Curs 7

November 24, 2021

Teza Church-Turing.

Orice functie efectiv/intutiv calculabila este recursiva/Turing calculabila.
Modele echivalente:

1. Functii recursive
2. Programe standard
3. λ-calcul (Alonzo Church)
4. Algoritm Markov (Andrey Markov, Jr.)
5. Automate coada
6. Sistem tag (Emil Post)
7. Masini cu registri (Marvin Minsky)
8. Automatul celular (Stanislaw Ulam si John von Neumann)
9. Sisteme de rescriere (include gramatica generativa Chomsky)
10. Etc.

Se poate depasi bariera Turing? Hipercalcul (Hypercomputation):accelerare, invatare inductiva, etc.

Codificarea programelor standard.

Etichete: $\{E, A_1, A_2, \ldots\}$; codificam o eticheta $\#(L)$ cu pozitia ei (numaratoarea incepe cu 1).
Variabile: $\{Y, X_1, Z_1, X_2, Z_2, \ldots\}$; codificam o variabila $\#(V)$ cu pozitia sa (numaratoarea incepe cu 1).
Codificarea unei instructiuni I va fi $\#(I) = < a, < b, c >>$, unde

- daca I nu este etichetata, atunci $a = 0$ altfel $a = \#(L)$, unde L este eticheta ei.
- $c = \#(V) - 1$, unde V este variabila care apare in I.
- $b = 0, 1, 2$ daca $I = V \leftarrow V, V \leftarrow V + 1, V \leftarrow V \dot{-} 1$.
- $b = \#(L) + 2$ daca I este $IF \;\; V \neq 0 \;\; GOTO \;\; L$.

Exemplu: I este $A_1 : Z_2 \leftarrow Z_2 + 1$
$\#(I) = < 2, < 1, 4 >> = < 2, 2 \cdot 9 - 1 > = < 2, 17 > = 4(2 \cdot 17 + 1) - 1 = 139$.
Codificarea unui program P cu instructiunile $I_1, I_2, \ldots, I_k$ este
$\#(P) = [\#(I_1), \#(I_2), \ldots, \#(I_k)] - 1$.

Codificarea programelor.

Exemplu: Ce numar are programul:

$E : X_1 \leftarrow X_1 + 1$

$\quad IF\ X_1 \neq 0\ GOTO\ E$

$\#(I_1) =< 1, < 1, 1 >>=< 1, 5 >= 21$

$\#(I_2) =< 0, < 3, 1 >>=< 0, 23 >= 46.$

$\#(P) = 2^{21} \cdot 3^{4}6 - 1.$

Exemplu: Care este programul cu numarul 14999.

Codificarea programelor.

Exemplu: Ce numar are programul:

$E : X_1 \leftarrow X_1 + 1$

$\quad IF\ X_1 \neq 0\ GOTO\ E$

$\#(I_1) =< 1, < 1, 1 >>=< 1, 5 >= 21$

$\#(I_2) =< 0, < 3, 1 >>=< 0, 23 >= 46.$

$\#(P) = 2^{21} \cdot 3^4 6 - 1.$

Exemplu: Care este programul cu numarul 14999.

$14999 + 1 = 15000 = 2^3 \cdot 3 \cdot 5^4$

Deci programul are 3 instructiuni $I_1, I_{2,3}$ cu $\#(I_1) = 3$, $\#(I_2) = 0$, $\#(I_3) = 4$.

- $< a, < b, c >>= 3 \Rightarrow 2^a(2 < b, c > +1) - 1 = 3 \Rightarrow a = 2, 2 < b, c > +1 = 1 \Rightarrow a = 2, b = 0, c = 0$
  Deci $I_1 \equiv A_2 : Y \leftarrow Y$.
- $< a, < b, c >>= 0 \rightarrow a = b = c = 0$. Deci $I_2 \equiv Y \leftarrow Y$.
- $< a, < b, c >>= 4 \rightarrow a = 0, b = 0, c = 1$. Deci Deci $I_3 \equiv X_1 \leftarrow X_1$.

Problema opririi.

Definim predicatul:
$HALT(x,t) \equiv$ programul codificat cu numarul t se opreste pe intrarea x.
Teorema. *Predicatul HALT nu este calculabil cu programe standard.*
Dem. Pp $HALT$ calculabil si construim programul P:

$$A : IF\ HALT(X, X)\ GOTO\ A$$

Functia calculata de P este

Problema opririi.

