tale che $f(x, y) = \phi_{k(x)}(y)$. Abbiamo $\phi_x(x) \downarrow$ sse $\phi_{k(x)} = g$. Ne segue che $\phi_x(x) \downarrow$ sse $k(x) \in A$. Abbiamo così ridotto K_0 ad A , e quindi A è indecidibile.

Secondo caso. Supponiamo $e \in A$. Sia B il complemento di A . Ragionando come nel primo caso mostriamo che B è indecidibile. Ma allora anche A è indecidibile. \square

3)

Teorema di Rice

Afferma che, per ogni proprietà non banale delle funzioni calcolabili, il problema di decidere quali funzioni soddisfino tale proprietà e quali no, è indecidibile.

Proprietà banale: proprietà che non effettua alcuna discriminazione tra le funzioni calcolabili, cioè che vale o per tutte o per nessuna.

Teorema di Rice. Sia

$$P = \{\langle M \rangle \mid M \text{ è una MdT che verifica la proprietà } \mathcal{P}\}$$

un linguaggio che soddisfa le seguenti due condizioni:

1. L'appartenenza di M a P dipende solo da $L(M)$, cioè

$$\forall M_1, M_2 \text{ MdT tali che } L(M_1) = L(M_2), \langle M_1 \rangle \in P \Leftrightarrow \langle M_2 \rangle \in P$$

2. P è un problema non banale, cioè

$$\exists M_1, M_2 \text{ MdT tali che } \langle M_1 \rangle \in P, \langle M_2 \rangle \notin P$$

P è indecidibile.

- ▶ Ogni proprietà non banale del linguaggio di una MdT è indecidibile.
- ▶ Nota la differenza tra una proprietà di $L(M)$ e una proprietà di M :
 - **Esempio:** $L(M) = \emptyset$ è una proprietà del linguaggio.
 - **Esempio:** "M ha almeno 1000 stati" è una proprietà della MdT.
 - " $L(M) = \emptyset$ " è indecidibile; "M ha almeno 1000 stati" è facilmente decidibile, basta guardare alla codifica di M e contare.

Conseguenze del Teorema di Rice

Non possiamo decidere se una MdT:

- Accetta \emptyset .
- Accetta un linguaggio finito.
- Accetta un linguaggio regolare, ecc.

Esercitazioni

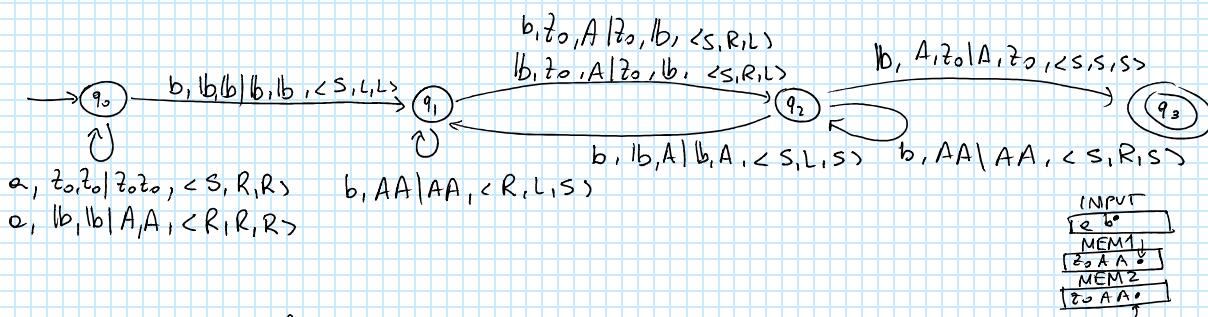
ESERCITAZIONE

martedì 28 ottobre 2014 08:32

MT a K nastri di memoria

- 1 organo di controllo
- 1 nastro di lettura
- K nastri di memoria
- 1 nastro di output

1) DISSEGNARE MT A 2 NASTRI CHE riconosca $L = \{a^m b^m \mid m \geq 1\}$



2) Es. $L = \{a^m b^m \mid m \geq 1\}$ con 3 nastri.

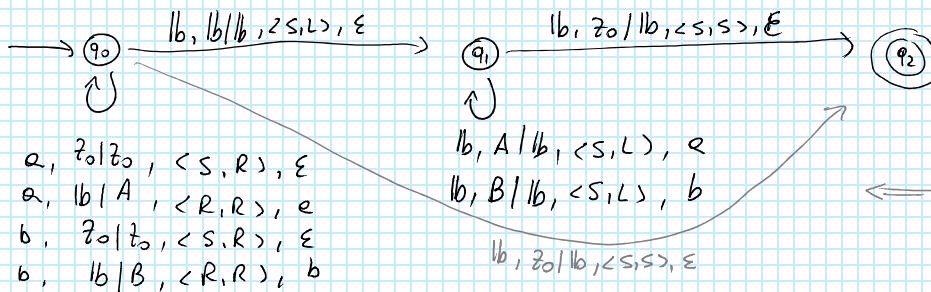
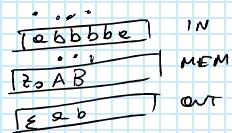
3) Realizzatore con MT a 1 nastro che funziona

$$f : \{a, b\}^+ \rightarrow \{a, b\}^+ \\ w \rightarrow w w^R$$

es. $f(abb) = abb bba$

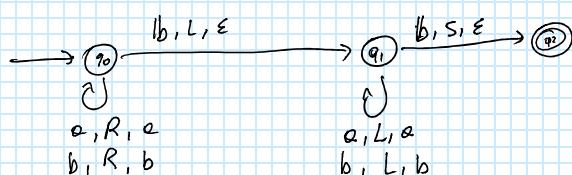
Inoltre nastri di memoria

- nastro input
- nastro output



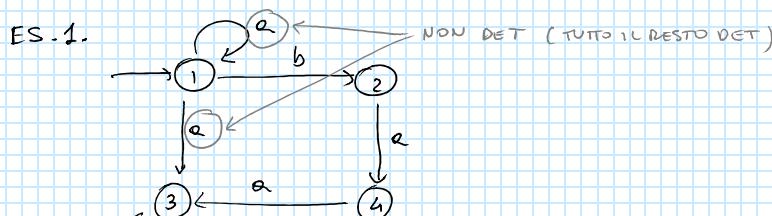
nel caso di $\{a, b\}^* \rightarrow \{a, b\}^*$
no anche

Riconoscibile anche con MT a zero nastri (solo IN e OUT)



MODelli NON DETERMINISTI

ES. AUT. a STATI FINITI NON DET.



$\begin{matrix} \curvearrowleft \\ b \end{matrix}$

$A = (\Sigma, Q, i, F, \tau)$ tutti i sottoinsiemi di A che contengono anche uno stato in F
 $A_{det} = (\Sigma, P(Q), \{i\}, F', \tau')$
 L'gli stati sono dei "super-stati"

$$\forall q \in P(Q), \forall a \in \Sigma \quad \tau'(q, a) = \bigcup_{q_i \in q} \tau(q_i, a)$$

ES. 1 SVOLTO

$$i = \{1\}$$

$$\tau'(\{1\}, e) = \{1, 3\}$$

$$\tau'(\{1\}, b) = \{2\}$$

$$\tau'(\{1, 3\}, e) = \{1, 3, 4\}$$

$$\tau'(\{1, 3\}, b) = \{2, 3\}$$

$$\tau'(\{2\}, e) = \emptyset$$

$$\tau'(\{2\}, b) = \emptyset$$

$$\tau'(\emptyset, e) = \tau'(\emptyset, b) = \emptyset$$

$$\tau'(\{1, 3, 4\}, e) = \{2, 3, 4\}$$

$$\tau'(\{1, 3, 4\}, b) = \{2, 3\}$$

$$\tau'(\{2, 3\}, e) = \{4\}$$

$$\tau'(\{2, 3\}, b) = \{3\}$$

$$\tau'(\{1, 2, 3, 4\}, e) = \{1, 2, 3, 4\}$$

$$\tau'(\{1, 2, 3, 4\}, b) = \{2, 3\}$$

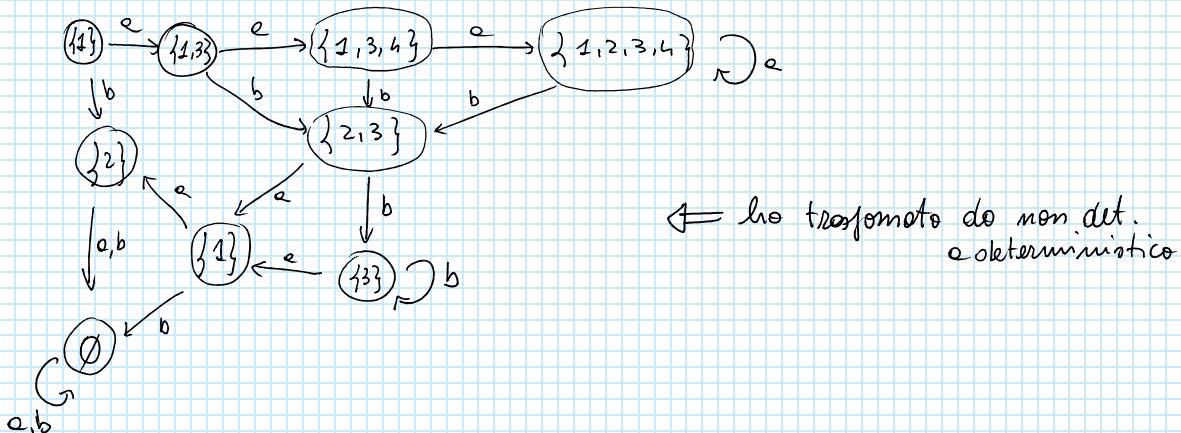
$$\tau'(\{4\}, e) = \{2\}$$

$$\tau'(\{4\}, b) = \emptyset$$

$$\tau'(\{3\}, e) = \{4\}$$

$$\tau'(\{3\}, b) = \{3\}$$

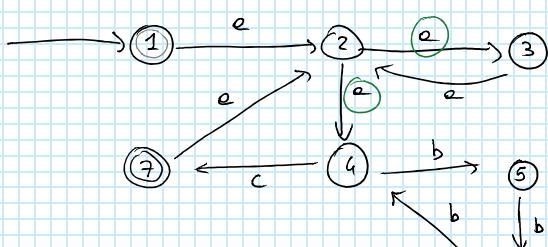
dunque



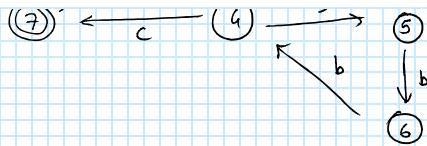
ES 1. costruire DFA che riconosce \mathcal{L}^+

