Masina Turing vs. alte modele de calcul

1. Masina Turing vs. alte modele de calcul
2. Automatul linear marginit
3. Problema lui POST

Masina Turing vs. alte modele de calcul

<u>Observaţia 1</u>

O MT poate să accepte un limbaj care poate fi acceptat şi de un model de calculabilitate mai slab: un automat finit, un automat pushdown etc. ?

De exemplu, poate o MT să recunoască un limbaj regulat ?

<u>Teorema 1</u>

Limbajul

REG_{MT} = { *<M> | M este o MT oarecare şi L(M) este un limbaj regulat* }

este nedecidabil.

Masina Turing vs. alte modele de calcul

demonstraţie

Ppa. $\exists$ R$\in$MT care decide asupra REG_{MT}

Construim, cu ajutorul R, o MT A care să decidă asupra limbajului ACC_{MT} :

A = “Fie secvenţa de intrare <M,w>, unde M$\in$MT şi w$\in\Sigma^*$:

1. Se construieşte următoarea $M_2 \in$MT :

   M_2 = “Fie secventa de intrare x$\in\Sigma^*$:

   1. Dacă x este de forma 0^n1^n atunci M_2 acceptă x.
   2. Altfel, se rulează M pe intrarea w şi, dacă M acceptă w atunci M_2 acceptă x.”
2. Se rulează R pe intrarea <M_2>.
3. Dacă R acceptă /respinge <M_2> atunci A acceptă / respinge <M,w>.

Masina Turing vs. alte modele de calcul

Fie P o proprietate oarecare netriviala a lb. r.e.

=> problema verificarii faptului ca limbajul acceptat de o MT data poseda proprietatea P NU este rezolvabila algoritmic.

Teorema lui Rice

Fie limbajul *P* = {<*M*> | *M este o MT oarecare*}.

Presupunem ca limbajul P satisface următoarele două proprietaţi:

(i) P nu este trivial

(adică: propr. avuta in vedere are loc pentru unele limbaje r.e. iar pentru altele nu are loc,

sau inca: $(\exists)\ M_1, M_2 \in MT$ astfel încât $\langle M_1 \rangle \in P$ şi $\langle M_2 \rangle \notin P$)

(ii) P este o proprietate a limbajelor acceptate de MT

(adica: aparteneţa unei MT M la P (prin descrierea ei <M>) depinde numai de limbajul acceptat de M,

sau inca: $\forall M_1, M_2 \in MT$ cu proprietatea $L(M_1) = L(M_2)$ atunci $\langle M_1 \rangle \in P \Leftrightarrow \langle M_2 \rangle \in P$).

Atunci, limbajul P este nedecidabil.

Masina Turing vs. alte modele de calcul

demonstratie

Analog: ppa limbajul P avand cele 2 proprietati este decidabil

=> $\exists R_P \in$ MT care decide asupra limbajului P.

Construim cu ajutorul lui R_P o alta MT, A, care sa decida asupra ACC_{MT}.

Pt aceasta: fie $R_\varnothing \in$ MT care respinge orice cuvant,

Putem pp ca $< R_\varnothing > \notin P$.

Cf. ip (i): $\exists$ cel putin o MT, R, a.i. $<R> \in P$.

=> putem construi A folosind capacitatea MT R_P de a distinge intre MT $R_\varnothing$ si R

Masina Turing vs. alte modele de calcul

A = “Fie secvenţa de intrare <M,w>, unde $M \in MT$ şi $w \in \Sigma^*$:

1. Se construieşte MT M_w, cu ajutorul descrierii lui M si w::

   M_w = “Fie secventa de intrare $x \in \Sigma^*$:

   2. Se simuleaza M pe intrarea w.

      Daca M se opreste si respinge w, atunci M_w respinge x;

      daca M accepta w, atunci se trece la Pasul 4.

   3. Se simuleaza R pe intrarea x.

      Daca R accepta x atunci M_w accepta x.”

4. Se utilizeaza MT R_P pt a determina daca $< M_w > \in P$.

   Daca da, atunci A accepta <M,w>, altfel respinge.”