Definim predicatul:
$HALT(x, t) \equiv$ programul codificat cu numarul t se opreste pe intrarea x.
Teorema. *Predicatul HALT nu este calculabil cu programe standard.*
Dem. Pp $HALT$ calculabil si construim programul P:

$$A : IF\ HALT(X, X)\ GOTO\ A$$

Functia calculata de P este $\psi_P(x) = \begin{cases} 0, \text{ daca } \overline{HALT(x,x)} \\ \text{nedefinit, altfel} \end{cases}$ Fie $\#(P) = t$, deci $HALT(x, t) \equiv \overline{HALT(x, x)}$. Luam $x = t$ si obtinem $HALT(t, t) \equiv \overline{HALT(t, t)}$, contradictie.

Programul universal.

Pentru fiecare $n \geq 1$, definim functia universala de n variabile:

$$\Phi^{(n)}(x_1, x_2, \ldots, x_n, t) = \varphi_P^{(n)}(x_1, x_2, \ldots, x_n), \#(P) = t.$$

TEOREMA. Pentru orice n, functia universala de n variabile este (partial) calculabila cu programe standard.

DEM. Vom construi un program U_n care va fi programul universal pentru calculul functiei universale de n variabile.

Vom nota cu:

K: numarul instructiunii curente din programul cu numarul t ce urmeaza a fi simulate

S: "starea" curenta a programului cu numarul t (va memora valorile tuturor variabilelor la un moment dat).

Programul universal.

Pentru fiecare $n \geq 1$, definim functia universala de n variabile:

$$\Phi^{(n)}(x_1, x_2, \ldots, x_n, t) = \varphi_P^{(n)}(x_1, x_2, \ldots, x_n), \#(P) = t.$$

TEOREMA. Pentru orice n, functia universala de n variabile este (partial) calculabila cu programe standard.

DEM. Vom construi un program U_n care va fi programul universal pentru calculul functiei universale de n variabile.

Vom nota cu:

K: numarul instructiunii curente din programul cu numarul t ce urmeaza a fi simulate

S: "starea" curenta a programului cu numarul t (va memora valorile tuturor variabilelor la un moment dat).

$$Z \leftarrow T + 1(X_{n+1} + 1) \ //Z = [\#(I_1, I_2, \ldots, I_m)]//$$

$$K \leftarrow 1$$

$$S \leftarrow \prod_{i=1}^{n} p_{2i}^{X_i}$$

Programul universal.

$C: \ \ IF \ (K > Lt(Z)) \vee (K = 0) \ GOTO \ F$

Programul universal.

$C: \; IF \; (K > Lt(Z)) \vee (K = 0) \; GOTO \; F$

$U \leftarrow r((Z)_K) \; //codul \;\; < b, c > \;\; al \; instructiunii \; K//$

Programul universal.

$C: \ IF \ (K > Lt(Z)) \vee (K = 0) \ GOTO \ F$

$U \leftarrow r((Z)_K) \ //codul \ <b, c> \ al \ instructiunii \ K//$

$V \leftarrow p_{r(U)} + 1$ //codul variabilei de apare in instructiunea $K//$

Programul universal.

$C: \; IF \; (K > Lt(Z)) \vee (K = 0) \; GOTO \; F$

$U \leftarrow r((Z)_K) \; //codul \; < b, c > \; al \; instructiunii \; K//$

$V \leftarrow p_{r(U)} + 1$ //codul variabilei de apare in instructiunea K//

$IF \; l(U) = 0 \; GOTO \; N$ //nu facem nimic//

$IF \; l(U) = 1 \; GOTO \; I$ //incrementam variabila//

$IF \; \overline{p_V | S} \; GOTO \; N$ //nu facem nimic//

$IF \; l(U) = 2 \; GOTO \; D$ //decrementam variabila//

Programul universal.

$C: \; IF \; (K > Lt(Z)) \vee (K = 0) \; GOTO \; F$

$U \leftarrow r((Z)_K) \; //codul \; < b, c > \; al \; instructiunii \; K//$

$V \leftarrow p_{r(U)} + 1$ //codul variabilei de apare in instructiunea K//

$IF \; l(U) = 0 \; GOTO \; N$ //nu facem nimic//

$IF \; l(U) = 1 \; GOTO \; I$ //incrementam variabila//

$IF \; \overline{p_V | S} \; GOTO \; N$ //nu facem nimic//

$IF \; l(U) = 2 \; GOTO \; D$ //decrementam variabila//

$$K \leftarrow \begin{cases} min_i[l((Z)_i) + 2 = l(U)], & \text{daca un astfel de } i \text{ exista,} \\ 0, & \text{altfel} \end{cases}$$

$GOTO \; C$

Programul universal.