$$\text{alone } \mathcal{L} = \{a^{2m}b^{3m}c \mid m \geq 0, m \geq 0\}$$

(non det e causa di $a = a$)



se metto ① riconosco anche \mathcal{L}^*



DETERMINIZZAZIONE:

$$i = \{1\}$$

$$\gamma'(\{1\}, a) = \{2\}$$

$$\gamma'(\{1\}, b) = \gamma'(\{1\}, c) = \emptyset$$

$$\gamma'(\{2\}, a) = \{3, 4\}$$

$$\gamma'(\{2\}, b) = \gamma'(\{2\}, c) = \emptyset$$

$$\gamma'(\{3, 4\}, a) = \{2\}$$

$$\gamma'(\{3, 4\}, b) = \{5\}$$

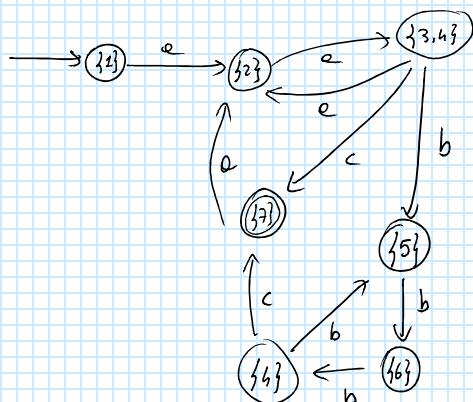
$$\gamma'(\{3, 4\}, c) = \{7\}$$

$$\gamma'(\{5\}, a) = \gamma'(\{5\}, c) = \emptyset$$

$$\gamma'(\{5\}, b) = \dots$$

$$\gamma'(\{6\}, b) = \dots$$

...

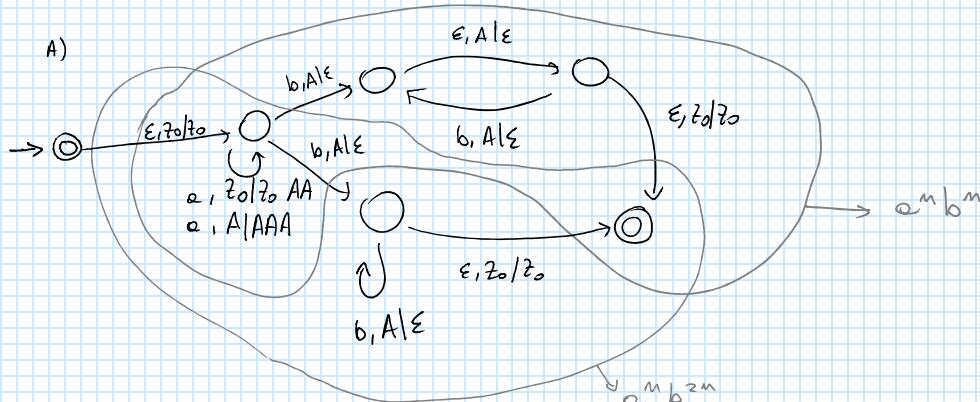


ES 2.

a) Costruire automa a pila non det.

$$L = \{a^m b^n \mid m \geq 0\} \cup \{a^m b^{2m} \mid m \geq 0\}$$

b) Provare che non esiste un a.o.pile det che riconosce L.



B) DIM: esiste un automa a pila det A t.c. $\lambda(A) = L$

→ Siano A_1 e A_2 due copie di A

→ costruire un nuovo automa A_0 sull'alfabeto $\{a, b, c\}$ modificando A_1 e A_2

$$A_1 = (\{a, b\}, Q_1, i_1, \gamma_1, F_1)$$

$$A_2 = (\{a, b\}, Q_2, i_2, \gamma_2, F_2)$$

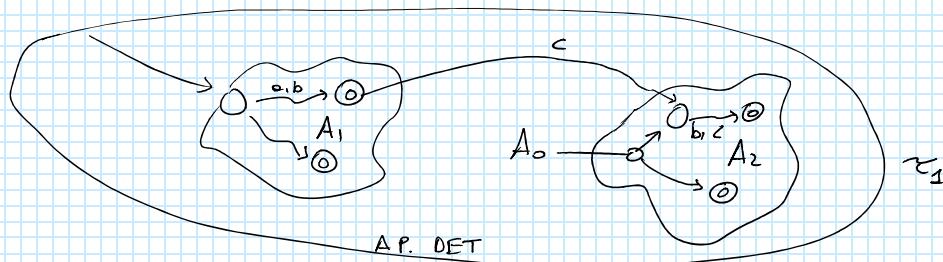
$$A_0 = (\{a, b, c\}, Q_1 \cup Q_2, i_1, \gamma_1, F_2)$$

→ considero tutte le transizioni di γ_1 e γ_2 , elimino le 'c'-transizioni di A_2
 $(q_1, \gamma_1, q') = \gamma_1$

→ rimpicciotto tutte le b-transizioni di A_2 con una c-transizione $(q, b, q') \in \gamma_2 \rightarrow (q, c, q') \in \gamma_2$

→ elimino tutte le c-tr. di A_1 che partono da uno stato finale

→ rimpicciotto $(q_1, b, q_2) \in \gamma_1, q \in F_1 \rightarrow (q_1, c, q'_2) \in \gamma$



$$L(A_0) \text{ sicuramente } L(A_0) \subset (a+b)^* c^*$$

$$a^* \not\subset L(A_0)$$

→ il prefisso di ogni stringa di $L(A_0)$ è trasm. di γ_1 e γ_2 deve essere della forma $a^n b^m a^n b^m$ (transizioni di A_2 poi scatto in transizione da A_1 in A_2 e poi legge solo delle c).

→ Accetto stringhe del tipo $a^n b^m c^i$ oppure $a^n b^m c^i$
 se A_0 accetta $a^n b^m c^i$
 allora A_0 accetta $a^n b^{m+i}$
 se A_0 accetta $a^n b^{m+i}$
 allora A_0 accetta $a^n b^{m+i}$ impossibile

$$\Rightarrow L(A_0) = \{ a^n b^m c^m \mid m \geq 0 \} \quad (\text{non riconoscibile da nessun AP anche NON DET})$$

\Downarrow
ASSURDA L'IPOTESI

ESERCITAZIONE

martedì 4 novembre 2014 08:42

Reti di Petri

$$R = (P, T, IF, OF)$$

↑ ↑
posti transizioni

$$\begin{aligned} IF : P \times T &\rightarrow IN & \text{funzione transizione in ingresso} \\ OF : T \times P &\rightarrow IN & \text{" " " uscite} \end{aligned}$$



NOTA: un arco di peso 2, fa arrivare sempre solo 1 token, ma ne preleva 2 (non sono fuori)

$p_1 \rightarrow 1 \leftarrow \text{numero di token dentro alle rispett. p.}$

$p_3 \rightarrow 2 \leftarrow$

PASSAGGIO MARCATURA:

la transizione è abilitata $\Leftrightarrow \forall p \in P \quad m(p) \geq IF(p, t)$
(m è il peso degli archi uscenti è \leq al numero di token contenuti)

$m + m'$ se esiste se esiste una transiz. abilitata e

$$m'(p) = m(p) - IF(p, t) + OF(t, p)$$

ACCETTORE DI PETRI

$$R = (P, T, IF, OF, \Sigma, L, m_0, F)$$

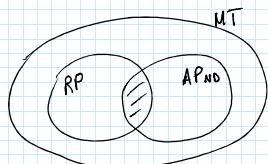
$$F \subseteq \{m : P \rightarrow \mathbb{N}\}$$

$$L : T \rightarrow \Sigma \cup \{\varepsilon\} \quad (\text{una transizione può avere un'etichetta, o non } (\varepsilon) \text{ ovvero})$$

DA PETRI A TURING

Ogni POSTO \rightarrow un nastro

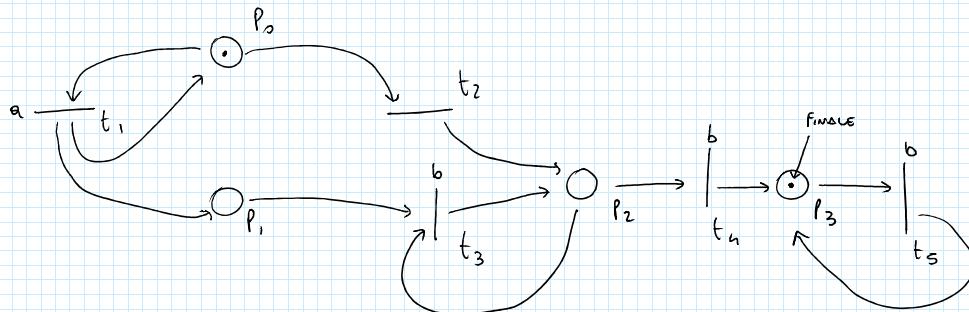
TRANSIZIONE \rightarrow sono le posizioni degli organi di controllo



NOTA: se con archi inhibitori \Rightarrow $MT = RP$

ES. 1

Costruire una rete di petri che riconosce: $L = \{a^m b^m \mid m < m\}$



initio con t_1 , nequio P_3 finale ($\&$ token solo in P_3)

$$\text{sequenze possibili } \left\{ \begin{array}{l} t_1 t_1 t_1 t_2 t_3 t_3 t_3 t_5 t_5^* \\ t_2 t_4 t_5^* \end{array} \right\} \Rightarrow t_1^m t_2 t_3^m t_5 t_5^*, m \geq 0$$

