Modifikácie Turingovho stroja

Argumentom podporujúcim CT hypotézu je aj robustnosť TS.

Modifikácie

- TS, ktorý má po skončení výpočtu zapísaný na páske len vstupný a výstupný reťazec
- TS s jednosmerne nekonečnou páskou
- TS s dvojrozmernou páskou
- TS s konečným počtom pások (každá más svoju čítaciu/zapisovaciu hlavu)

Fakt

Všetky uvedené modifikácie sú vzájomne ekvivalentné

Dôkaz

technikou simulácie: ukážeme, že výpočet jedného zariadenia sa dá simulovať na druhom zariadení a naopak

Programy s počítadlami (*Counter Programs, CP*)

- programy manipulujú s prirodzenými číslami uloženými v premenných
- tri elementárne operácie

  $X \leftarrow 0$ priradí premennej hodnotu 0

  $X \leftarrow Y + 1$

  $X \leftarrow Y - 1$ ak hodnota Y je 0, tak X priradí hodnotu 0
- jeden elementárny riadiaci príkaz

$$\textbf{if} X = 0 \textbf{ goto } G,$$

  kde X je premenná a G je návestie pripojené k príkazu

Programy s počítadlami - príklad

$$
\begin{array}{ll}
 & U \leftarrow 0 \\
 & Z \leftarrow 0 \\
\mathrm{A}: & \textbf{if} \;\; X = 0 \;\; \textbf{goto} \;\; \mathrm{G} \\
 & X \leftarrow X - 1 \\
 & V \leftarrow Y + 1 \\
 & V \leftarrow V - 1 \\
\mathrm{B}: & \textbf{if} \;\; V = 0 \;\; \textbf{goto} \;\; \mathrm{A} \\
 & V \leftarrow V - 1 \\
 & Z \leftarrow Z + 1 \\
 & \textbf{if} \;\; U = 0 \;\; \textbf{goto} \;\; \mathrm{B}
\end{array}
$$

vykonanie **goto** G je ekvivalentom úspešného ukončenia výpočtu

Turingove stroje a programy s počítadlami

TS manipulujú so symbolmi, PS s číslami

je možná ich vzájomná simulácia?

Simulácia Turingovho stroja programom s počítadlami

obsah pásky $\longrightarrow$ čísla

pre jednoduchosť predpokladajme, že abeceda TS má desať znakov
znaky očíslujeme a reťazce znakov prevedieme na čísla

Príklad

$$\begin{array}{ccccc} \# - 0 & ! - 1 & * - 2 & a - 3 & b - 4 \\ c - 5 & d - 6 & e - 7 & f - 8 & g - 9 \end{array}$$

reťazec

$$\ldots \# \# \; a \; b \; * \; e \; b \; ! \; \# \; a \; g \; a \; \# \; \# \ldots$$

v ktorom je snímaný symbol b, prevedieme na dvojicu čísel

$$3427 \; a \; 393014$$

Simulácia Turingovho stroja programom s počítadlami

zmena symbolu na páske

zmena poslednej číslice v druhom čísle

Príklad ak symbol b prepíšeme symbolom g, tak číslo 393014 *sa zmení na* 393019, *PC pripočíta 5 krát hodnotu 1*

posun hlavy doprava

prvé číslo vynásobíme 10 a pripočítame k nemu poslednú číslicu druhého čísla

druhé číslo vydelíme 10 (celočíselne)

analogicky pre posun hlavy doľava

zmena stavu

stavu TS zodpovedá skupina inštrukcií CP; zmena stavu je simulovaná skokom na novú skupinu inštrukcií (príkaz goto)

CP, ktorý simuluje TS, má len dve počítadlá!

Simulácia programu s počítadlami Turingovym strojom

čísla $\longleftarrow$ symboly

hodnota každej premennej je zapísaná ako postupnosť symbolov = číslic; jednotlivé hodnoty sú vzájomne oddelené špeciálnym symbolom (napr. *)

inštrukcie $\longleftarrow$ stavy

každej inštrukcii programu zodpovedá skupina stav TS, vykonenie inštrukcie je simulované prechodom do príslušného stavu a realizácia postupnosti krokov, ktoré potrebným spôsobom upravia obsah premennej

Simulácie ako redukcie

Ak model A simuluje model B, tak máme redukciu medzi týmito modelmi.

Redukcia prevedie program modelu A a jeho vstup X na program modelu B a jeho vstup Y.

CT hypotéza ukazuje, že naše úvahy pri konštrukcii redukcií boli korektné.

Simulácie ako redukcie

Ak model A simuluje model B, tak máme redukciu medzi týmito modelmi.