$C:\ IF\ (K > Lt(Z)) \vee (K = 0)\ GOTO\ F$

$U \leftarrow r((Z)_K)\ //codul\ < b, c >\ al\ instructiunii\ K//$

$V \leftarrow p_{r(U)} + 1$ //codul variabilei de apare in instructiunea K//

$IF\ l(U) = 0\ GOTO\ N$ //nu facem nimic//

$IF\ l(U) = 1\ GOTO\ I$ //incrementam variabila//

$IF\ \overline{p_V | S}\ GOTO\ N$ //nu facem nimic//

$IF\ l(U) = 2\ GOTO\ D$ //decrementam variabila//

$$K \leftarrow \begin{cases} min_i[l((Z)_i) + 2 = l(U)], & \text{daca un astfel de } i \text{ exista,} \\ 0, & \text{altfel} \end{cases}$$

$GOTO\ C$

$I:\ S \leftarrow S * p_V$

$GOTO\ N$

Programul universal.

$C: \ IF \ (K > Lt(Z)) \vee (K = 0) \ GOTO \ F$

$U \leftarrow r((Z)_K) \ //codul \ < b, c > \ al \ instructiunii \ K//$

$V \leftarrow p_{r(U)} + 1$ //codul variabilei de apare in instructiunea K//

$IF \ l(U) = 0 \ GOTO \ N$ //nu facem nimic//

$IF \ l(U) = 1 \ GOTO \ I$ //incrementam variabila//

$IF \ \overline{p_V | S} \ GOTO \ N$ //nu facem nimic//

$IF \ l(U) = 2 \ GOTO \ D$ //decrementam variabila//

$$K \leftarrow \begin{cases} min_i[l((Z)_i) + 2 = l(U)], & \text{daca un astfel de } i \text{ exista,} \\ 0, & \text{altfel} \end{cases}$$

$GOTO \ C$

$I: \ S \leftarrow S * p_V$

$GOTO \ N$

$D: S \leftarrow S/p_v$

Programul universal.

$C: \; IF \; (K > Lt(Z)) \vee (K = 0) \; GOTO \; F$

$U \leftarrow r((Z)_K) \; //codul \; < b, c > \; al \; instructiunii \; K//$

$V \leftarrow p_{r(U)} + 1$ //codul variabilei de apare in instructiunea K//

$IF \; l(U) = 0 \; GOTO \; N$ //nu facem nimic//

$IF \; l(U) = 1 \; GOTO \; I$ //incrementam variabila//

$IF \; \overline{p_V | S} \; GOTO \; N$ //nu facem nimic//

$IF \; l(U) = 2 \; GOTO \; D$ //decrementam variabila//

$$K \leftarrow \begin{cases} min_i[l((Z)_i) + 2 = l(U)], & \text{daca un astfel de } i \text{ exista,} \\ 0, & \text{altfel} \end{cases}$$

$GOTO \; C$

$I: \; S \leftarrow S * p_V$

$GOTO \; N$

$D: S \leftarrow S/p_v$

$N: \; K \leftarrow K + 1$

$GOTO \; C$

Programul universal.

$C:\ \ IF\ (K > Lt(Z)) \vee (K = 0)\ GOTO\ F$

$U \leftarrow r((Z)_K)\ //codul\ \ < b, c >\ \ al\ instructiunii\ K//$

$V \leftarrow p_{r(U)} + 1$ //codul variabilei de apare in instructiunea K//

$IF\ l(U) = 0\ GOTO\ N$ //nu facem nimic//

$IF\ l(U) = 1\ GOTO\ I$ //incrementam variabila//

$IF\ \overline{p_V|S}\ GOTO\ N$ //nu facem nimic//

$IF\ l(U) = 2\ GOTO\ D$ //decrementam variabila//

$$K \leftarrow \begin{cases} min_i[l((Z)_i) + 2 = l(U)], & \text{daca un astfel de } i \text{ exista,} \\ 0, & \text{altfel} \end{cases}$$

$GOTO\ C$

$I:\ \ S \leftarrow S * p_V$

$GOTO\ N$

$D: S \leftarrow S/p_v$

$N:\ \ K \leftarrow K + 1$

$GOTO\ C$

$F: Y \leftarrow (S)_1$

Codificarea masinilor Turing.

$M = (Q, V, U, \delta, q_0, B, F)$

- $\{q_0, q_1, q_2, \ldots\}$ enumerare a tuturor starilor. Codificam q_i fie cu $qbin_i$ sau $q0^i$
- $\{s_0, s_1, s_2, \ldots\}$ enumerare a tuturor simbolurilor. Codificam s_i fie cu $sbin_i$ sau $s0^i$. Convenim $s_0 = B$.
- Codificarea masinii Turing M va fi

$$< M >= w_1\$w_2\$w_3\$w_4\$q_0\$s_0\$w_5,$$

  unde
  - w_1 este un string format din toate codificarile starilor din Q,
  - w_2 este un string format din toate codificarile simbolurilor din V,
  - w_3 este un string format din toate codificarile simbolurilor din $U \setminus V$,
  - w_4 este un string ce codifica functia δ; este o secventa de substringuri de forma $(qbin_{i_1} sbin_{j_1} qbin_{k_1} sbin_{l_1} L/R)$,
  - w_5 este un string format din toate codificarile starilor din F.