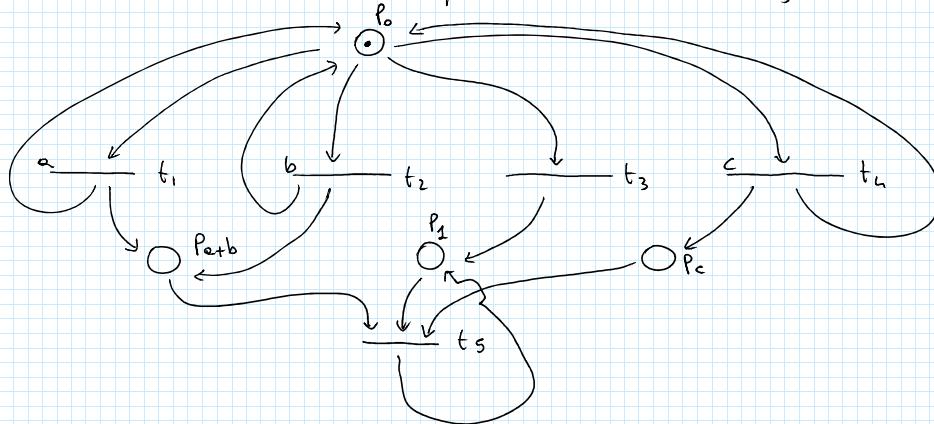
quindi

$$L(R) = \left\{ a^m b^m b b^* \mid m \geq 0 \right\}$$

ES. 2

$$L = \left\{ w \in \{a, b, c\}^* \mid \#_a w + \#_b w = \#_c w \right\}$$

$$\text{anagrammi di } \left\{ a^m b^m c^{n+m} \mid m, n \in \mathbb{N} \right\}$$



punto con t_1, t_2 o t_4 poiché non disabilitano P_0

es. $t_1 t_2 t_1 t_2 t_4 t_5 t_6 \dots$

poi sparo t_3 e il token no in P_1

- t_1, t_2, t_3 si disabilitano
- si abilita t_5

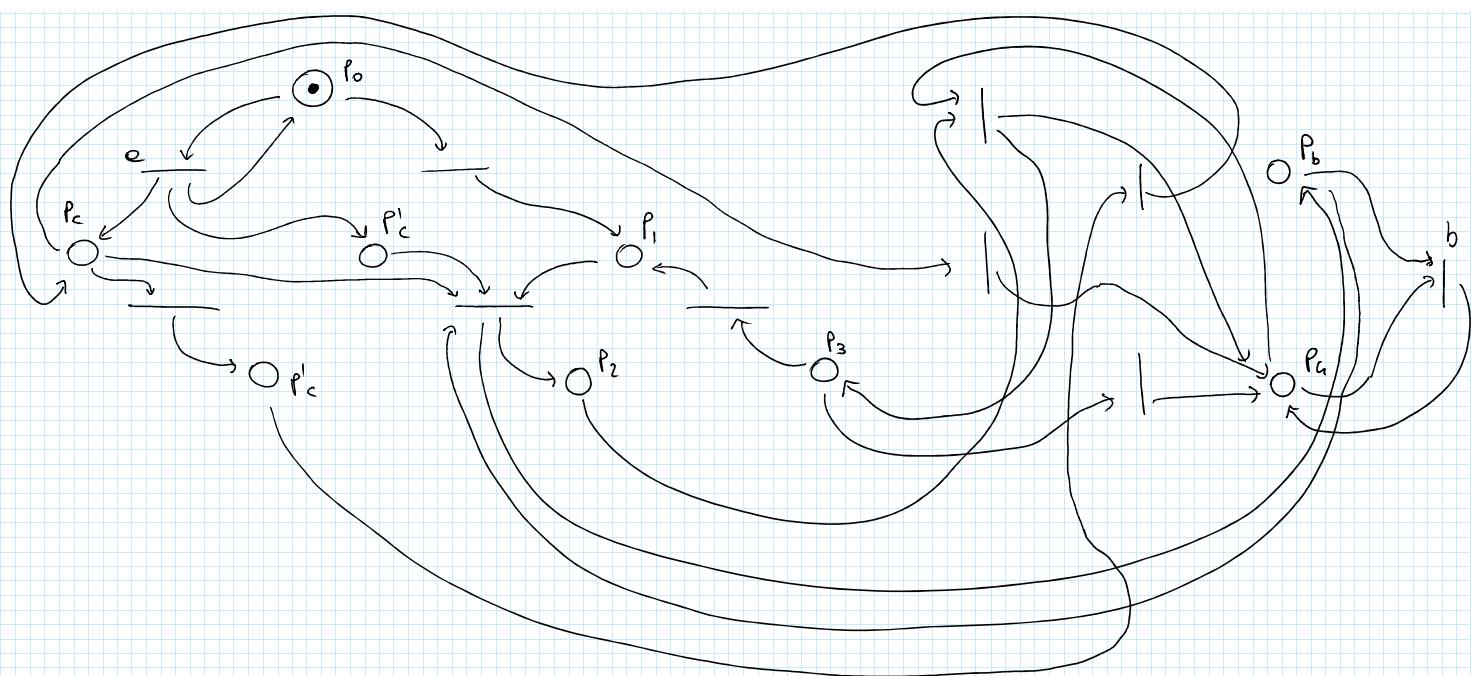
$\dots t_5 t_5 t_5 t_5$

abilitato:

a c a a b c c c

ES. 3

$$L = \left\{ a^m b^m \mid m \leq m \leq m^2 \right\}$$



ESERCITAZIONE

martedì 4 novembre 2014 09:49

GRAMMATICHE GENERATIVE

tipo 0 (non ristrette) [MT]

$$G = (V_T, V_N, P, S)$$

↑ ↑ ↗ \text{produzioni} \\
 simb. terminali non terminali

$V_T \cap V_N = \emptyset$
 $S \in V_N^*$
 $P \subseteq V^+ \rightarrow V^*$ P finito
 $(\alpha, \beta) \in P$ scrivo $\alpha \rightarrow \beta$

tipo 1 (sensibili al contesto) [MT A NASTRO SIN SOLO]

$$P \subseteq (V^+ \times V_N \times V^*) \times V^*$$

$\alpha \overbrace{A \beta}^{\leftarrow} \rightarrow \alpha \not\perp \beta$

tipo 2 (libere dal contesto) [AP No]

$$P \subseteq V_N \times V^*$$

$$A \rightarrow \beta$$

tipo 3 (grammatiche regolari) [AF]

$$A \rightarrow \alpha B \quad (\text{non terminali a sx})$$

$$A \rightarrow \alpha$$

$$A \rightarrow \epsilon$$

$$A, B \in V_N$$

$$\alpha \in V_T$$

ES. 1

scrivere grammatica che riconosce

$$\mathcal{L} = \{w \in \{a, b\}^* \mid \#_a w = \#_b w\}$$

tipo 0)

(MT)

$$S \rightarrow ABS | \epsilon$$

$$AB \rightarrow BA$$

$$BA \rightarrow AB$$

$A \rightarrow a$
 $B \rightarrow b$
 $ab \rightarrow ba$ (non riportate, si ponono risparmio anche simboli terminali)
 tipo 2)

(AP_{NO}) $S \rightarrow aSbS \mid bSaS \mid \epsilon$

$$S \rightarrow aSbS \rightarrow abSaSbS \rightarrow abab$$

Es. 2

$$\mathcal{L} = \{a^m b^m c^m a^m \mid m \geq 0, m \geq 1\}$$

$S \rightarrow S_1 S_2$
 $S_1 \rightarrow a S_1 b \mid \epsilon$
 $S_2 \rightarrow c S_2 a \mid ca$

ES. 3

$$\mathcal{L} = \{a^n b^n \mid n \geq 1\} \cup \{a^n b^{2n} \mid n \geq 1\}$$

$S \rightarrow S_1 \mid S_2$
 $S_1 \rightarrow a S_1 b \mid ab$
 $S_2 \rightarrow a S_2 bb \mid abb$

ES. 4

$$\mathcal{L} = \{a^n b^n c^n d^n \mid n \geq 1\}$$

Se ne una MT (non basta AP_{NO})

$S \rightarrow a S_1 B C Y$	
$S_1 \rightarrow a S_1 B \mid \epsilon$	
$B \rightarrow b X$	$X Y \rightarrow d Y$
$X b \rightarrow b X$	$X d \rightarrow d X$
$X C \rightarrow c C X$	$Y \rightarrow \epsilon$
$X c \rightarrow c X$	$C d \rightarrow d$

Svolgo:

$$\begin{aligned} & aS_1BCY \\ & \alpha S_1BBCY \\ & \alpha^m S_1B^mCY \\ & \alpha^n bX B^{n-1}CY \\ & \alpha^m (bX)^m CY \end{aligned}$$

$$\begin{aligned} & \alpha^m b^m X^m CY \\ & \alpha^m b^m X^{m-1} c CX Y \\ & \alpha^m b^m c^m CX^m Y \\ & \alpha^m b^m c^m CX^{m-1} d Y \\ & \alpha^m b^m c^m d^m \end{aligned}$$

ES. 5

$$\lambda = \left\{ a^m \mid m \geq 0 \right\}$$

tipo 0)

$$\begin{aligned} S &\rightarrow XAY \\ X &\rightarrow XC \mid \epsilon \\ CA &\rightarrow AAC \\ CY &\rightarrow Y \mid \epsilon \\ A &\rightarrow a \\ Y &\rightarrow \epsilon \end{aligned}$$

$$\begin{aligned} S &\quad XAY \quad XCA Y \\ &\quad XAAC Y \\ &\quad XAA Y & C \rightarrow \epsilon \\ &\quad XCAA Y & XCAAA Y \\ &\quad XAACAY & XAACAAA Y \\ &\quad XAA AAC Y & XAAAAAACAY \\ && \xrightarrow{x,y,c \rightarrow \epsilon} \\ && AAAAAAA A \end{aligned}$$