Redukcia prevedie program modelu A a jeho vstup X na program modelu B a jeho vstup Y.

CT hypotéza ukazuje, že naše úvahy pri konštrukcii redukcií boli korektné.

Príklad nerozhodnuteľnosť problému zastavenia

1. *formálne dokážeme nerozhodnuteľnosť problému pre TS*
2. *podľa CT hypotézy problém zastavenia nemôže byť rozhodnuteľný ani pre žiaden iný programovací jazyk vyššetj úrovne (je ekvivalentý TS!)*

Fenomén

algoritmus, ktorého vstupom je iný algoritmus

Univerzálny algoritmus

jeden z najdôležitejších dôsledkov CT hypotézy

Existencia univerzálneho algoritmu, ktorý má schopnosť chovať sa ako akýkoľvek iný algoritmus.

- vstupom pre univerzálny algoritmus je popis akéhokoľvek algoritmu A a akéhokoľvek jeho vstupu X
- univerzálny algoritmus simuluje výpočet A na X
- výpočet univerzálneho algoritmu sa zastaví práve ak výpočet A na X sa zastaví; ako výstup poskytne univerzálny algoritmus presne tú istú odpoveď ako poskytne A na X

ak fixujeme algoritmus A a meníme X, tak univerzálny algoritmus sa chová presne ako algoritmus A

Univerzálny algoritmus

- môže byť vstupom univerzálneho algoritmu program v akomkoľvek programovacom jazyku?
- využijeme CT hypotézu a poznatok o ekvivalencii všetkých známych formalizmov pre popis algoritmov
- ku konštrukcii univerzálneho algoritmu potrebujeme len jazyk L_1, v ktorom napíšeme program U pre univerzálny algoritmus; program akceptuje ako vstup ľubovoľný program napísaný vo fixovanom konkrétnom jazyku L_2
- program U je nezávislý na výbere modelu, pretože podľa CT hypotézy
  1. môže byť napísaný v akomkoľvek jazyku
  2. dokáže simulovať akýkoľvek algoritmus popísaný v akomkoľvek jazyku

Vhodným kandidátom pre jazyky L_1 a L_2 sú Turingove stroje.

Univerzálny Turingov stroj

- potrebujeme popísať Turingov stroj ako lineárny reťazec nad konečnou abecedou symbolov
- stačí linearizovať prechodový diagram
- *mark ** mark YES* $\langle \#/\#, L\rangle$ ** mark* $move_a$ $\langle a/\#, R\rangle$ *** $move_a$ $move_a$ $\langle a/a/, R\rangle$ **. . .*
- linearizovaný prechodový diagram prevedieme štandartným spôsobom na reťazec nad fixovanou abecedou (napr. binárnou)
- podobne linearizujeme a kódujeme aj vstup simulovaného TS
- samotný program univerzálneho TS je jednoduchý svojim princípom: uchováva si aktuálny stav simulovaného TS, obsah jeho pásky a čítaný symbol; z linearizovaného popisu simulovaného TS odvodí, aké akcie sa majú realizovať v ďalšom kroku výpočtu simulovaného TS

Univerzálny program s počítadlami

- vstupom je dvojica čísel; prvé číslo je kódom nejakého programu s počítadlami, druhé číslo je kódom jeho vstupu
- univerzálny program je možné skonštruovať tak, aby využíval len dve počítadlá

Modifikované programy s počítadlami

Motivácia

- TS manipuluje jednom kroku výpočtu s jedným symbolom pásky (*s jednou číslicou čísla*)
- CP mení jednou inštrukciou hodnotu premennej o 1 (*exponenciálne menej efektívne v porovnaní s TS*)
- narovnanie diskrepancie
- CP musí mať možnosť k číslu pridať alebo odobrať číslicu v konštantnom čase

Modifikácia

množinu inštrukcií CP rozšírime o 2 nové inštruckie

$X \leftarrow X \times 10$

$X \leftarrow X/10$ celočíselné delenie

Polynomiálna redukcia

Existencia redukcií medzi programovacími jazykmi vyššej úrovne (dostatočne silnými výpočtovými modelmi) ukazuje, že trieda rozhodnuteľných problémov je invariantná voči voľbe jazyka (modelu).

Otázka

Aká je zložitosť redukcie?

Fakt

Ak oba modely manipulujú s číslami v inej než unárnej sústave, tak redukcia má polynomiálnu časovú zložitosť .

