Bonyolultságelmélet

Monday 26th September, 2016, 19:04

Univerzális RAM-gép

Dacára egyszerű szintaxisának, a RAM-gép is Turing-teljes számítási modell: minden „hagyományos” programozási nyelven megvalósítható algoritmusra létezik RAM program is.

Így például RAM interpretert is írhatunk RAM nyelven. . .

Tétel

Létezik olyan U univerzális RAM-program, amelynek ha adott egy M RAM-program és annak egy x bemenete, úgy viselkedik, mint M az x-en, vagyis $U(M; x) = M(x)$.

Természetesen M kódolva adott U számára (utasítások, tömbcímzések. . . számokkal).

A megállási probléma – első eldönthetetlen problémánk

Megállás

- Input: M program, x input.
- Output: Megáll-e M az x-en futtatva?

Tétel

A Megállás probléma eldönthetetlen.

Bizonyítás

Tegyük fel, hogy Megállás eldönthető egy M_0 programmal.
Legyen D a következő (Ram-programot váró) program:

$$D(M) = \text{if } M_0(M;M) \text{ then } \nearrow \text{ else halt.}$$

Ekkor:

$$D(D) = \nearrow \Leftrightarrow \; M_0(D;D) = \text{„igen"} \; \Leftrightarrow \; D(D) \neq \nearrow,$$

ami ellentmondás.

Visszavezetés: eldönthetetlenség bizonyítása

Ha már ismerünk egy eldönthetetlen problémát, másokról is meg tudjuk mutatni, hogy eldönthetetlenek.

Hogyan?

Azt használjuk fel, hogy ha

- A eldönthetetlen és
- $A \leq_{\mathcal{R}} B$,

akkor B is eldönthetetlen.

Mindenen Megállás

Állítás

Az alábbi probléma algoritmikusan eldönthetetlen.

- Adott: M RAM-program.
- Kérdés: M megáll-e minden bemenetén?

Bizonyítás

Adott M és x esetén legyen M_x olyan program, hogy az M_x minden y inputjára:

$$M_x(y) \;=\; M(x).$$

(azaz: M_x tetszőleges y input esetén futtassa M-et x-en.)
Így

$$M(x) \neq \nearrow \Leftrightarrow \;\; M_x \text{ megáll minden bemenetén.}$$

Ez az $M; x \mapsto M_x$ hozzárendelés kiszámítható, tehát rekurzív visszavezetés; és mivel az eldönthetetlen Megállás problémát visszavezettük erre a problémára, ez is eldönthetetlen.

Üres Inputon Megállás

Legyen Üres Inputon Megállás a következő probléma:

- Input: egy M program.
- Output: megáll-e M az üres inputon (azaz mikor minden input regisztert 0-val inicializálunk)?

Eldönthetetlen!

Visszavezetjük rá a Megállás problémát.
Adott M-hez és x-hez legyen M_x az a program, ami tetszőleges y inputra

- ha y az üres input, akkor futtatja M-et x-en;
- egyébként mondjuk megáll.

Ez az M_x az $M; x$ párból elkészíthető és nyilván

$$M \text{ megáll } x\text{-en} \iff M_x \text{ megáll üres inputon.}$$

Tehát Megállás $\leq_{\mathcal{R}}$ Üres Inputon Megállás, így ez utóbbi is eldönthetetlen probléma.

Mit láttunk eddig?

A Megállás, Mindenen Megállás és Üresszón Megállás problémák eldönthetetlensége szerint...

- nincs olyan algoritmus, mely inputként kap egy Java forráskódot és hozzá egy inputot, és megválaszolja, hogy a megadott program megáll-e a megadott inputon;
- nincs olyan algoritmus, mely inputként kap egy forráskódot és megválaszolja, hogy a program eshet-e végtelen ciklusba egyáltalán, vagy akár csak azt, hogy paraméter nélkül elindítva végtelen ciklusba fog-e esni!

A helyzet azonban ennél sokkal rosszabb...