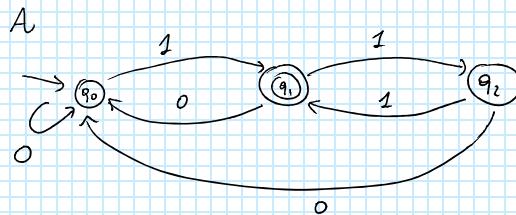
ESERCITAZIONE

lunedì 10 novembre 2014 - 14:23

segue grammatiche

(grammatiche di tipo 3 ~ ASF)

- 1) trovare grammatica che riconosce $L(A)$ $[AF \rightarrow GR \text{ tipo 3}]$



$$G = (\{0,1\}, \{q_0, q_1, q_2\}, Q_0, P)$$

$$P: Q_0 \rightarrow 1Q_1 | 0Q_0$$

$$Q_0 \rightarrow 1Q_1 \rightarrow 11Q_2 \rightarrow 110Q_0 \rightarrow$$

$$Q_1 \rightarrow 1Q_2 | 0Q_0$$

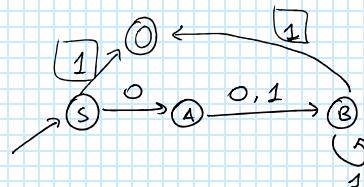
$$\rightarrow 1101Q_0 \rightarrow 11010Q_0$$

$$Q_2 \rightarrow 1Q_1 | 0Q_0$$

$$\rightarrow 110101Q_1$$

- 2) $G = (\{0,1\}, \{A, B, S\}, S, P)$ $[GR \rightarrow AF]$

$$P: S \rightarrow 0A | 1B$$



$$A \rightarrow 0B | 1B$$

$$B \rightarrow 1 | 1B | \epsilon$$

(grammatiche di tipo 2 ~ AP_{NO})

- 1) grammatica autonoma $[GR \rightarrow AP]$

$G = (V_T, V_N, S, P)$ esiste un APND che accette per pila vuota con un solo stato.

$$A_G = (\{q\}, V_T, V = V_T \cup V_N, \tau, q, S)$$

per ogni produzione del tipo $A \rightarrow \beta$ si ha $\tau(q, \epsilon, A) = (q, \beta)$

e per ogni $a \in V_T$ $\tau(q, a, a) = (q, \epsilon)$

$$L = \{a^m b^m \mid m \geq 1\} \cup \{a^m b^{2m} \mid m \geq 1\}$$

$$G = (\{a, b\}, \{S_1, S_2\}, S, P)$$

$$P: S \rightarrow S_1 | S_2$$

$$S_1 \rightarrow aS_1b | ab$$

$$S_2 \rightarrow \alpha S_2 b b | \alpha b b$$

costruire AP: (in blu si con stato finale)



\Rightarrow TEST

$\alpha \alpha b b$

$$\begin{matrix} \alpha, S_1 & \alpha, S_1 \\ \alpha, S_1 | S_2 & \alpha, S_2 | \alpha b b \\ \alpha, S_1 | \alpha S_1 b & \alpha, \alpha | \epsilon \\ \alpha, S_1 | \alpha b & b, b | \epsilon \\ \alpha, z_0 | S_2 & \alpha, z_0 | z_0 \end{matrix}$$

$$\begin{matrix} \alpha, S_1 & \alpha, S_1 \\ \alpha, S_1 | S_2 & \alpha, S_2 | \alpha b b \\ \alpha, S_1 | \alpha S_1 b & \alpha, \alpha | \epsilon \\ \alpha, S_1 | \alpha b & b, b | \epsilon \\ \alpha, z_0 | S_2 & \alpha, z_0 | z_0 \end{matrix}$$

$$\begin{matrix} \alpha, S_1 & \alpha, S_1 \\ \alpha, S_1 | S_2 & \alpha, S_2 | \alpha b b \\ \alpha, S_1 | \alpha S_1 b & \alpha, \alpha | \epsilon \\ \alpha, S_1 | \alpha b & b, b | \epsilon \\ \alpha, z_0 | S_2 & \alpha, z_0 | z_0 \end{matrix}$$

$$\begin{matrix} \alpha, S_1 & \alpha, S_1 \\ \alpha, S_1 | S_2 & \alpha, S_2 | \alpha b b \\ \alpha, S_1 | \alpha S_1 b & \alpha, \alpha | \epsilon \\ \alpha, S_1 | \alpha b & b, b | \epsilon \\ \alpha, z_0 | S_2 & \alpha, z_0 | z_0 \end{matrix}$$

NOTA: se un automa ha un solo stato, il linguaggio riconosciuto è sicuramente quello a potenze minima.

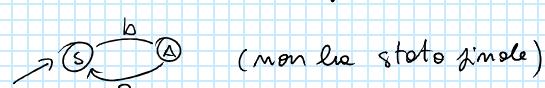
2)

$$G = (\{\alpha, b\}, \{A, S\}, S, P)$$

$$\begin{array}{l} P: \quad S \rightarrow A b \\ \quad \quad \quad A \rightarrow S \alpha \end{array}$$

$$L(G) ?$$

$L(G) = \emptyset$ riconosce il linguaggio nullo



$$3) \quad G = (\{\alpha, b, c\}, \{S, A\}, S, P)$$

$$\begin{array}{l} P: \quad S \rightarrow \alpha S c | A \\ \quad \quad \quad A \rightarrow b A c | \epsilon \end{array}$$

$$\text{DIM che } L(G) = \underbrace{\{\alpha^n b^m c^{n+m} \mid n, m \geq 0\}}_{L} \quad L = L(G)$$

¶

$$\textcircled{1} \quad L \subseteq L(G)$$

$$\textcircled{2} \quad L(G) \subseteq L$$

$$\textcircled{1} \quad w \in L \quad w = \alpha^n b^m c^{n+m}$$

$$S \rightarrow \alpha S c \xrightarrow{n-1} \alpha^n S c^n \xrightarrow{} \alpha^n A c^n \xrightarrow{} \alpha^n b A c c^n \xrightarrow{m-1} \alpha^n b^m A c^{m-1} c^n$$

$$\textcircled{2} \quad \text{sic w una forma di frase di } G$$

$$w = \alpha^K S c^K, \quad K \geq 0 \quad \text{oppure} \quad w = \alpha^K A c^K, \quad K \geq 0$$

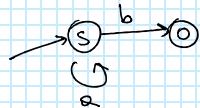
$$w = \alpha^k b^j A c^j c^K, \quad K \geq 0$$

1) Provare l'equivalenza tra grammatiche

$$G_1: S \rightarrow aSb \mid b$$

$$G_2: \begin{array}{l} S \rightarrow bAb \\ A \rightarrow aAa \end{array}$$

$$\mathcal{L}(G_1) = \mathcal{L}(G_2) ?$$



$$\mathcal{L}(G_1) = a^* b$$

$$S \rightarrow Ab \rightarrow Aa^*b \rightarrow a^*b$$

quindi sono equivalenti

$$2) G_1: S \rightarrow aAb$$

$$\begin{array}{l} aA \rightarrow a\underset{\sim}{a}Ab \\ A \rightarrow \epsilon \\ \Downarrow \end{array}$$

di tipo 1, non è un automa

ma una grammatica

$$G_2: S \rightarrow aSb \mid ab$$

$$\Downarrow$$

$$\mathcal{L}(G_2) = \{a^m b^m \mid m \geq 1\}$$

$$S \rightarrow aAb \rightarrow a aAb \xrightarrow{*} a^* A b^* \rightarrow a^* b^*$$

$$\mathcal{L}(G_1) = \{a^m b^m \mid m \geq 1\}$$



$\searrow \rightarrow$ grammatiche equivalenti
ma G_2 è la potente minima.

3) Scrivere grammatica che produce le espressioni aritmetiche che contengono + e *

$$\text{es. } (((2+3)*5)+(3+2))*4$$

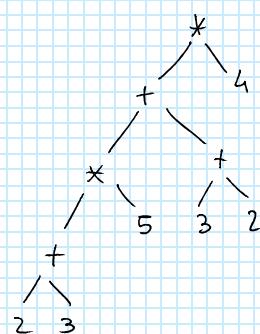
terminali: (,), +, *, R

$$E \rightarrow E + T \mid T$$

$$T \rightarrow T * F \mid F$$

$$F \rightarrow i \mid (E) \quad i \in R$$

albero



$$E \rightarrow T \rightarrow T * F \rightarrow T * 4 \rightarrow$$

$$\rightarrow F * 4 \rightarrow (E) * 4 \rightarrow (E + T) * 4$$

$$\xrightarrow{2} (T + F) * 4 \xrightarrow{3} ((E) + (E)) * 4$$

$$\rightarrow ((T * F) + (E + T)) * 4 \rightarrow (((E) * 5) + (3 + 2)) * 4$$

~ " " ~ ~ ~

$\rightarrow (((2+3)*5) \dots)$ fine

4)

$$S \rightarrow () | ((S)) | (S)S | S(S)$$

G genera il linguaggio delle espressioni matematiche ben parentizzate?

esempio:

$$()()() \notin L(G) \quad \text{quindi NO}$$

5)

$$L = \overline{\{ u c u^R \mid u \in \{a,b\}^* \}}$$

costruire una grammatica t.c. $L(G) = L$

1) $\overline{L} : S \rightarrow aSa \mid bSb \mid c \quad (\text{sarebbe } \wedge \text{ di } \wedge \text{ quindi il linguaggio normale})$

- 2) $w \in L$ se e solo se
- ① in w non c'è c in mezzo
 - ② in w c'è c in posizione non mediana
 - ③ in w ci sono caratteri diversi in posizioni simmetriche.

$$S \rightarrow QSQ \mid \underbrace{\varepsilon}_{\textcircled{1}} \mid \underbrace{a|b}_{\textcircled{2}} \mid \underbrace{QRc}_{\textcircled{3}} \mid \underbrace{cRQ}_{\textcircled{3}} \mid \underbrace{aRb}_{\textcircled{3}} \mid \underbrace{bRa}_{\textcircled{3}}$$

$$R \rightarrow QR \mid \varepsilon$$

$$Q \rightarrow a \mid b \mid c$$

6) es. $\overline{\{a^n b^m \mid n \geq 1\}}$

succursamento

$$\begin{array}{c} \varepsilon \cup \overline{\{a^* b^*\}} \cup \overline{\{a^n b^m \mid n \neq m\}} \\ \swarrow \qquad \qquad \qquad \searrow \\ S \rightarrow S^1 \mid S'' \mid S^m \end{array}$$