Masina Turing vs. alte modele de calcul

Definitia MT, A → MT, M_w simuleaza R daca M accepta w, adica:

$$\lim bajul\ L(M_w) = \begin{cases} L(R), & daca\ M\ \ accepta\ w; \\ \phi, & altfel. \end{cases}$$

Am convenit ca <R>$\in$P => (cf (ii)) <M_w>$\in$P => (def.A) M accepta w => <M,w>$\in ACC_{MT}$.

=> contradictie.

Masina Turing vs. alte modele de calcul

1. Masina Turing vs. alte modele de calcul
2. Automatul linear marginit
3. Problema lui POST

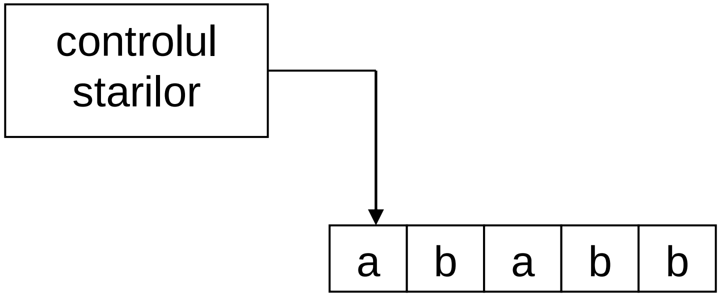
Masina Turing vs. alte modele de calcul

Definitia 1

Un automat linear marginit (ALM) este un tip restrictionat de MT in care cursorul nu se poate deplasa pe banda de lucru inafara zonei care contine secventa de intrare.

Daca functia de tranzitie a MT impune deplasarea cursorului dincolo de oricare dintre cele 2 capete ale secventei de intrare, atunci acesta este fortat sa ramana pe loc.

Masina Turing vs. alte modele de calcul



Observatia 2

a) Un ALM = o MT cu memorie limitata

=> poate rezolva numai probleme care necesita un spatiu de memorie egal cu cel necesar memorarii secventei de intrare.

b) $\Gamma \supset \Sigma \rightarrow$ spatiul de memorie disponibil pt ALM creste cu un factor constant

=>spunem ca spatiul de memorie creste linear cu dimensiunea intrarii.

c) ALM: memorie limitata dar putere de calcul mare:

MT pentru ACC_{AFD}, ACC_{GIC}, EMP_{AFD}, EMP_{GIC} sunt ALM.

d) ACC_{ALM} este identica cu problema nedecidabila ACC_{MT} dar este decidabila.

Masina Turing vs. alte modele de calcul

<u>Definitia 2</u>

Configuratie a unei $M \in MT$ = un triplet format din:

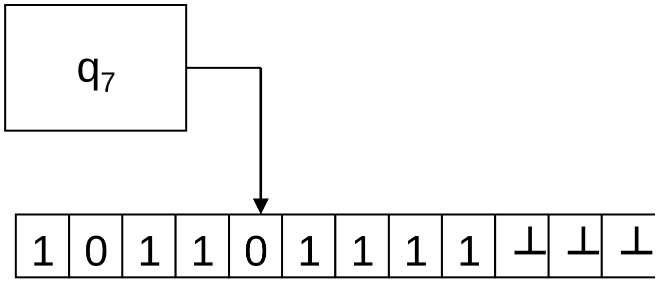
- starea curenta a M, q;
- continutul curent al benzii, $v \cdot w$;
- pozitia curenta a cursorului.

<u>Notatie</u>

vqw, $v,w \in \Gamma^*$, $q \in Q$.

<u>Exemplul 1</u>

Configuratia $1011q_7 01111$ inseamna



Masina Turing vs. alte modele de calcul

Definitia 3

Configuratia C_1 produce configuratia C_2 $\Leftrightarrow$

MT trece – corect – din C_1 in C_2 intr-un singur pas $\Leftrightarrow$

Fie $a,b,c \in \Gamma$,

$v,w \in \Gamma^*$,

$q_i, q_j \in Q$

Spunem ca o configuratie $C_1=vaq_ibw$ produce

- configuratia $C_2=vq_jacw$ daca $\delta(q_i,b)=(q_j,c,L)$;
- configuratia $C_2=vacq_jw$ daca $\delta(q_i,b)=(q_j,c,R)$.