Rekurzív felsorolhatóság

Definíció

Egy A probléma rekurzívan felsorolható (vagy Turing-felismerhető), ha van olyan M RAM-program, amire

$$M(x) = \begin{cases} \text{ACCEPT} & \text{, ha } x \in A \\ \nearrow & \text{, egyébként.} \end{cases}$$

A rekurzívan felsorolható problémák osztályát $\mathbf{RE}$ jelöli.

Nyilván $\mathbf{R} \subseteq \mathbf{RE}$. (Hiszen egy RAM programot ACCEPT-től különböző válasz helyett végtelen ciklusba egyszerű küldeni.)

Rekurzív felsorolhatóság

Tétel

$$\mathbf{R} \subsetneq \mathbf{RE}.$$

Bizonyítás

Az eldönthetetlen MEGÁLLÁS probléma rekurzívan felsorolható, pl. azzal a RAM-programmal, mely futtatja az univerzális RAM-programot az input $M; x$ páron, majd ha az megáll, ACCEPT választ ad.

A „rosszabb helyzet”: Rice tétele

Rice tétele

A rekurzívan felsorolható problémák egyetlen nemtriviális tulajdonsága sem dönthető el algoritmikusan.

Vagyis: ha $\emptyset \neq \mathcal{C} \subsetneq \mathbf{RE}$ a rekurzívan felsorolható problémák egy nemtriviális osztálya, akkor a következő probléma eldönthetetlen: adott egy M program, igaz-e, hogy $L(M) \in \mathcal{C}$?

Másképp mondva: automatikusan semmi nemtriviálisat nem tudunk eldönteni egy program viselkedéséről a forráskódjának ismeretében.

Ezért válnak szükségessé például kommentek, tervezési minták használáta, statikus és dinamikus tesztelés. . .

További eldönthetetlen problémák

Hilbert 10. problémája

- Adott: $p(x_1, \ldots, x_n)$ egész együtthatós polinom.
- Kérdés: Létezik-e egész értékű zérushelye?

Tétel (Matijasevič)

Hilbert 10. problémája algoritmikusan megoldhatatlan.

Post megfelelkezési problémája

- Adott: $u_1, v_1, \ldots, u_n, v_n \in \Sigma^*$.
- Kérdés: Létezik-e olyan $i_1, \ldots, i_k$ $(k > 0,\ 1 \leq i_j \leq n)$ sorozat, hogy

$$u_{i_1} \ldots u_{i_k} = v_{i_1} \ldots v_{i_k} \ ?$$

Tétel (Post)

A fenti probléma algoritmikusan megoldhatatlan.

Post megfelelkezési problémája

Példa

Ha pl. $\Sigma = \{0, 1\}$ és a szavak

$$\begin{aligned} u_1 &= 01 & v_1 &= 0 \\ u_2 &= 110010 & v_2 &= 0 \\ u_3 &= 1 & v_3 &= 1111 \\ u_4 &= 11 & v_4 &= 01, \end{aligned}$$

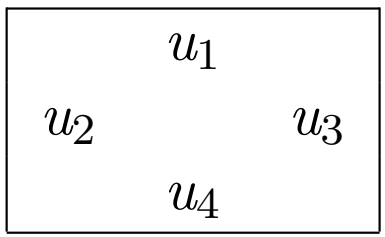
van-e megoldás?

Van, pl. $1, 3, 2, 4, 4, 3$:

$$\begin{aligned} u_1 u_3 u_2 u_4 u_4 u_3 &= 01 \cdot 1 \cdot 110010 \cdot 11 \cdot 11 \cdot 1 &&= 01111001011111, \\ v_1 v_3 v_2 v_4 v_4 v_3 &= 0 \cdot 1111 \cdot 0 \cdot 01 \cdot 01 \cdot 1111 &&= 01111001011111. \end{aligned}$$

Egy dominó probléma

- Adott: Dominó típusok véges halmaza.
- Kérdés: Kirakható-e ezekkel a dominókkal az egész sík hézagmentesen?

Típus: 

, $u_i \in \Sigma^*$.

A dominók nem forgathatóak.

Tétel

A dominó probléma algoritmikusan megoldhatatlan.

Egy megoldatlan probléma

- Adott: m pozitív egész szám.
- Kérdés: Kongruens-e m?