Deci $< M >\in \{(,), 0, 1, q, s, L, R, \$\}^*$. Se poate chiar $< M >\in \{0, 1\}^*$?

Codificarea binara a masinilor Turing.

- q_i se codifica cu 0^i,
- s_i se codifica cu 0^i,
- 1 se va folosi ca
- L si R se codifica cu 11 si, respectiv, 111,
- \$ se codifica cu 1111,
- (si) se codifica cu 11111 si, respectiv, 111111.

Orice cuvant se poate codifica printr-un sir binar care incepe cu 1.
Limbajul universal:
$L_u = \{<< M >, < w >>|$
sirul codificat cu $< w >$ este acceptat de masina codificata cu $< M >\}$.
Exista o masina Turing care accepta L_u, numita masina Turing universala.
Afirmatia rezulta si din constructia programului universal.

Codificarea masinilor Turing.

Specific, construim o masina Turing U cu 4 benzi astfel:

- Prima banda contine $<< M >< w >>$.
- A doua banda contine $< w >$.
- A treia banda memoreaza starea curenta a masinii M.
- A patra banda este auxiliara. Se foloseste pentru calcule de decodificare si codificare.
- U cauta o tranzitie codificata in $< M >$ in care starea este cea pastrata pe banda 3 iar simbolul citit este cel de pe banda 2. Atentie: codificarile lor! Pentru aceasta identificare foloseste banda 4.
- Schimba banda 3 pentru a memora noua stare.
- Schimba banda 2 pentru a schimba simbolul citit cu cel scris.
- Muta capul de citire/scriere pe banda 2.
- U se opreste daca starea memorata pe banda 3 este finala in M.

Masini Turing cu $(15, 2), (9, 3), (6, 4), (5, 5), (4, 6), (3, 9), (2, 18)$ stari/simboluri. Depinde numarul de instructiuni.

Multimi/limbaje recursive si recursiv enumerabile.

Algoritm: masina Turing determinista care se opreste pe fiecare intrare.
Definitie

- Un limbaj L (multime A) este recursiv enumerabil (enumerabila) daca exista o masina Turing M a.i. $L(M) = L$ (χ_A este Turing calculabila.)
- Un limbaj L (multime A) este recursiv (recursiva) daca exista o masina Turing M, care se opreste pe fiecare intrare, a.i. $L(M) = L$ (χ_A este Turing calculabila.)

Echivalenta problema de decizie-limbaj: Fie P un predicat de n variabile.
Construim limbajul $L_P = \{< P(x_1, x_2, \ldots, x_n) >| P(x_1, x_2, \ldots, x_n) = 1\}$.

Multimi/limbaje recursive si recursiv enumerabile.

Algoritm: masina Turing determinista care se opreste pe fiecare intrare.
Definitie

- Un limbaj L (multime A) este recursiv enumerabil (enumerabila) daca exista o masina Turing M a.i. $L(M) = L$ (χ_A este Turing calculabila.)
- Un limbaj L (multime A) este recursiv (recursiva) daca exista o masina Turing M, care se opreste pe fiecare intrare, a.i. $L(M) = L$ (χ_A este Turing calculabila.)

Echivalenta problema de decizie-limbaj: Fie P un predicat de n variabile.
Construim limbajul $L_P = \{< P(x_1, x_2, \ldots, x_n) >| \; P(x_1, x_2, \ldots, x_n) = 1\}$.
P este decidabila ddaca L_P este recursiv.

TEOREMA.

1. Limbajul universal este recursiv enumerabil.
2. Limbajul $L_h = \{<< M >, < w >>| \; M$ se opreste pe intrarea $w\}$ este recursiv enumerabil.

Sunt ele recursive?

Multimi/limbaje recursive si recursiv enumerabile.

Fie $M_1, M_2, \ldots,$ o enumerare a masinilor Turing a.i. $< M_1 >, < M_2 >, \ldots$ este o enumerare a codificarilor lor in ordine lexicografica. Analog, fie $w_1, w_2, \ldots$ o enumerare a cuvintelor binare.

TEOREMA. Limbajul (diagonal) $L_d = \{w_i \mid w_i \notin L(M_i)\}$ nu este recursiv enumerabil.

Multimi/limbaje recursive si recursiv enumerabile.