Zložitosť redukcií medzi TS a modifikovanými CP

Zložitosť výpočtu

TS počet krokov výpočtu

CP počet vykonaných inštrukcií

Zložitosť výpočtu je funkciou dĺžky vstupu; hodnota funkcie pre argument N zhora ohraničuje zložitosť výpočtov na všetkýh vstupoch dĺžky N.
Dĺžkou vstupu pre TS je počet znakov vstupného reťazca, dĺžkou vstupu pre CP je počet číslic počiatočných hodôt premenných.

Redukcia TS $\longrightarrow$ modifikované CP

krok výpočtu je simulovaný zmenou hodnoty každého počítadla; zmena je realizovateľná konštantným počtom inštrukcií

Redukcia modifikované CP $\longrightarrow$ TS

každá inštrukcia je simulovaná konštantným počtom krokov

TS a modifikované CP sú polynomiálne ekvivalentné

Polynomiálna redukcia - dôsledky

Nech výpočtové modely A a B sú polynomiálne ekvivalentné.

Ak algoritmický problém P je riešiteľný na A s časovou zložitosťou $\mathcal{O}(f(N))$ (f je funkcia dĺžky vstupu), tak existuje program pre B, ktorý rieši problém P a jeho časová zložitosť je $\mathcal{O}(p(f(N)))$, pričom p je nejaká (fixovaná) polynomiálna funkcia.

Naopak, ak P je riešiteľný na B v čase $\mathcal{O}(g(N))$, tak existuje program pre A, ktorý rieši P s časovou zložitosťou $\mathcal{O}(q(f(N)))$, pričom q je nejaká (fixovaná) polynomiálna funkcia.

Ak TS rieši problém v polynomiálnom čase, tak aj modifikový CP rieši tento problém v polynomiálnom čase (a naopak).
Ak neexistuje polynomiálny TS pre daný problém, tak neexistuje ani polynomiálny modifikoaný CP pre tento problém.

Robustnosť triedy prakticky riešiteľných problémov

CT hypotéza ukazuje robustnosť pojmu rozhodnuteľný problém. Polynomiálna ekvivalencia zjemňuje toto pozorovanie na prakticky riešiteľné problémy.

Sekvenčná výpočtová hypotéza

Pojem prakticky riešiteľného problému je robustný, tj. je nezávislý na konkrétnej voľbe výpočtového modelu resp. programovacieho jazyka.

Hypotéza sa nevzťahuje na modely s neohraničeným zdrojom paralelizmu, preto sa označuje ako "sekvenčná".

Triedy P, NP, PSPACE, EXPTIME sú robustné

Triedy s lineárnou časovou zložitosťou nie sú robustné.

Nedeterministické Turingove stroje

pre rozhodovacie problémy

- v prechodovom diagrame je povolené, aby s jedného stavu vychádzal ľubovoľný počet hrán označených zhodým spínačom (*symbolom, ktorý sa číta*)
- stroj má možnosť výberu, ktorý z prechodov použije
- pre vstup X dá TS odpoveď "Áno" (*akceptuje*) práve ak existuje taká postupnosť výberu prechodov, pre ktorú výpočet skončí v koncovom stave *YES*
  (*stroj uváži všetky možné výpočty na X a akceptuje X práve ak aspoň jeden z výpočtov skončí v stave YES*)
- v opačnom prípade, tj. ak žiaden výpočet neskončí v stave *YES*, dá odpoveď "Nie"

Nedeterministické TS sú ekvivalentné (deterministickým) TS.

P=NP? problém - revízia

Formálna definícia tried P a NP

Trieda P (NP) obsahuje rozhodovacie problémy, ktoré sú riešiteľné Turingovymi strojmi (nedeterministickými TS) s polynomiálnou časovou zložitosťou.

P=NP? problém

Sú deterministické a nedeterministické Turingove stroje polynomiálne ekvivalentné?

P=NP? problém - revízia

Definícia NP-ťažkého a NP-úplného probému

Rozhodovací problém sa nazýva NP-ťažký ak každý problém z triedy P je na neho polynomiálne redukovateľný.

Rozhodovací problém sa nazýva NP-úplný ak je NP-ťažký a naviac patrí do triedy NP.

Fakty

Ak nejaký NP-úplný problém patrí do triedy P, tak P $=$ NP.

Ak P $\neq$ NP, tak žiaden NP-úplný problém nie je riešiteľný algoritmom polynomiálnej zložitosti.