Tehát azt kérdezzük, hogy létezik-e olyan derékszögű háromszög, mely oldalai racionális hosszúságúak és melynek területe m.

Példa

1, 2, 3, 4 nem kongruensek. 5 kongruens (Fibonacci):

$$a = \frac{3}{2},\ b = \frac{20}{3},\ c = \frac{41}{6}$$

Nem ismert, hogy a probléma algoritmikusan eldönthető-e.

Állítás

Ha A rekurzív, akkor $\overline{A}$ is az.

Ötlet

Az Accept és a Reject sorok cseréjével.

Állítás

Egy A probléma pontosan akkor rekurzív, ha A és $\overline{A}$ rekurzívan felsorolhatók.

Ötlet

- A rekurzív $\Rightarrow$ A rekurzívan felsorolható
  A rekurzív $\Rightarrow$ $\overline{A}$ rekurzív, így $\overline{A}$ rekurzívan felsorolható
- Tegyük fel, hogy A és $\overline{A}$ rekurzívan felsorolhatók. Ekkor A felismerhető egy M, $\overline{A}$ pedig egy $\overline{M}$ programmal. Szimuláljuk M-et és $\overline{M}$-et párhuzamosan!

Definíció

Legyen $\mathcal{C}$ problémák egy osztálya. Ekkor

$$\mathbf{co}\mathcal{C} \;=\; \{\overline{L} : L \in \mathcal{C}\}.$$

Példák

- co$\mathbf{R}$: azon problémák, melyek komplementere rekurzív
- co$\mathbf{RE}$: azon problémák, melyek komplementere rekurzívan felsorolható

$\mathbf{R}$, $\mathbf{RE}$ és co$\mathbf{RE}$ viszonya

Következmény

$$\begin{aligned} \mathbf{R} &= \text{co}\mathbf{R} \\ \mathbf{RE} &\neq \text{co}\mathbf{RE} \\ \mathbf{R} &= \mathbf{RE} \cap \text{co}\mathbf{RE} \end{aligned}$$

Nemdeterminizmus

Láttunk rá bizonyítékot, hogy minden „realisztikus” számítási modell szimulálható RAM-gépen, lényegi időigény-romlás nélkül.

Most bevezetünk egy nemrealisztikus számítási modellt.

Egy nemdeterminisztikus RAM-programban

- több különböző programsor is kaphatja ugyanazt a sorszámot
- egy lehetséges futás minden lépésében a PC által kijelölt sorszámú lehetséges utasítások egyike hajtódik végre
- (utána a PC eggyel nő vagy ugrás esetén a megfelelő értékre áll be)

A program elfogadja az inputot, ha van olyan lehetséges futás, mely ACCEPT utasítással terminál, egyébként elutasítja azt

`https://www.youtube.com/watch?v=lufECeWtN34`

Nemdeterminisztikus generálás

Egy bit

1. $a := 0$
1. $a := 1$
2. $\ldots$

Egy n-bites szám

function ND(n)

1. $r := 0$
2. if $n == 0$ **return** r
3. $r := r + r$
4. $\boldsymbol{r := r + 1}$
4. $\boldsymbol{r := r + 0}$
5. $n := n - 1$
6. goto 2

Példa: Hamilton-Út

- Input: $\vec{G} = (V, E)$ (irányított) gráf. Feltehető, hogy $V = \{1, \ldots, n\}$.
- Output: Van-e $\vec{G}$-ben Hamilton-út (azaz minden csúcsot érintő út)?

for $i := 1 \ldots n$ **do**
    $W[i] :=$ND$(1 + \lfloor \log n \rfloor)$
for $i := 1 \ldots n - 1$ **do**
    if $(W[i], W[i+1]) \notin E$ **then** Reject
for $i := 1 \ldots n$ **do**
    for $j := 1 \ldots n$ **do**
        if $W[j] == i$ **then** break
    if $j > n$ **then** Reject
Accept

Nemdeterminisztikus időigény

Egy nemdeterminisztikus program időigénye $f(n)$, ha bármely n méretű inputon tetszőleges lehetséges futás $f(n)$ lépésben véget ér.