Fie $M_1, M_2, \ldots,$ o enumerare a masinilor Turing a.i. $< M_1 >, < M_2 >, \ldots$ este o enumerare a codificarilor lor in ordine lexicografica. Analog, fie $w_1, w_2, \ldots$ o enumerare a cuvintelor binare.

TEOREMA. Limbajul (diagonal) $L_d = \{w_i \mid w_i \notin L(M_i)\}$ nu este recursiv enumerabil.

DEM. Pp. $L_d = L(M_j)$. Atunci $w_j \in L_d = L(M_j)$ ddaca $w_j \notin L(M_j) = L_d$.

TEOREMA. Limbajele L_u si L_h nu sunt recursive.

DEM. 1. Pp L_u este recursiv, $L_u = L(M)$ a.i. M se opreste pe fiecare intrare. Construim M' care lucreaza astfel pe un cuvant binar w primit la intrare:

- Determina j a.i. $w = w_j$.
- Din j determina M^* a.i. $M^* = M_j$.
- Simuleaza M pe intrarea $<< M_j >, < w_j >>$. Daca M accepta, atunci M' respinge si vice versa.
- $L(M') = L_d$, contradictie.

Demonstratie pentru 2?

Multimi/limbaje recursive si recursiv enumerabile.

TEOREMA.

1. X este recursiva ddaca X si $\complement X$ sunt recursiv enumerabile.
2. Daca X si Y sunt recursive/recursiv enumerabile, atunci $X \cup Y$, $X \cap Y$ sunt recursive/recursiv enumerable.

DEM. Simplu exercitiu.

O *proprietate* a unei clase de limbaje $\mathcal{C}$ este o sumbultime S a lui $\mathcal{C}$. Proprietatea se numeste *triviala* daca $S = \mathcal{C}$ sau $S = \emptyset$. Altfel, se numeste netriviala. Alegem $\mathcal{C}$ ca fiind clasa limbajelor recursiv enumerabile RE. Pentru o proprietate S a clasei RE definim $L_S = \{< M >| \ L(M) \in S\}$.

Teorema Rice.

TEOREMA. Orice proprietate netriviala pe RE este nedecidabila.
DEM. Fie S o proprietate netriviala pe RE, si $\emptyset \notin S$. Aratam ca L_S nu este recursiv. Pp ca $L_S = L(M_S)$, M_S se opreste pe fiecare intrare.
Fie $L \in S$ a.i. $L = L(M_L)$. Alegem si fixam $<< M >, w >$ si construim M' a.i. $L(M') = \begin{cases} L, & \text{daca } w \in L(M), \\ \emptyset, & \text{altfel} \end{cases}$ Cum calculeaza M':

- Initial M' ignora intrarea sa x si simuleaza M pe w.
- Daca M accepta, atunci simuleaza M_L pe x.
- Accepta ddaca M_L accepta.

Construim M_u astfel:

- Pe intrarea $<< M >, < w >>$ determina $< M' >$.
- Simuleaza M_S pe $< M' >$.
- Accepta ddaca M_S accepta. (M_S se opreste pe fiecare intrare.)

Observatie: M_u se opreste pe fiecare intrare!!!
$L \in S \Leftrightarrow < M' > \in L(M_S) \Longrightarrow << M >, < w >> \in L(M_u)$, contradictie.
Cazul $\emptyset \notin S$?

Teorema Rice.

TEOREMA. Orice proprietate netriviala pe RE este nedecidabila.
DEM. Fie S o proprietate netriviala pe RE, si $\emptyset \notin S$. Aratam ca L_S nu este recursiv. Pp ca $L_S = L(M_S)$, M_S se opreste pe fiecare intrare.
Fie $L \in S$ a.i. $L = L(M_L)$. Alegem si fixam $<< M >, w >$ si construim M' a.i. $L(M') = \begin{cases} L, & \text{daca } w \in L(M), \\ \emptyset, & \text{altfel} \end{cases}$ Cum calculeaza M':

- Initial M' ignora intrarea sa x si simuleaza M pe w.
- Daca M accepta, atunci simuleaza M_L pe x.
- Accepta ddaca M_L accepta.

Construim M_u astfel:

- Pe intrarea $<< M >, < w >>$ determina $< M' >$.
- Simuleaza M_S pe $< M' >$.
- Accepta ddaca M_S accepta. (M_S se opreste pe fiecare intrare.)

Observatie: M_u se opreste pe fiecare intrare!!!
$L \in S \Leftrightarrow < M' > \in L(M_S) \Longrightarrow << M >, < w >> \in L(M_u)$, contradictie.
Cazul $\emptyset \notin S$? $\emptyset \notin \complement S$

Problema Corespondentei lui Post.