Turingove stroje a dolné odhady zložitosti problémov

- dôkaz nerozhodnuteľnosti problému
  redukcia problému o ktorom je už dokázané, že je nerozhodnuteľný, na problém o ktorom chceme dokázať, že je nerozhodnuteľný
  príklad redukcia problému zastavenia na problém domina
- dôkaz vzťahu medzi zložitostnými triedami
  metóda diagonalizácie
  príklady P $\subset$ EXPTIME, PSPACE $\subset$ EXPSPACE
- dôkaz, že problém nepatrí do zložitostnej triedy
  dôsledok úplnosti
  príklad žiaden EXPTIME-úplný problém nepatrí do triedy P

pre dôkaz horných odhadov zložitosti problémov nie sú TS vhodné; naopak je vhodné použiť programovací jazyk vyššej úrovne

Redukcia problému zastavenie na problém domina

problém domina
: úlohou je pokryť hornú polovicu nekonečnej plochy s podmienkou, že prvá dlaždica v T (nazveme ju t) je umiestnená niekde v spodnom riadku

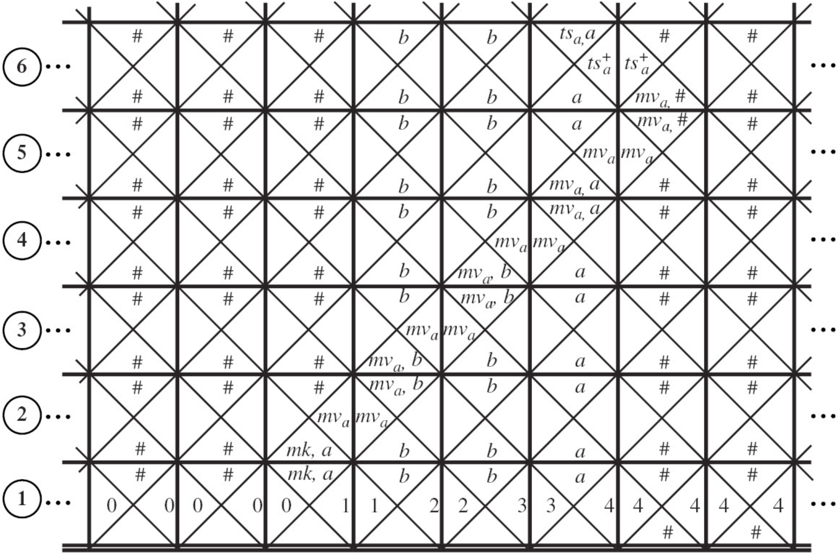
problém zastavenia
: odpoveď "Áno" pre vstup $\langle M, X \rangle$ taký, že výpočet M na X sa nezastaví

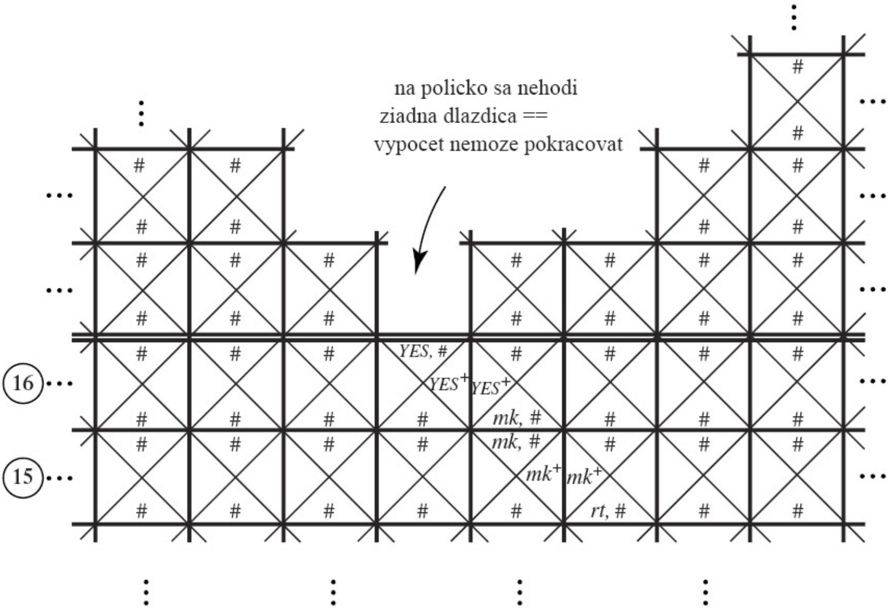
Redukcia problému zastavenie na problém domina

Redukcia

- **Vstup** dvojica $\langle M, X\rangle$
- **Výstup** množina typov dlaždíc T a dlaždica t

Princíp konštrukcie $\langle T, t\rangle$ pokrytie dlaždicami korešponduje s výpočtom; pokrytie nekonečnej plochy je možné len v prípade existencie nekonečného výpočtu


6
5
4
3
2
1


na policko sa nehodi
ziadna dlazdica ==
vypocet nemoze pokracovat
YES, #
YES+
YES+
mk, #
mk, #
mk+
mk+
rt, #
16
15