Masina Turing vs. alte modele de calcul

<u>Lema 1</u>

Fie $M \in ALM$: $|Q|=q$, $|\Sigma|=s$

=> numarul de configuratii distincte ale M pentru o banda de lucru de lungime n este

$$q \cdot n \cdot s^n.$$

demonstratie

O configuratie este un pas de calcul efectuat de M pe intrarea primita; ea consta din starea controlului, pozitia cursorului si continutul benzii.

cf. ip.: M poate fi in una dintre cele q stari,
lungimea benzii este n,
secventele citite sunt compuse din s simboluri

=> pe banda se pot afla s^n secvente distincte

=> numarul total de configuratii este dat de produsul celor 3 factori $q \cdot n \cdot s^n$.

Masina Turing vs. alte modele de calcul

<u>Teorema 2</u>

Limbajul

$$ACC_{ALM} = \{<M,w> \mid M \in ALM,\ w \in \Sigma^* \text{ şi } M \text{ accepta intrarea } w\}$$

este decidabil.

Masina Turing vs. alte modele de calcul

demonstraţie

L = “Fie secventa de intrare <M,w>, unde M$\in$ALM si w$\in\Sigma^*$

1. Se simuleaza M pe intrarea w timp de $q{\cdot}n{\cdot}s^n$ pasi sau pana ce M se opreste.
2. Daca M s-a oprit, atunci: daca M a acceptat, L accepta iar daca M a respins, L respinge.
3. Daca M nu s-a oprit atunci L respinge.”

Observatia 3

MT si ALM difera in mod esential: ACC_{ALM}= decidabil dar ACC_{MT}=nedecidabil.

Totusi, alte probleme raman nedecidabile si pt ALM.

Masina Turing vs. alte modele de calcul

Metoda istoricului calculului

- reductibilitatea PROBLEMEI ACCEPTABILITATII pt MT, ACC_{MT}, la anumite limbaje
- testarea existentei unui obiect sau a unei proprietati.

Exemplul 2

Problema a 10-a a lui Hilbert: existenta radacinilor intregi pentru p, $\forall$p=polinom.

Istoricul unui calcul efectuat de o MT pe o secventa de intrare data = sirul de configuratii prin care respectiva MT trece atunci cand proceseaza intrarea respectiva.

Masina Turing vs. alte modele de calcul

<u>Definitia 4</u>

Fie $M \in MT$ si $w \in \sum^*$.

Istoricul unui calcul de acceptare efectuat de M pe w este o secventa de configuratii $C_1, C_2, \ldots, C_f$, unde:

- C_1 este configuratia de start a lui M pentru w,
- C_f este o configuratie finala de acceptare a lui M si
- $\forall\ 2 \leq i \leq f$: C_i se obtine corect din C_{i-1} prin aplicarea regulilor de tranzitie ale M.

Istoricul unui calcul de respingere: C_f este o configuratie finala de respingere a lui M

<u>Observatia 4</u>

Secventa de configuratii: FINITA

Masina Turing vs. alte modele de calcul

Teorema 3

Limbajul

$EMP_{ALM} = \{<M> \mid M \in ALM$ *şi* $L(M)=\varnothing\}$

este nedecidabil.

ideea demonstraţiei

Folosim metoda reducerii la absurd si reductibilitatea de la problema acceptabilitatii.

Fie $M \in MT$ si $w=w_1w_2\ldots w_n \in \Sigma^*$; construim un $B \in ALM$ astfel:

- L(B) consta din toate istoricele calculelor de acceptare efectuate de M asupra w;
- daca M accepta w atunci limbajul $L(B) \neq \varnothing$ (consta dintr-o singura secventa); daca M nu accepta w, atunci $L(B)=\varnothing$.