Problema $PCP(x, y)$

Input: un alphabet V cu cel putin doua simboluri, $n \geq 2$ si doua liste Post

$x = (x_1, x_2, \ldots, x_n)$,

$y = (y_1, y_2, \ldots, y_n)$,

cu $x_i, y_i \in V^*, 1 \leq i \leq n$.

Output: Exista $k \geq 1$ si $1 \leq i_1, i_2, \ldots, i_k \leq n$ a.i.

$x_{i_1} x_{i_2} \ldots x_{i_k} = y_{i_1} y_{i_2} \ldots y_{i_k}$?

1.	2.	3.	4.
b	a	aba	bb
ba	ba	ab	b

1.	2.	1.	3.	3.	4.
b	a	b	aba	aba	bb
ba	ba	ba	ab	ab	b

Problema Corespondentei lui Post.

PCP modificata (MPCP): Input: un alphabet V cu cel putin doua simboluri, $n \geq 2$ si doua liste Post x, y.
Output: Exista $k \geq 1$ si $1 \leq i_1, i_2, \ldots, i_k \leq n$ a.i.
$x_1 x_{i_1} x_{i_2} \ldots x_{i_k} = y_1 y_{i_1} y_{i_2} \ldots y_{i_k}$?
Propozitie. Daca PCP este decidabila, atunci MPCP este decidabila.
Dem. Fie $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$, si $\beta = (\beta_1, \beta_2, \ldots, \beta_n)$, doua liste Post peste alphabetul V. Vom folosi algoritmul pentru PCP pentru a arata ca $MPCP(\alpha, \beta)$ este decidabila.
Construim doua liste Post x, y peste alphabetul $V \cup \{\#, \$\}$ astfel:

$$x = (x_0, x_1, x_2, \ldots, x_n, x_{n+1}), y = (y_0, y_1, y_2, \ldots, y_n, y_{n+1}),$$

$$\begin{aligned} x_0 &= \# x_1, x_i = h_1(\alpha_i), 1 \leq i \leq n, x_{n+1} = \$, \\ y_0 &= y_1, y_i = h_2(\beta_i), 1 \leq i \leq n, y_{n+1} = \#\$, \\ h_1(a) &= a\#, \quad h_2(a) = \#a, a \in V. \end{aligned}$$

Exemplu: $\alpha = (b, a, aba, bb)$, atunci
$x = (\#b\#, b\#, a\#, a\#b\#a\#, b\#b\#, \$)$.

Problema Corespondentei lui Post.

Afirmatie. $MPCP(\alpha, \beta)$ are solutie ddaca $PCP(x, y)$ are solutie. Fie $1, i_1, \ldots, i_k$ o solutie pentru $MPCP(\alpha, \beta)$, deci:

$$\alpha_1 \alpha_{i_1} \ldots \alpha_{i_k} = \beta_1 \beta_{i_1} \ldots \beta_{i_k}.$$

Atunci: $x_0 x_{i_1} x_{i_2} \ldots x_{i_k} x_{n+1} = y_0 y_{i_1} y_{i_2} \ldots y_{i_k} y_{n+1}$, deci $PCP(x, y)$ are solutie.

Reciproc, $PCP(x, y)$ are solutie. Atunci primul indice este 0 iar ultimul este $n+1$. Prin stergerea simbolului $\#$ se obtine o solutie pentru $MPCP(\alpha, \beta)$.

TEOREMA. Problema MPCP nu este decidabila.

DEM. PpA MPCP este decidabila. Vom construi doua liste Post α, β a.i. $MPCP(\alpha, \beta)$ are solutie ddaca o masina Turing data se opreste pe o intrare oarecare a sa.

Fie $M = (Q, V, U, \delta, q_0, B, F)$ o masina Turing determinista si $w \in V^*$ o intrare a sa.

Problema Corespondentei lui Post.

Lista α	Lista β	Conditie	Grup
$\#$	$\#q_0w\#$	Fara conditii	0
a	a	pentru orice $a \in U \setminus \{B\}$	
$\$$	$\$$		
$\#$	$\#$		1
Pentru orice $q, p \in Q,\ a, b, c \in U \setminus \{B\}$:			
qa	bp	daca $\delta(q, a) = (p, b, R)$	
cqa	pcb	daca $\delta(q, a) = (p, b, L)$	
$q\#$	$bp\#$	daca $\delta(q, B) = (p, b, R)$	
$cq\#$	$pcb\#$	daca $\delta(q, B) = (p, B, L)$	2
Pentru orice $q \in Q$ si $a \in U \setminus \{B\}$:			
qa	$\$$	daca $\delta(q, a)$ nedefinita	
$q\#$	$\$\#$	daca $\delta(q, B)$ nedefinita	3

Problema Corespondentei lui Post.