Hamilton-út időigénye

Az előző fólia algoritmusának időigénye például négyzetes:

- nemdeterminisztikusan generál n darab, egyenként $\approx \log n$-bites számot, ez $n \log n$ lépés;
- (determinisztikusan) ellenőrzi, hogy az n szám ebben a sorrendben valóban egy séta-e a gráfban, ez n lépés;
- (determinisztikusan) ellenőrzi, hogy minden csúcs előfordul-e a sétában, ez n^2 lépés.

Megjegyzés

Az utolsó ellenőrzés hatékonyabban is elvégezhető, most a pszeudokód egyszerűségét tartottuk szem előtt.

Példa: SAT

- Input: konjunktív normálformájú formula
- Output: kielégíthető-e?

Algoritmus

- minden változónak nemdeterminisztikusan beállítjuk az értékét 1-re vagy 0-ra (lineáris időigény);
- determinisztikusan ellenőrizzük, hogy a kapott kiértékelés kielégítő-e (lineáris időigény);
- ha igen, ACCEPT, egyébként REJECT választ adunk.

Példa: 3 – Színezés

- Input: $G = (V, E)$ (irányítatlan) gráf.
- Output: kiszínezhető-e G helyesen, vagyis adhatunk-e minden csúcsnak egy-egy színt egy háromelemű színhalmazból, hogy szomszédos csúcsok különböző színt kapjanak?

Algoritmus

- minden csúcsnak nemdeterminisztikusan adunk egy színt az $\{R, G, B\}$ halmazból;
- determinisztikusan ellenőrizzük, hogy a kapott színezés helyes-e;
- ha igen, Accept, egyébként Reject választ adunk.

Az **NP** osztály

Az **NP** osztályba mindazok a problémák tartoznak, melyek eldönthetők polinom időigényű nemdeterminisztikus programmal.

Azaz, $L \in \mathbf{NP}$, ha van olyan M polinom időkorlátos nemdeterminisztikus program, melyre

- ha $x \in L$, akkor M-nek létezik elfogadó futása x-en, és
- ha $x \notin L$, akkor M-nek minden futása elutasítja x-et.

Példa

Hamilton-Út, Sat és $3-$Színezés **NP**-beli problémák.

Mivel minden $f(n)$ időigényű program felfogható $f(n)$ időigényű nemdeterminisztikus programként is, így nyilván $\mathbf{P} \subseteq \mathbf{NP}$.

Összefoglalás

- Megismertük az univerzális programot.
- Megismertük a megállási problémát és láttuk, hogy eldönthetetlen.
- Megismertük a Mindenen Megállás problémát és visszavezetéssel láttuk, hogy szintén eldönthetetlen.
- Megismertük az Üres Inputon Megállás problémát és visszavezetéssel láttuk, hogy szintén eldönthetetlen.
- Megismertük, mit jelent, hogy egy probléma rekurzívan felsorolható (avagy Turing-felismerhető, félig eldönthető).
- **RE**: a rekurzívan felsorolható problémák osztálya.
- Láttuk, hogy a megállási probléma rekurzívan felsorolható, és ezért $\mathbf{R} \subsetneq \mathbf{RE}$.
- Rice tétele: a rekurzívan felsorolható problémák egyetlen nemtriviális tulajdonsága se dönthető el.
- Megismertük Hilbert tizedik problémáját, Post megfelelkezési problémáját, melyek szintén csak félig eldönthetők.
- co$\mathcal{C}$: a $\mathcal{C}$-beli problémák komplementereinek osztálya.
- Láttuk, hogy $\mathbf{R} = \mathbf{RE} \cap \mathrm{co}\mathbf{RE}$, $\mathbf{R} = \mathrm{co}\mathbf{R}$ és $\mathbf{RE} \neq \mathrm{co}\mathbf{RE}$.
- Megismertük a nemdeterminisztikus algoritmusokat és azok időigényét.
- **NP**: a nemdeterminisztikusan polinomidőben eldönthető problémák osztálya.
- Láttuk, hogy Hamilton-Út, Sat és $3 -$ Színezés **NP**-beliek.
- Láttuk, hogy $\mathbf{P} \subseteq \mathbf{NP}$.