=> Daca am putea determina daca $L(B)= \varnothing$ → am putea determina daca M accepta w si am obtine contradictia cautata.

Masina Turing vs. alte modele de calcul

Fie $M \in MT$ si $w \in \Sigma^*$; vom demonstra existenta $B \in ALM$ si vom descrie modul in care o MT poate obtine o descriere a lui B pornind de la descrierile lui M si w:

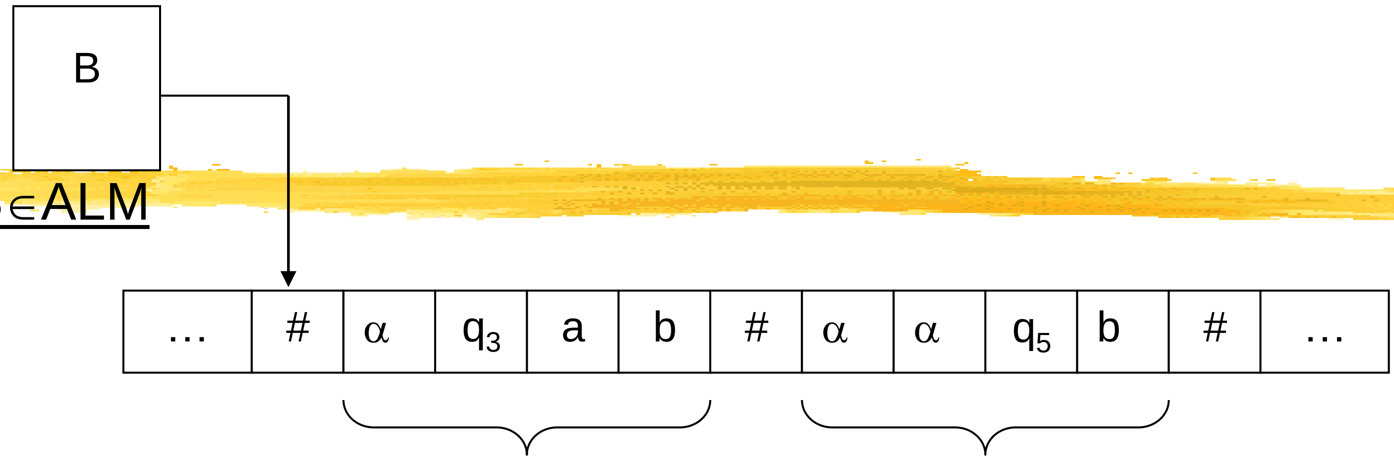
B accepta secventa de intrare x $\Leftrightarrow$ x este un istoric al unui calcul de acceptare al M pe w.

=>presupunem ca un istoric al unui calcul de acceptare se prezinta sub forma unei secvente unice:

$$\#\underbrace{}_{C_1}\#\underbrace{}_{C_2}\#\underbrace{}_{C_3}\#\ldots\#\underbrace{}_{C_f}\#$$

Masina Turing vs. alte modele de calcul

<u>Modul de lucru al B∈ALM</u>



1) B primeste secventa de intrare x;
2) B verifica daca x reprezinta un istoric al unui calcul de acceptare al M pe w:
   - B divizeaza secventa x in secventele C_1, C_2, …,C_f, in conformitate cu delimitatorii #;
   - B verifica daca acestea satisfac cele 3 conditii din definitia unui istoric de calcul:
     - C_1 este configuratia de start a lui M pe w:

dar $C_1=q_0w_1w_2…w_n$; aceasta informatie este cuprinsa in chiar descrierea <M,w> => B o poate verifica direct;

Masina Turing vs. alte modele de calcul

- C_f=configuratie finala de acceptare a lui M pe w:

dar C_f trebuie sa contina starea q_a => B trebuie sa scaneze secventa C_f pt a determina prezenta lui q_a;

- pentru fiecare pereche de configuratii adiacente, B verifica daca

C_{i+1} provine, corect, din C_i:

Aceasta verificare presupune:

- testarea faptului ca C_i si C_{i+1} sunt identice, cu exceptia locatiilor din C_i aflate sub cursor adiacente acestuia;
- actualizarea acestor locatii cf. functiei de tranzitie a lui M;
- executarea unei miscari de “du-te vino” intre locatiile care se corespund din C_i si C_{i+1}, pentru a verifica daca actualizarea s-a facut corect;
- punctarea simbolului din locatia curent verificata, pentru a tine evidenta pozitiei curente a cursorului in cele 2 configuratii in timpul acestei pendulari.