Lista α	Lista β	Conditie	Grup
$a\$b$	$\$$	pentru orice $a, b \in U \setminus \{B\}$	
$\$b$	$\$$	pentru orice $b \in U \setminus \{B\}$	
$a\$$	$\$$	pentru orice $a \in U \setminus \{B\}$	4
$\$\#\#$	$\#$	fara conditii	5

Afirmatie. $MPCP(\alpha, \beta)$ are solutie ddaca M se opreste pe intrarea w.
Dem. Fie $C_0, C_1, \ldots, C_n, \ldots$ secventa de configuratii pe intrarea w. Atunci, pentru orice $j \geq 0$, listele partiale α si β vor arata astfel:

$$\begin{aligned} \alpha &: \#C_0\#C_1\#\ldots\#C_{j-1}\# \\ \beta &: \#C_0\#C_1\#\ldots\#C_{j-1}\#C_j\#. \end{aligned}$$

Problema Corespondentei lui Post.

Inductie dupa j: $j \rightarrow j+1$. Fie $C_j : xqay$ si $\delta(q,a) = (p,b,R)$. Atunci $C_{j+1} : xbpy$ (celelalte cazuri se trateaza similar).

$$\begin{aligned} \alpha &: \#C_0\#C_1\#\ldots\#C_{j-1}\# \\ \beta &: \#C_0\#C_1\#\ldots\#C_{j-1}\#xqay\# \end{aligned}$$

Problema Corespondentei lui Post.

Inductie dupa j: $j \to j+1$. Fie $C_j : xqay$ si $\delta(q,a) = (p,b,R)$. Atunci $C_{j+1} : xbpy$ (celelalte cazuri se trateaza similar).

$$\begin{aligned}\alpha &: \#C_0\#C_1\#\ldots\#C_{j-1}\# \\ \beta &: \#C_0\#C_1\#\ldots\#C_{j-1}\#xqay\#\end{aligned}$$

$$\begin{aligned}\alpha &: \#C_0\#C_1\#\ldots\#C_{j-1}\#x \\ \beta &: \#C_0\#C_1\#\ldots\#C_{j-1}\#xqay\#x\end{aligned}$$

Problema Corespondentei lui Post.

Inductie dupa j: $j \to j+1$. Fie $C_j : xqay$ si $\delta(q,a) = (p,b,R)$. Atunci $C_{j+1} : xbpy$ (celelalte cazuri se trateaza similar).

$$\begin{aligned}\alpha &: \#C_0\#C_1\#\ldots\#C_{j-1}\# \\ \beta &: \#C_0\#C_1\#\ldots\#C_{j-1}\#xqay\#\end{aligned}$$

$$\begin{aligned}\alpha &: \#C_0\#C_1\#\ldots\#C_{j-1}\#x \\ \beta &: \#C_0\#C_1\#\ldots\#C_{j-1}\#xqay\#x\end{aligned}$$

$$\begin{aligned}\alpha &: \#C_0\#C_1\#\ldots\#C_{j-1}\#xqa \\ \beta &: \#C_0\#C_1\#\ldots\#C_{j-1}\#xqay\#xbp\end{aligned}$$

Problema Corespondentei lui Post.

Inductie dupa j: $j \to j+1$. Fie $C_j : xqay$ si $\delta(q,a) = (p,b,R)$. Atunci $C_{j+1} : xbpy$ (celelalte cazuri se trateaza similar).

$$\begin{aligned} \alpha &: \#C_0\#C_1\#\ldots\#C_{j-1}\# \\ \beta &: \#C_0\#C_1\#\ldots\#C_{j-1}\#xqay\# \end{aligned}$$

$$\begin{aligned} \alpha &: \#C_0\#C_1\#\ldots\#C_{j-1}\#x \\ \beta &: \#C_0\#C_1\#\ldots\#C_{j-1}\#xqay\#x \end{aligned}$$

$$\begin{aligned} \alpha &: \#C_0\#C_1\#\ldots\#C_{j-1}\#xqa \\ \beta &: \#C_0\#C_1\#\ldots\#C_{j-1}\#xqay\#xbp \end{aligned}$$

$$\begin{aligned} \alpha &: \#C_0\#C_1\#\ldots\#C_{j-1}\#xqay\# \\ \beta &: \#C_0\#C_1\#\ldots\#C_{j-1}\#xqay\#xbpy\#. \end{aligned}$$

Problema Corespondentei lui Post.

Deci, M nu se opreste pe w implica $MPCP(\alpha, \beta)$ nu are solutie.
Pp. ca M se opreste la pasul $j+1$.

$$\begin{array}{lcl} \alpha & : & \#C_0\#C_1\#\ldots\#C_{j-1}\#xqa \\ \beta & : & \#C_0\#C_1\#\ldots\#C_{j-1}\#xqay\#x\$ \end{array}$$

Problema Corespondentei lui Post.