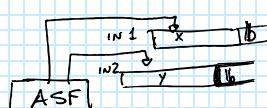
$S \rightarrow aSa \mid S_1$
 $S_1 \rightarrow aS_2 \mid S_2b$
 $aS_2 \rightarrow aaS_2$
 $S_2b \rightarrow S_2bb$
 $S_1, S_2 \rightarrow \varepsilon$

FORMALIZZAZIONE

1) Definire una macchina

"automa a 2 nastri" 2FA

che legge da uno tra i 2 nastri gli input
e riceve dello stato in cui si trova



$$\mathcal{A} = (\Sigma, Q, q_0, \tau, F)$$

(PARTO DA UN AF E AGGIUNGO 1 NASTRO)

$$Q = Q_1 \cup Q_2 \quad \text{con} \quad Q_1 \cap Q_2 = \emptyset$$

$$\tau : (Q \times \Sigma) \rightarrow Q \rightarrow$$

configurazione $(q, x, y) \quad x, y \in \Sigma^*$

transizione $(q, x, y) \xrightarrow{} (q', x', y') \Leftrightarrow q \in Q_1, \tau(q, \alpha) = q', x = \alpha x' \xrightarrow{\alpha \in \Sigma} \boxed{\alpha \in \Sigma}$

$(q, x, y) \xrightarrow{} (q', x, y') \Leftrightarrow q \in Q_2, \tau(q, \alpha) = q', y = \alpha y' \xrightarrow{\alpha \in \Sigma} \boxed{\alpha \in \Sigma}$

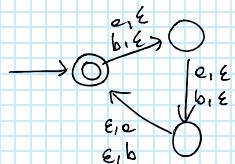
$$C_0 = (q_0, x_0, y_0) \quad x_0, y_0 \in \Sigma^*$$

$$C_f = (q_f, \epsilon, \epsilon) \quad x_f, y_f \in \Sigma^* \quad q_f \in F$$

$$L(A) = \{ w \in \Sigma^* x \in \Sigma^* \mid C_0 \xrightarrow{w} C_f \}$$

2)

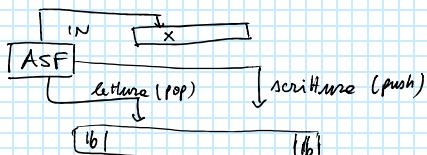
$$L = \{ (w, u) \in (\{a, b\}^*)^2 \mid |w| = 2|u| \}$$



3) FORMALIZZARE UN AUTOMA A CODA

- legge caratteri da un nostro impianto e da un fronte di coda
- in base ai caratteri letti in IN in coda, e allo stato corrente cambia stato e scrive una stringa in fondo alla coda
- avviene ϵ -mosse
- accetto per stato finale

Prendo un automa a Pilota e si modifica la pila in coda.



$$Ac = \{ Q, I, \Gamma, i, \tau, F, z_0 \}$$

$$\tau : (Q \times (I \cup \{\epsilon\}) \times (\Gamma \cup \{\epsilon\})) \rightarrow Q \times \Gamma^*$$

configurazione: $C = (q, x, \alpha) \quad x \in \Sigma^*, \alpha \in \Gamma^*$

$$C_0 = (i, x_0, z_0)$$

$$C_f = (q_f, \epsilon, \alpha) \quad q_f \in F$$

$$L(A_C) = \{ w \in \Sigma^* \mid \text{col}^w c_f \}$$

transizioni:

$$(q, x, \alpha) \xrightarrow{} (q', x', \alpha') \quad \text{se e solo se}$$

- ① $x = a x'$, $\alpha = b \beta$, $\alpha' = \beta \gamma$, $\tau(q, a, b) = (q', \gamma)$ $\xrightarrow{\beta, \gamma \in \Gamma^*, b \in I}$
- ② $x = x'$, $\alpha = b \beta$, $\alpha' = \beta \gamma$, $\tau(q, \epsilon, b) = (q', \gamma)$ $\xrightarrow{c \in I}$
- ③ $x = a x'$, $\alpha' = \alpha \gamma$, $\tau(q, a, \epsilon) = (q', \gamma)$
- ④ $x = x'$, $\alpha' = \alpha \gamma$, $\tau(q, \epsilon, \epsilon) = (q', \gamma)$

NOTA:

UN AUTOMA A CODA PUÒ SIMULARE UNA MACCHINA DI TURING
(ma non può essere elettronica)

VERIFICO CHE UNA TRANSIZIONE DI UNA MT a metà simbolo è simulabile da un AUTOMA A CODA:

es:

$$\xrightarrow{\#|u|A|B|q|C|D|r|\$} \xrightarrow{\#|u|A|q'|B|C|D|r|\$|}$$

simulo questa transizione con un automa a codice AC

	operazione	contenuto codice	stato
①	legge $\# u ABqCDr\$$ da sx a dx e lo memorizza in codice	$\# u ABqCDr\$$	$q \$$
②	finché non trovi q ricepisce in fondo alla codice il fronte di codice	$qCDr\$ \# u A$ (B è in memoria)	q_B
③	quando leggo q metto in codice $q'B$	$CDr\$ \# u Aq'B$	q_B
④	leggo C in q_B , metto in codice c' , metto q_C	$Dr\$ \# u Aq'C$	q_C
⑤	leggo e ricopio fino a $\$$	$\# u Aq'Bc'Dr\$$	$q\$$

9) FORMALIZZARE AUTOMA A 2 PILE (DET)

si devono mettere insieme 1) e 2)

$$(q, x, \alpha_1, \alpha_2) \xrightarrow{} (q', x', \alpha'_1, \alpha'_2)$$

$$\tau: (Q \times (I \cup \{\epsilon\}) \times \Gamma \times \Gamma) \rightarrow Q \times \Gamma^* \times \Gamma^*$$

(la potenza è quella di una MT)

LOGICA DEL I ORDINE

ES.

2+3 termine

ES.

$2+3$ termine

$2+3 = x$ f.b.f. (formula sulla quale si può dire se vero o falso)

$\leq \dots$

SIMBOLI GLA

$\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$

(,)

\forall, \exists

$$N = \{x, y, \dots\} \quad C = \{a, b, c, \dots\}$$

$$F = \{f^d, \dots\} \quad P = \{p^d, \dots\}$$

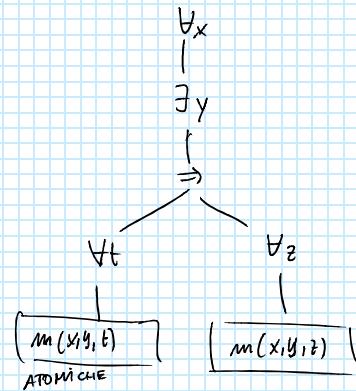
$$t = x | a | f^m \left(\underbrace{t, \dots, t}_m \right)$$

$$FbF := P^m \left(\underbrace{t, \dots, t}_m \right) \mid \neg F \mid F \wedge F \dots$$

$$\forall x F \mid \exists x F$$

$$4) \forall x \exists y (\forall t m(x, y, t) \Rightarrow \forall z m(x, y, z))$$

QUANT. \forall \exists \forall \exists \forall \forall \Rightarrow \forall
 VAR x y t z t t t z
 pred.
 comm.



2) PRE-COND PER PROGRAMMA

CON INPUT 2 ARRAY di caratteri a, b

$$l(a) = m \quad l(b) = n$$

Pre: a è più lungo di b $\underline{\text{e}}$ in b non c'è nemmeno carattere che $\underline{\text{e}}$ b non ruota

①

②

③

$$\textcircled{1} M > m \wedge m > 0$$

$$\textcircled{2} \forall i \forall j ((N(i) \wedge 0 \leq i \wedge i \leq m-1 \wedge N(j) \wedge 0 \leq j \leq m) \Rightarrow b[i] \neq a[j])$$

tipizzazione

$N(x) = "x è naturale"$
 $C(x) = "x è costante"$

NOTA: tutte le volte che si quantifica (\forall, \exists) su una variabile, si deve anche definire il tipo

Tesi di church-Turing

Le funzioni calcolabili sono calcolabili da una MT.

Ogni funzione calcolabile è codificabile con un naturale \Rightarrow
esistono funzioni non calcolabili

$f: N \rightarrow N$ (sono più delle MT)

$$\left| \{f: N \rightarrow \{0,1\} \mid f \text{ calcolabile}\} \right| > \left| \{\text{MT}\} \right|$$

!!

$$\left| P(N) \right|$$

Se A insieme
 $|P(A)| > |A'|$

(in sostanza ci sono più problemi che MT e quindi algoritmi per risolverli)

Halting Problem

input: P programma, x input di P

$$\text{output: } \begin{cases} 1 & \text{se } P(x) \downarrow \text{ termina} \\ 0 & \text{se } P(x) \uparrow \text{ non termina} \end{cases}$$

Equivalent a calcolare

$$\varphi(P, x) = \begin{cases} 1 & \text{se } P(x) \downarrow \\ 0 & \text{altrimenti} \end{cases} \quad \varphi: N^2 \rightarrow \{0,1\}$$

funzione Allora NON ESISTE un programma (MT) che calcoli φ

DIM:

per assurdo sia ϕ un programma che calcola φ

$$\phi(P, x) = \varphi(P, x)$$

programma

USO ϕ come subroutine di un programma P'

se $\phi(P, P) = 1$, P' entra in loop infinito
altrimenti ritorna 1.

CALCOLO $P'(P')$

P' termina o non termina

① se P' termina ($P'(P') \downarrow$) cioè $\varphi(P', P') = 1$

allora $\phi(P', P') = 1$ e dunque P' entra in un loop su P' : assurdo.

② $P'(P') \uparrow$, quindi $\varphi(P', P) = 0$, da cui $\phi(P', P) = 0$,

cioè $\phi(P', P') \neq 1$. $P'(P') = 1$, dunque termina: assurdo.