3) Daca cele 3 conditii sunt satisfacute, B incheie prin a accepta intrarea primita.

Masina Turing vs. alte modele de calcul

demonstratie

Fie R∈MT care decide asupra limbajului $EMP_{ALM;}$

Definim S∈MT care decide asupra limbajului ACC_{MT} astfel:

S = “Fie secventa de intrare <M,w>, unde M∈MT, w∈Σ*:

1. Se construieste B∈ALM pe baza descrierilor lui M si w, conform indicatiilor de mai sus.
2. Se ruleaza R pe intrarea <B>.
3. Daca R respinge, atunci S accepta; daca R accepta, atunci S respinge.”

Observatie

R accepta <B> → L(B)=∅ → M nu are niciun istoric de calcul de acceptare pt. w → M nu accepta w => S respinge <M,w>.

R respinge <B> → L(B)≠∅ : singura secventa pe care o poate accepta B este un istoric de calcul de acceptare al M pt w → M trebuie sa accepte w => S accepta <M,w>.

Masina Turing vs. alte modele de calcul

Teorema 4

Limbajul

$ALL_{GIC} = \{<G> \mid G \in GIC \text{ şi } L(G) = \Sigma^* \}$

este nedecidabil.

ideea demonstraţiei

Folosim metoda reducerii la absurd si reductibilitatea de la problema acceptabilitatii precum si tehnica istoricului calculului efectuat pt un cuvant.

Fie $M \in MT$ si $w \in \Sigma^*$, oarecare: sa pp ca reprezentam un istoric al calculului de acceptare al M pe w prin $\#C_1\#C_2\#...\#C_f\#$, unde C_i este configuratia lui M la pasul i din calculul lui w.

Masina Turing vs. alte modele de calcul

construim o $G \in GIC$ care genereaza toate cuvintele din $\Sigma^* \Leftrightarrow$

$\Leftrightarrow$ M nu accepta w.

Daca M accepta w $\rightarrow$ (cf. def. G de mai sus) $\exists x \in \Sigma^*$: $x \notin L(G)$; acest cuvint este tocmai istoricul calculului de acceptare al M pe w.

=> G este construita astfel incat sa genereze toate cuvintele care nu sunt istorice ale calculelor de acceptare ale M pe w.

<u>Observatii</u>

1) $x \in \Sigma^*$ nu poate fi un istoric de calcul de acceptare $\Leftrightarrow$

$\Leftrightarrow$ nu indeplineste cel putin una dintre cele 3 conditii din Definitia 2, adica:

(1') nu incepe cu C1;

(2') nu se termina cu o configuratie finala de acceptare;

(3') $\exists\ 1 \leq i \leq f-1$: configuratia C_i nu produce corect, in conformitate cu regulile de tranzitie din M, configuratia C_{i+1}.

2) in loc sa construim o $G \in GIC$ vom construi un $D \in APD$ deoarece exista o procedura algoritmica de convertire a unui APD intr-o GIC si un APD este – in cazul nostru - mai usor de construit.