Deci, M nu se opreste pe w implica $MPCP(\alpha, \beta)$ nu are solutie.
Pp. ca M se opreste la pasul $j+1$.

$$\begin{aligned}\alpha &: \#C_0\#C_1\#\dots\#C_{j-1}\#xqa\\ \beta &: \#C_0\#C_1\#\dots\#C_{j-1}\#xqay\#x\$\end{aligned}$$

$$\begin{aligned}\alpha &: \#C_0\#C_1\#\dots\#C_{j-1}\#xqay\#\\ \beta &: \#C_0\#C_1\#\dots\#C_{j-1}\#xqay\#x\$y\#.\end{aligned}$$

Folosim valorile listelor α, β din grupul 4 pana cand ajungem in situatia

$$\begin{aligned}\alpha &: \#C_0\#C_1\#\dots\#C_{j-1}\#xqay\#\dots\#\\ \beta &: \#C_0\#C_1\#\dots\#C_{j-1}\#xqay\#x\$y\dots\#\$\#.\end{aligned}$$

Folosim valorile din grupul 5:

$$\begin{aligned}\alpha &: \#C_0\#C_1\#\dots\#C_{j-1}\#xqay\#\dots\#\$\#\#\\ \beta &: \#C_0\#C_1\#\dots\#C_{j-1}\#xqay\#x\$y\dots\#\$\#\#.\end{aligned}$$

Problema Corespondentei lui Post.

Particularizari

1. Alfabetul are o singura litera.

Problema Corespondentei lui Post.

Particularizari

1. Alfabetul are o singura litera.(Decidabila)
2. Lungimea listelor este cel mult 2.

Problema Corespondentei lui Post.

Particularizari

1. Alfabetul are o singura litera.(Decidabila)
2. Lungimea listelor este cel mult 2.(Decidabila)
3. Lungimea listelor este cel putin 7.

Problema Corespondentei lui Post.

Particularizari

1. Alfabetul are o singura litera.(Decidabila)
2. Lungimea listelor este cel mult 2.(Decidabila)
3. Lungimea listelor este cel putin 7.(Nedecidabila)
4. Lungimea listelor intre 3 si 6.

Problema Corespondentei lui Post.

Particularizari

1. Alfabetul are o singura litera.(Decidabila)
2. Lungimea listelor este cel mult 2.(Decidabila)
3. Lungimea listelor este cel putin 7.(Nedecidabila)
4. Lungimea listelor intre 3 si 6. (Nu se stie)
5. PCP marginita.

Problema Corespondentei lui Post.

Particularizari

1. Alfabetul are o singura litera.(Decidabila)
2. Lungimea listelor este cel mult 2.(Decidabila)
3. Lungimea listelor este cel putin 7.(Nedecidabila)
4. Lungimea listelor intre 3 si 6. (Nu se stie)
5. PCP marginita.(NP-completa)
6. Fiecare lista are toate elementele distincte.

Problema Corespondentei lui Post.

Particularizari

1. Alfabetul are o singura litera.(Decidabila)
2. Lungimea listelor este cel mult 2.(Decidabila)
3. Lungimea listelor este cel putin 7.(Nedecidabila)
4. Lungimea listelor intre 3 si 6. (Nu se stie)
5. PCP marginita.(NP-completa)
6. Fiecare lista are toate elementele distincte.(Nedecidabila)
7. PCP 1-marcata (nu exista 2 elemente care incep cu aceeasi litera)

Problema Corespondentei lui Post.

Particularizari

1. Alfabetul are o singura litera.(Decidabila)
2. Lungimea listelor este cel mult 2.(Decidabila)
3. Lungimea listelor este cel putin 7.(Nedecidabila)
4. Lungimea listelor intre 3 si 6. (Nu se stie)
5. PCP marginita.(NP-completa)
6. Fiecare lista are toate elementele distincte.(Nedecidabila)
7. PCP 1-marcata (nu exista 2 elemente care incep cu aceeasi litera)(Decidabila, PSPACE)
8. PCP 2-marcata.

Problema Corespondentei lui Post.

Particularizari

1. Alfabetul are o singura litera.(Decidabila)
2. Lungimea listelor este cel mult 2.(Decidabila)
3. Lungimea listelor este cel putin 7.(Nedecidabila)
4. Lungimea listelor intre 3 si 6. (Nu se stie)
5. PCP marginita.(NP-completa)
6. Fiecare lista are toate elementele distincte.(Nedecidabila)
7. PCP 1-marcata (nu exista 2 elemente care incep cu aceeasi litera)(Decidabila, PSPACE)
8. PCP 2-marcata.(Nedecidabila).