DEF

Una funzione f è ricorsiva o decidibile, se esiste un programma P che la computa: $f(x) = P(x) \quad \forall x \in \mathbb{N}$

DEF

Un insieme $S \subseteq \mathbb{N}$ è ricorsivamente numerabile se esiste un programma P tale che $P(x) = 1 \quad \forall x \in S$

$$P(x) \uparrow \quad \forall x \notin S$$

DEF

$$\text{HALT} := \left\{ i \in \mathbb{N} \mid P_i(i) \downarrow \right\} \subseteq \mathbb{N}$$

\uparrow
progr. codificato da i

$$= \left[\begin{array}{l} \left\{ P \mid P(P) \downarrow \right\} \\ \text{alternativa meno concreta} \end{array} \right]$$

HALT è indecidibile ma è ricorsivamente enumerabile.

problema decidibile \rightarrow trova algoritmo che lo risolve
 indecidibile \rightarrow nessun algoritmo lo può risolvere

RIDUZIONI

Sono X e Y problemi (cioè $X \subseteq \mathbb{N}$, $Y \subseteq \mathbb{N}$)

una riduzione da X a Y è una funzione f .

- ① Se $x \in X$, allora $f(x) \in Y$
- ② Se $x \notin X$, allora $f(x) \notin Y$
- ③ f è totale e ricorsiva

ES:

$$U_2 := \left\{ \text{codifiche di } P \mid P \text{ ha per output } U_2 \text{ per almeno un input (colonn } P \Rightarrow U_2 \text{)} \right\} \subseteq \mathbb{N}$$

HALT si può ridurre a U_2

Oss l'input di entrambi i problemi è un/naturale (che codifica un) programma.

Devo realizzare una f che sia una trasformazione sintattica di programmi
 tale che se

$$\begin{aligned} P \in \text{HALT} &\text{ allora } f(P) \in U_2 \\ P \notin \text{HALT} &\text{ allora } f(P) \notin U_2 \end{aligned}$$

DATO IL PROGRAMMA P , COSTRUISCO IL PROGR. P' CON:

- 1- calcolo $P(P)$ (chiamlo P su P)
- 2- se termina la computazione del punto 1, stampo U_2

$$P \xrightarrow{f} P'$$

Se $P(P) \downarrow$ allora per ogni input P' restituisce U_2 (se $P \in \text{HALT}$, $P' \in U_2$)

Se $P(P) \uparrow$ allora per ogni input P' rimane al punto 1,
 P' non termina, quindi P' non produce U_2

QUINDI U_2 NON E' DECIDIBILE.

ES:

$$\begin{aligned} \text{TOT} &:= \left\{ P \mid P(x) \downarrow \text{ per ogni } x \right\} \subseteq \mathbb{N} \\ &= \left\{ i \in \mathbb{N} \mid \forall j \in \mathbb{N} \quad P_i(j) \downarrow \right\} \end{aligned}$$

RIDUCO HALT a TOT

Dato un certo P costruisco P' .

1. computa $P(P)$
2. se è quonolo termina, restituisce a_2

Se $P \in \text{HALT}$ allora P' è il programma costante che stampa a_2 .
Quindi $P' \in \text{TOT}$.

Se $P \notin \text{HALT}$, P' non termina. Quindi $P' \notin \text{TOT}$.

(quindi non esiste una soluzione effettiva che stabilisce se per ogni input un programma ha un OUTNT).

Oss

programmi diversi ponono calcolare la stessa funzione.

$P = P'$ se calcolano la stessa funzione

$$\forall x \in N \quad P(x) = P'(x)$$

(programmi sintatticamente diversi ponono essere semanticamente equivalenti)

Un insieme $S \subseteq N$ RISPETTA LE FUNZIONI

se $P \in S$ e $P' \equiv P$ allora $P' \in S$

TEOREMA DI RICE

Se S è un insieme che rispetta le funzioni, e $S \neq \emptyset$ e $S \neq N$
allora S è inoldecidibile

ES.

- Non è possibile stabilire se un programma sia in grado o meno di calcolare una funzione ricorsiva primitiva.
E' decidibile stabilire se un programma calcola o meno una funzione ricorsiva. (sì)
- E' decidibile se una MT ha meno di 5 stati?
- $S = \{i \in N \mid \text{MT}_i \text{ ha meno di 5 stati}\}$ (sì) (se generica NO)
- E' decidibile stabilire se un AP è equivalente a un AP con meno di 5 stati?
(vedi esercitazione precedente) (sì)

ES.

E' decidibile stabilire se il dominio di una funzione computabile è finito?
assumo "funzione computabile" come MT

$$Dg = \{x \mid g(x) \text{ è definito}\}$$

Sia $S \subseteq N$, insieme delle funzioni computabili con dominio finito.

S rispetta le funzioni

$$S = \{i \in N \mid P_i \text{ termina su un n° finito di input}\}$$

$$S \neq \emptyset \quad S \neq N \quad \Rightarrow \quad \text{per Rice NO}$$

ES.

E' decidibile dati $i, j \in N$ stabilire se $i \leq 1000$ e $j \leq 1000$
e $P_i(j) \downarrow$?

la risposta non è no

ES.

$$g(y, x) = \begin{cases} 1 & \text{se } f_y(x) \text{ è pari} \\ 0 & \text{altrimenti} \end{cases}$$

g è computabile? (risp. no)

mentito:

$$\text{Sia } S_0 = \{ f_y \mid f_y(0) \text{ è pari}\} = \{ f_y \mid g(y, 0) = 1\}$$

S₀ rispetta le funzione? (sì, ovvio)

S₀ ≠ ∅, S₀ ⊆ N

Rice. S₀ non è ricorsiva e quindi g non è calcolabile.

OSS g è una funzione totale

ES.

$$g'(y, x) = \begin{cases} 1 & \text{se } f_y(x) \text{ è pari} \\ 1 \text{ indefinito} & \text{altrimenti} \end{cases}$$

(y, x) ∈ D_{g'} se f_y(x) non è pari

g è computabile? (sì)

mentito:

il progr. prende 2 intesi y, x

• se f_y(x) termina

 se f_y(x) pari return 1
 else loop.

ES.

$$h(y) = \begin{cases} 1 & \text{se } f_y(x) \text{ è pari } \forall x \in N \\ 0 & \text{altrimenti} \end{cases}$$

è calcolabile (no)

S = {f_y | f_y totale con codominio contenuto nei pari}

ESERCITAZIONE

martedì 18 novembre 2014 08:35

$$1) \quad g_1: \mathbb{N}^2 \rightarrow \mathbb{N}$$

$$g_1(y, x) = \begin{cases} 1 & \text{se } f_y(x) \text{ è pari} \\ 0 & \text{altr.} \end{cases}$$

$$D_{g_1} = \mathbb{N}^2$$

non computabile *

$$h_1: \mathbb{N}^2 \rightarrow \mathbb{N}$$

$$h_1(y, x) = \begin{cases} 1 & \text{se } f_y(x) \text{ è pari} \\ \perp & \text{altr.} \end{cases}$$

può non terminare

$$D_{h_1} = \{(y, x) \mid f_y(x) \text{ è pari}\}$$

c'è un programma che lo simula completamente

è calcolabile
(computabile,
decidibile,
ricorsiva)

* per il teorema di Rice

$$\text{Sia } S_0 = \{y \in \mathbb{N} \mid f_y(0) \text{ è pari}\}$$

oss se g_1 fosse calcolabile, anche S_0 lo sarebbe

$$\{y \in \mathbb{N} \mid g_1(y, 0) = 1\}$$

se calcolo g_1 allora calcolo S_0

$$A \Rightarrow B \text{ riconducibile a}$$

$$\neg B \Rightarrow \neg A \text{ quindi ho fatto una riduzione. ora dim. che } S_0 \text{ non è calcolabile}$$

$S_0 \neq \emptyset$, $S_0 \neq \mathbb{N}$, si rispetta le funzioni sia $y \in S_0$ e sia $y' \in \mathbb{N}$

$$\text{tale che } f_y = f_{y'} \rightarrow f_{y'}(0) = f_y(0) \rightarrow y' \in S_0$$

↑ è pari quindi anche

t. Rice S_0 non ricorsivo $\rightarrow g_1$ pure

DIMOSTRAZIONE ALTERNATIVA (DIAGONALE)

$$\text{Sia } K: \mathbb{N} \rightarrow \mathbb{N}$$

$$K(x) = \begin{cases} 2, & \text{se } f(x) \text{ non è pari} \\ \perp & \text{altrone} \end{cases}$$

(puoi dire dispari o non definito)
→ se non è pari
→ se non è definito

$$g_1(x, x) = 0$$

se g_1 fosse calcolabile, anche K lo sarebbe

Mostro che K non è calcolabile

P.A. K sia calcolabile, allora $\exists j \in \mathbb{N}$ tale che $K = f_j$

Calcolo $K(j)$:

$$\textcircled{1} \quad K(j) = 2, \text{ allora } f_j(j) = 2, \text{ allora } K(j) = \perp, \text{ ASSURDO}$$

$$\textcircled{2} \quad j \notin D_K: K(j) = 1, \text{ allora } f_j(j) \neq \perp, \text{ allora } f_j(j) \text{ non è pari},$$

$$\text{allora } K(j) = 2, \text{ ASSURDO}$$

2)

Siano

$$g_2: \mathbb{N} \rightarrow \mathbb{N}$$

$$\textcircled{1} \quad g_2(y) = \begin{cases} 1 & \text{se } \forall x \in \mathbb{N}, f_y(x) \text{ è pari} \\ 0 & \text{altr.} \end{cases}$$

N.B. f_y è totale

$h_2 : \mathbb{N} \rightarrow \mathbb{N}$

$$\textcircled{2} \quad h_2(y) = \begin{cases} 1 & \text{se } \forall x \in \mathbb{N}, f_y(x) \text{ è pari} \\ 1 & \text{altr.} \end{cases}$$

$$\textcircled{1} \quad S = \{y \in \mathbb{N} \mid f_y \text{ totale, } \forall m (f_y) \subseteq P\} = g_2^{-1}(1)$$

S soddisfa il t. Rice $\rightarrow S$ non è ricorsiva

- $\textcircled{2}$ h_2 non è ricorsiva (decidibile) (computabile) se un elemento è S primo o poi
 h_2 è la funzione semicaratteristica di S . lo riconosce a sapere ma non è non potrò mai sapere
 (insieme delle funzioni totali, computabili. $\forall m \in S$)
 Se h_2 fosse comput., S sarebbe ricorsivamente enumerabile.
 \hookrightarrow semidecidibile

DIM. DIAG.