Masina Turing vs. alte modele de calcul

Astfel, D va incepe prin “ a ghici” nedeterminist pe care dintre cele 3 reguli sa le verifice; => in arborele de derivare va exista un nod din care pornesc 3 ramuri de calcul:

- pe prima ramura se verifica daca secventa de intrare incepe cu configuratia initiala, C_1 si accepta in cazul negativ;
- pe a 2a ramura se verifica daca secventa de intrare se incheie cu configuratia finala de accepatre, C_f si accepta in cazul negativ (verificarea revine la identificarea prezentei lui q_a in C_f);
- pe a 3a ramura se verifica daca exista vreo configuratie C_i care nu produce corect configuratia adiacenta C_{i+1}. Se procedeaza astfel:

    - D scaneaza intrarea $\#C_1\#C_2\#...\#C_f\#$ pana decide in mod nedeterminist ca a ajuns la configuratia C_i;

    - D introduce C_i in stiva, pana la delimitatorul #;

    - D extrage din stiva pt a face comparatia cu C_{i+1}; cele 2 secvente ar trebui sa coincida, mai putin in dreptul si in jurul cursorului unde relatia dintre ele ar trebui sa respecte definitia functiei de tranzitie a lui M;

    - D ar trebui sa accepte daca exista o nepotrivire intre configuratii sau daca actualizarea nu a respectat definitia lui δ_M.

Masina Turing vs. alte modele de calcul

Problema prezenta in aceasta strategie este ca extragerea C_i din stiva se face in ordinea inversa introducerii, ceea ce impiedica compararea ei cu C_{i+1}.

Pt a evita aceasta dificultate, vom reprezenta de la bun inceput secventa de intrare formata din sirul de configuratii $C_1, C_2, \ldots, C_f$, astfel:

$$\#\underbrace{\qquad}_{C_1}\#\underbrace{\qquad}_{C_2^R}\#\underbrace{\qquad}_{C_3}\#\underbrace{\qquad}_{C_4^R}\#\ \ldots\ \#\underbrace{\qquad}_{C_f}\#$$

=> acum D poate introduce in stiva configuratia a.i. atunci cand o extrage sa o poata compara cu configuratia urmatoare.

Proiectam $D \in APD$ a.i. sa accepte orice secventa care nu este un istoric de calcul de acceptare in forma modificata de mai sus.

Masina Turing vs. alte modele de calcul

1. Automatul linear marginit
2. Masina Turing vs. alte modele de calcul
3. Problema corespondentei lui Post

Masina Turing vs. alte modele de calcul

Definim o multime de dominouri astfel:

$$\left\{\left[\frac{b}{ca}\right],\left[\frac{a}{ab}\right],\left[\frac{ca}{a}\right],\left[\frac{abc}{c}\right]\right\}$$

Putem intotdeauna forma o secventa din dominourile din multime a.i. "secventa numaratorilor" sa coincida cu "secventa numitorilor" (i.e. $\exists$ intotdeauna o "coincidenta"?)

Exemplu:

$$\left\{\left[\frac{a}{ab}\right],\left[\frac{b}{ca}\right],\left[\frac{ca}{a}\right],\left[\frac{a}{ab}\right],\left[\frac{abc}{c}\right]\right\} \Rightarrow abcaaabc$$

Contraexemplu:

$$\left\{\left[\frac{abc}{ab}\right],\left[\frac{ca}{a}\right],\left[\frac{acc}{ba}\right]\right\}$$

=> Problema corespondentei lui Post (PCP) revine la a determina, pentru orice multime de dominouri, daca prezinta o coincidenta sau nu.

Masina Turing vs. alte modele de calcul

Formalizam PCP astfel:

- o instanta a PCP este o colectie P de dominouri definita prin:

$$P = \left\{ \left[\frac{t_1}{b_1} \right], \left[\frac{t_2}{b_2} \right], \ldots, \left[\frac{t_n}{b_n} \right] \right\}.$$

- o coincidenta este o secventa de indici $i_1, i_2, \ldots, i_k$ astfel incat:

$$t_{i_1} t_{i_2} \ldots t_{i_k} = b_{i_1} b_{i_2} \ldots b_{i_k} .$$

- PCP consta in a determina daca o colectie oarecare de dominouri P prezinta o coincidenta.

Teorema 5

Limbajul

{ <*P*> | *P este o instanta a PCP care prezinta o coincidenta* }.

este nedecidabil.

Masina Turing vs. alte modele de calcul

1. Masina Turing vs. alte modele de calcul
2. Automatul linear marginit
3. Problema corespondentei lui Post