Sia C l'insieme (delle cod. di) tutte le funzioni totali computabili

Se C fosse r.e. anche S lo sarebbe

$$C = \{f : \mathbb{N} \rightarrow \mathbb{N} \mid \exists g : \mathbb{N} \rightarrow \mathbb{N}, g \in S, \forall n \in \mathbb{N} f(n) = \frac{g(n)}{2}\}$$

Proviamo che C non è r.e.

PER ASSURDO:

sia C r.e. cioè: esiste $r : \mathbb{N} \rightarrow \mathbb{N}$ enumerazione delle codifiche delle funzioni totali computabili.

Sia f computabile chiamala $E(f)$ la codifica di f

Sia $x \in \mathbb{N}$, chiamala $F(x)$ la preimmagine di E tramite x
 cioè $F(x)$ è la funzione comput. codificata da x

$$E(F(x)) = x \quad F(E(f)) = f$$

[Se r è una enumerazione, allora per ogni funzione tot. ric. f
 esiste $m \in \mathbb{N}$ t.c. $r(m) = E(f)$]

\neg [Sia $K : \mathbb{N} \rightarrow \mathbb{N}$, $K(m) = F(r(m))(m) + 1$]
 espressione contraddittoria

K è calcolabile e totale

$$\exists m \in \mathbb{N} \text{ t.c. } r(m) = E(K)$$

$$\text{calcolo } K(m) = F(r(m))(m) + 1 = F(E(K))(m) + 1 = K(m) + 1$$

NOTA: dimostrazione utilizzabile come "conosciuta"

ASSURDO.

3)

Sia dato φ e ψ g.b.f. della logica del I ordine
 soddisfacibili (ma non vuole)

Sia φ pre-condizione dell'implementazione di P programme
 ψ post-condizione "

Sia assegnata un'implementazione di P.

- E' decidibile se l'implementazione data soddisfa le pre/post condizioni?
(soluzione o sì o no \Rightarrow problema (domanda) chiuso \Rightarrow decidibile)
- E' decidibile stabilire se una qualche implementazione di P soddisfa le pre/post condizioni? \blacktriangleleft
(no, a causa del "quodlibet")

$$S = \{ i \in N \mid i \text{ codifica il codice di un'implementazione che soddisfa le pre/post condizioni} \}$$

\uparrow non è ricorsivo, \blacktriangleleft non è soddisfacibile

4)

E' decidibile stabilire se la funzione $f_i: N \rightarrow N$ calcolata dall'i-esimo MT soddisfa le formule

$$\forall x \left(\underbrace{(x < 0 \Rightarrow f_i(x) \neq 5)}_{\text{sempre Vero}} \wedge \underbrace{(x \geq 0 \Rightarrow (f_i(x) > 37 \vee (f_i(x) \neq \perp \Rightarrow f_i(x) < 100)))}_{\text{sempre Falso}} \right)$$

dove $<, >, 5, 37$ si interpretano nel modo usuale

\downarrow
equivale a dire

$$\forall x (f_i(x) > 37 \vee f_i(x) < 100)$$

che è sempre verificata (\vee)

Quindi la formula è DECISA

5) E' decidibile se due automi a-s.f. sono equivalenti?

input A_1, A_2 AF

output si: se $L(A_1) = L(A_2)$

mo: altr.

\hookrightarrow equiv: $\underbrace{L(A_1) \cap L(A_2)}_{\emptyset (\neq \{\epsilon\})} = \emptyset$

$\exists A_3$ t.c. $L(A_3) = L(A_1) \cap \overline{L(A_2)}$ quindi si

potrei usare il pumping-lemma, calcolando tutti gli $m-1$; se il primo restituisce sempre vuoto allora ho verificato. (ma di solito non si fa così)

LO STESSO PROBLEMA E' DECIDIBILE

\hookrightarrow AP si (in libro)

\hookrightarrow MT NO

6) PREMESSA:

$$f: \mathbb{N} \rightarrow \{0,1\}$$

$$|\{MT\}| < |\{f: \mathbb{N} \rightarrow \{0,1\}\}| \quad \text{cioè ci sono meno MT delle funzioni possibili.}$$

ES:

$$\text{Sia } f: \mathbb{N} \rightarrow \{0,1,\dots,9\}$$

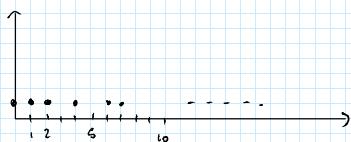
$f(x)$ è la x -esima cifra decimale di π

f è computabile.

$$g: \mathbb{N} \rightarrow \{0,1\}$$

$$g(x) = \begin{cases} 1 & \text{se esistono esattamente } x \text{ cifre 5 consecutive in } \pi \\ 0 & \text{altrimenti} \end{cases}$$

Es. SE $\pi = 3,1415926535897\dots$



g è computabile? (non so) (non so dire se esiste una MT che calcola g o no)

ALTRA FUNZIONE:

$$h: \mathbb{N} \rightarrow \{0,1\}$$

$$h(x) = \begin{cases} 1 & \text{se esistono almeno } x \text{ cifre 5 consecutive in } \pi \\ 0 & \text{altr.} \end{cases}$$

h può essere solo una tra le seguenti funzioni

$$H = \left\{ h_i : \mathbb{N} \rightarrow \{0,1\} \mid \underbrace{h_i}_{\text{comput.}}(x) = \begin{cases} 0, & x > i \\ 1, & x \leq i \end{cases} \right\} \cup \left\{ \underbrace{h_{\infty}(x) = 1}_{\text{computabile}} \right\} \ni h$$

quindi H è una funzione computabile.

7)

Sia K la successione definita da

$$K(m) = \begin{cases} K_0 & ; m=0 \\ \begin{cases} 3K(m-1) + 1 & ; K(m-1) \text{ dispari} \\ \frac{K(m-1)}{2} & ; K(m-1) \text{ pari} \end{cases} & \end{cases}$$

es. $K_0 = 3$

$3, 10, 5, 16, 8, 4, 2, 1, 4, 2, 1 \dots$ periodica.

se domanda è risposta chiusa (sì o no) \rightarrow è decidibile

se la domanda è:

stabilire se per $\forall k_0$ le risposte convergono a 1 \rightarrow si decidibile

(ancora non dimostrato, prob. open)

8)

$$f(x,y) = \begin{cases} 1 & \text{se } f_x(y) \downarrow \text{ e } 0 \leq x,y \leq 1000 \\ 0 & \text{altr. } (f_x(y) \uparrow) \end{cases}$$

Sia M MT dato e y input dato.

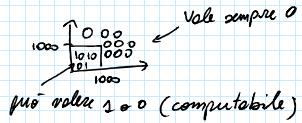
→ risposta è sì o no ⇒ è decidibile
(problema chiuso)

input: \emptyset

output: 1 se $M(y) \downarrow$
0 altr.

è computabile, ma $2^{1000.000}$ funzioni computabili

(la funzione è a un sistema finito di funzioni computabili)



9)

$$L = \{a^n b^m \mid 0 \leq n \leq m\} \quad D = \Sigma^* = \{a, b\}^*$$

trovare formule del I ordine che descrive questo linguaggio

$$\forall x (\underbrace{x \in L}_{L(x)} \iff \exists y \exists z (x = yz \wedge y = ay' \wedge z = bz' \wedge y' \in L_a \wedge z' \in L_b \wedge \ell(y') < \ell(z')))$$

$$\begin{aligned} \forall x (x \in L_a \iff \exists y \in L_a (x = ay)) \\ \forall x (x \in L_b \iff \exists y \in L_b (x = by)) \end{aligned}$$

$$\forall x (x \neq \epsilon \Rightarrow \ell(\epsilon) < \ell(x)) \quad (\text{stringa non vuota})$$

vedere
poi vede: "parole riconosciute da grammatica?"

Tutorato

1) DECIDIBILITÀ

Si consideri il problema:

dato un generico traduttore ASF T e una stringa $y \in \Omega^*$, chiede se esiste $x \in I^*$ f. c. $y = T(x)$ dove $T : I^* \rightarrow \Omega^*$ è la traduzione definita da T .

Il problema è decidibile?

NOTA:

Se forse stato
TRASDUTTORE e STRANGA \rightarrow prob clamor

Input $\langle T, y \rangle$

Output / si, esiste x t.c. $T(x) = y$
 / no, altrimenti.

SI, DECIDIBILE, DEMOSTRAZIONE:

- Se T non ha ϵ -morse allora è "facile" risolvere il problema
(perché $l(x) = l(y)$ (output))
 - In la traduzione ovviene letters per letters è facilmente decidibile

PROCEDIMENTO

Elenco tutte le stringhe x su I in ordine di lunghezza crescente

Per ogni calcolo $\tau(x)$ finché $|\tau(x)| > |y|$

Se durante tale enumerazione trova una sottostringa u di x t.c. $x = zws$

$$\delta^*(q_0, z) = q \quad \delta^*(q, w) = q \quad e \quad \mu^*(q, w) = e$$

w può essere eliminato da x

senza combiere il risultato della traduzione $\tau(x) = \tau(z)$

$$x = \underbrace{a \cdot a}_{y} \quad b \cdot b \quad \tau(x) = \tau(a \cdot a - b \cdot b) = \tau(a^2 - b^2)$$

- 2). E' data una grammatica G

Stabilire se esiste A ASF tale che $L(A) = L(G)$

11

- la grammatica è STABILITA
 - equivalente a dire se $L(G)$ è regolare

11

Problema chiuso DECIDIBILE

- Per una generica grammatica G

- $G \sim MT$
 - equivalente a dire se $L(G)$ è regolare

- $S = \{i \in N \mid M_i \text{ calcola un ling. regolare}\}$ non è ricorsiva
Se $i \in S \wedge M_j \equiv M_i$ allora $j \in S \rightarrow$
S non soddisfa il thm Rice (e s riduzione di G)
G non soddisfa Rice
non è decidibile
INDECIDIBILE

- Per una generica MT M
stabilire se ha meno di s stati
 \Downarrow
 $S = \{i \in N \mid M_i \text{ ha meno di } s \text{ stati}\}$
se $i \in S \wedge M_j \equiv M_i$, allora M_j ha meno di s stati?
NO \Rightarrow DECIDIBILE (DECISO)

NOTA:

$$S = \{i \in N \mid f_i \text{ ha meno di } s \text{ stati}\}$$

$$S' = \{i \in N \mid f_i \text{ calcola un linguaggio regolare}\}$$

ogni funzione computabile è computabile da una MT, e ogni MT è equivalente a un'altra MT con un solo stato

- 3) • Date due generiche MT M_1 e M_2 stabilire se $L(M_1) = L(M_2)$

\Rightarrow equivalente a: $\underbrace{L(M_1) \cap L(M_2)}_{L(M)} = \emptyset$

$$L(M)$$

equivalente a: mi chiedo se esiste w tale che $M(w) \downarrow$
(halting problem)

- Date una generica MT M_i , $w \in \Sigma^*$
stabilire se $w \in L(M_i)$
(equivalente e sopre, equivalente dell'halting problem)

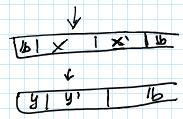
- 4) Formalizzare una MA con simbolo di reset e sx
è una MT e 1 mostro, quando legge uno speciale simbolo in input, sposta immediatamente le testine del mostro di memoria nella prima posizione.



$$\begin{aligned} \text{MA}_T &= (Q, T, M, q_0 \in Q, r \in T, t_0 \in M, F \subseteq Q, \tau: Q \times T \cup \{\epsilon\} \times M \cup \{\epsilon\} \rightarrow \\ &\rightarrow Q \times M \cup \{\epsilon\} \times \{sx, dx, st\} \times \{sx, dx, st, re\}) \end{aligned}$$

\uparrow
combi state

configurazione: $c = (q, \downarrow x^1, y \downarrow y^1)$
 $x \in I^*$



$c_i = (q_0, \downarrow x, \downarrow z)$ conf. iniziale

$c_f = (q, \downarrow x^1, y \downarrow y^1)$ conf. finale con $q \in F$

$c \vdash c_1$

$c_1 = (q_1, x_1 \downarrow x_1^1, y_1 \downarrow y_1^1)$

$$x^1 = e \cdot x^{11}$$

$$\tau(q, e, b) = (q_1, f, sx, ss)$$

$$y^1 = b \cdot y^{11}$$

$$x \downarrow x^1 \rightarrow \bar{x} \downarrow cx^1$$

$$x = \bar{x} \cdot c$$

$$y \downarrow y^1 \rightarrow \bar{y} \downarrow ey^1$$

$$y = \bar{y} \cdot d$$

5) LOGICA

Definire nella logica del I ordine il linguaggio

$$L = \{a^m b^m \mid m \geq 0\}$$

mondo solo le costanti
 e, e, b e la funzione (\cdot)

• CASO CON EL. A POTENZA

$$D = \{a^m b^m\}^* \cup N \quad \begin{matrix} \leftarrow \text{tipizzato} \\ \text{mi serve un predicato unario } N \end{matrix} \quad N(x): "x \in \mathbb{N}"$$

$$\forall x (L(x) \Leftrightarrow \exists m \in (x = a^m b^m)) \quad \begin{matrix} \neg \exists \\ \text{non} \end{matrix} \text{ meglio}$$

$$\forall x (L(x) \Leftrightarrow (\neg N(x) \wedge \exists m (N(m) \wedge x = a^m b^m)))$$

• CASO SENZA POTENZE (con ricorrenza)

$$\forall x (L(x) \Leftrightarrow (x = e \vee \exists y (x = e \cdot y \cdot b \wedge L(y))))$$

NOTA: SE HO $a^m b^{2m}$ basta che aggiungo uno b qui

6) • CASO

$$L = \{a^m b^m \mid m \geq 3\}$$

automa e pot. minima. $\Rightarrow AP_{DET}$ con 3 stati obbligati che mettono a .

• CASO

$$L = \{a^m b^m \mid m \geq 3\}^+ \wedge (\text{note: tutte le permutazioni con } n \text{ come } \geq 3)$$

gli AP DET non sono chiusi rispetto a $*$ o $+$
 quindi ci poniamo enza dei cui due li licenziano ma non sempre

7) • LUNGHEZZA DI UNA STRUNGA IN LOGICA I ORDINE MINIMA

$$L = \{a^m b^m \mid 0 < m \leq m\}$$

$$\forall x (L(x) \Leftrightarrow \exists y \exists z (La(y) \wedge L_b(y) \wedge l(y) \leq l(z)))$$

$$D = \varepsilon^* \cup N \quad \begin{matrix} \leftarrow \text{per conteggi strunge} \end{matrix}$$

$$\forall x \forall m ((\exists N(x) \wedge N(m)) \Rightarrow (\ell(x) = m \Leftrightarrow (x = \varepsilon \wedge m = 0) \vee \exists y (\exists N(y) \wedge m = s(\ell(y)) \wedge x = ay \vee x = by))$$

\uparrow successive. please come primitive.

\hookrightarrow

$$\forall x (\mathcal{L}_a(x) \Leftrightarrow (\exists N(x) \wedge (\exists y (\exists N(y) \Rightarrow (x = ay \wedge \mathcal{L}_a(y)) \vee x = \varepsilon)))$$

- LUNGHEZZA CON LOGICA I ORDINE CON PIÙ PRIMITIVE

$$l_n(x) : \forall g (g = l_n(x)) \Leftrightarrow \neg (\exists h (h > g) \Rightarrow x[h] \neq \varepsilon)$$

- DEFINIZIONE DI POSIZIONE IN VETTORE

$$x[m] = a \Leftrightarrow \exists y \exists z (\ell(y) = m-1 \wedge x = ya^z)$$

8) $L = a^* (ba)^*$

$$ba^* \sim x = ba \vee \exists y (y \in L_{ba^*} \wedge x = bay)$$

$$\forall x (x \in L \Leftrightarrow \underbrace{\exists y \exists z (y \in a^* \wedge z \in (ba)^* \wedge x = ya^z \vee x = yzb)}_{\text{se ho complementare metto } \neg \text{ qui!}})$$

se ho complementare metto \neg qui!

1) DA TEMA 17/07/2007

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

- ① dato una generica $f: \mathbb{N} \rightarrow \mathbb{N}$ stabilire se ha minimo
 - ② è semidecidibile
 - ③ cosa cambia se si considerano solo funzioni totali
-
- ④ • in \mathbb{N} esiste sempre minimo per le funzioni
 - equivalente a stabilire se $Df \neq \emptyset \Rightarrow$ NON SI PUÒ STABILIRE A priori (È UNA MT che computa una generica funzione termini o no halting problem).
 - INDECIDIBILE
 - ⑤ sì: perché nel caso in finiti è possibile discutere lo minimo, ma non si può sapere quando e se si fermerà.
 - ⑥ visto che le funzioni totali sono definite su tutto il dominio, hanno sicuramente minimo \Rightarrow DECIDIBILE (sì, DECISO)

NOTA !! se fosse data una funzione finita, sarebbe stata una domanda chiusa (per ④⑤⑥ anche ③)

2) TEMA 20/SETT/2013

FORZA G in 6×7

- ① decidibile stabilire se dato una conf. esiste una sequenza di mosse che porta un giocatore a vincere?
domanda chiusa (DECIDIBILE)
 - se forse generica la configurazione?
 - DECIDIBILE (perché ci sono comunque finiti configurazioni computabili per stabilirlo)

② con schema h.K e configurazione generica.
come 1b perché sempre insieme chiuso

③ se lo schema forse infinito?

DECISO, sì

perché non decide se c'è una strategia che porta a sicuramente a vincere,
ma se c'è una sequenza di mosse che comunque porta a terminare la partita.

- ④ Abbiamo una funzione che dice la migliore di due configurazioni e lei pone: è decidibile il problema di stabilire se dato un generico programma, ritorna la mossa migliore?
NO, NON DECIDIBILE.
Perché riconducibile al problema di stabilire se una generica macchina di turing

input: $M \vdash M$
 output: si se $L(M) = L(\bar{M})$
 no altr.
 posso calcolare una funzione
 equivalente ad un'altra $M\bar{M}$.
 indecidibile per HtP (halting problem)
 $L(M) \wedge L(\bar{M})$

E' SEMI DECIDIBILE

3) 12/06/2012

- ① A famiglie di macchine
- G " " grammatiche
- ② $L(A)$ è ricorsivamente contenuto in $L(G)$ ← insieme di tutti i linguaggi generati...
 ↗ insieme di tutti i linguaggi riconosciuti dalle famiglie (es AP) A di macchine.
- ③ $L(A)$ è ricorsivamente chiuso rispetto al complemento
- ④ $L(G)$ è ricorsivamente chiuso rispetto a \cap
 (dato un algoritmo che dato una certa grammatica G , genera
 una grammatica sempre del tipo dell'insieme dato)

DOMANDA: dato una generica g in G e una generica e in A
 stabilire se tutte le stringhe generate da g sono accettate da A .
 no 2 input (non è a risposta chiusa)

INDECIDIBILE

4) LOGICA

$$L = \{ a^m b^{m^2} \mid m > 0 \}$$

$$(m+1)^2 = m^2 + 2m + 1$$

a a b b b b b b b b

$$\begin{aligned}
 X \in L \iff & x = abv \quad \exists y \exists z (L_a(y) \wedge L_b(z) \wedge L_b(m) \wedge \\
 & \wedge x = ayzmb \wedge l(y) = l(u) \wedge yz \in L)
 \end{aligned}$$

vanno definiti L_a, L_b , ecc.

