Turing-gépek: példák, nem kiszámítható problémák

Hajnal Péter

Bolyai Intézet, TTIK, SZTE, Szeged

2020. ősz

A PALINDROM nyelv

Definíció

Legyen

$$PALINDROM = \{\omega = \omega_1, \ldots \omega_n : \text{ahol } \omega_i = \omega_{n+1-i} \text{ minden } i \in \{1, 2, \ldots, n\} \text{ esetén}\}$$

a palindrom szavak nyelve.

Tehát döntési problémával állunk szemben. Adott egy szó, el kell döntenünk, hogy előlről és hátulról olvasva ugyanazt olvasuk-e. Így nem kell outputszalag, ezt az S állapothalmaznak ELVET és az ELFOGAD eleme helyettesíti.

Két Turing-gépet/algoritmust vázolunk. Ennek során elmondjuk, hogyan néznek ki a Turing-gépről készített pillanatfelvételek és szemezgetünk az állapothalmazból. Az egyszerűség kedvéért feltesszük, hogy $\Sigma = \{0, 1\}$.

Első modell: Egy szalag

- Az első egy algoritmus egy egyszalagos Turing-gép lesz a $\Gamma = \{0, 1, 0^{\checkmark}, 1^{\checkmark}\}$ munkaábécével.

Első algoritmus/TG

• A START állapotból egyet jobbra lép az input szem/kéz (a szalaghatároló jel utáni első mező, az input első karaktere felett lesz). Az új állapot ELŐL-PIPÁL lesz.

• A karaktert „megjegyzi", felülírja a pipált változatával „hogy emlékezzen, ezt a karaktert már ellenőrizte".
// Ez magyarázza meg az állapot nevét.
A megjegyzéshez a HÁTUL-TESZT-0, HÁTUL-TESZT-1 állapotokat használjuk.

• Ezután megkeresi az utolsó inputkaraktert.
// Ez magyarázza meg az állapot nevét.
A kereséskor az input szem/kéz folyamatosan jobbra mozog. Ez a mozgás akkor áll le, amikor az inputszalagon az $\triangleleft$ jel olvasható.

Első algoritmus/TG (folytatás)

• Ekkor egyet visszalép (ezt a későbbiekben egy kissé felülírjuk). HÁTUL-TESZT-0-MOST, HÁTUL-TESZT-1-MOST állapotba kerül. Az állapot bitje az előlről hozott „emlékezet".

• Ekkor a keresett karakter felett áll és teszteli, hogy megegyezik-e az első karakterrel:
Ha a látott bit nem egyezik a hozott emlékezettel, akkor a gép ELVET állapotba kerül, leáll.
Ha egyezik, akkor balra lép és HÁTUL-PIPÁL állapotba kerül.

• Ebben az állapotban felülírja a karaktert a pipált változatával. Egyet balra lép. Az olvasott karaktertől függően ELŐL-TESZT-0, ELŐL-TESZT-1 állapotba kerül.

• Ebben az állapotban balra megy, amíg el nem ér egy pipált karaktert és ennek elérésekor visszalép.

Első algoritmus/TG (folytatás)

• Ezzel ELŐL-TESZT-0-MOST és ELŐL-TESZT-1-MOST állapotok egyikébe kerül, ahol a bit a „hátulról hozott emlékezet".

• Vagy leállunk, vagy pipálunk és egyet jobbra lépünk. Az itt látott karakter alapján HÁTUL-TESZT-0, HÁTUL-TESZT-1 állapotok egyikébe kerülünk. Az állapothalmaz egy kis memóriát szimulál.

• Korábban azt mondtuk, hogy ekkor az utolsó input karaktert keressük meg. Most pontosítunk: folyamatosan jobbra mozog az utolsó pipálatlan karakterig. Azaz a mozgás akkor áll le, amikor az inputszalagon az $\triangleleft$ jel vagy pipált jel található, majd egyet visszalép.

• Ha az HÁTUL-TESZT-0, HÁTUL-TESZT-1, ELŐL-TESZT-0, ELŐL-TESZT-1, HÁTUL-PIPÁL, ELŐL-PIPÁL állapotba jutáskor egy pipált karaktert látunk, akkor ELFOGAD állapotba kerülünk: Ez a helyzet/konfiguráció azt jelenti, hogy a bal és jobb pipák „összeértek", inputunk palindrom szó.

Első algoritmus/TG: Állapothalmaz

A Turing-gép eddigi munkájához következő állapothalmazt használtuk

$S = \{$START, HÁTUL-PIPÁL, HÁTUL-TESZT-0, HÁTUL-TESZT-1, HÁTUL-TESZT-0-MOST, HÁTUL-TESZT-1-MOST, ELŐL-PIPÁL ELŐL-TESZT-0, ELŐL-TESZT-1, ELŐL-TESZT-0-MOST, ELŐL-TESZT-1-MOST, ELFOGAD, ELVET$\}$.

Első algoritmus/TG: Átmenetifüggvény (töredék)

$$(\mathit{START}, \triangleright) \mapsto (\text{ELŐL-PIPÁL}, *, J)$$

$$(\text{ELŐL-PIPÁL}, 0) \mapsto (\text{HÁTUL-TESZT-0}, 0^{\checkmark}, J)$$

$$(\text{ELŐL-PIPÁL}, 1) \mapsto (\text{HÁTUL-TESZT-1}, 1^{\checkmark}, J)$$

$$(\text{HÁTUL-TESZT-1}, 0) \mapsto (\text{HÁTUL-TESZT-1}, 0, J)$$

$$(\text{HÁTUL-TESZT-1}, \triangleleft) \mapsto (\text{HÁTUL-TESZT-1-MOST}, *, B)$$

$$(\text{HÁTUL-TESZT-1-MOST}, 0) \mapsto (\mathit{ELVET}, *, *)$$

$$(\text{HÁTUL-TESZT-1-MOST}, 1) \mapsto (\text{HÁTUL-PIPÁL}, 1, B)$$

Első algoritmus/TG: Átmenetifüggvény (töredék)

$$(\text{HÁTUL-PIPÁL}, 1) \mapsto (\text{ELŐL-TESZT-1}, 1^{\checkmark}, B)$$

$$(\text{HÁTUL-PIPÁL}, 1^{\checkmark}) \mapsto (\text{ELFOGAD}, *, *)$$

$$(\text{ELŐL-TESZT-1-MOST}, 0^{\checkmark}) \mapsto (\text{ELFOGAD}, *, *)$$

$$(\text{ELŐL-TESZT-1-MOST}, \triangleright) \mapsto (\text{ELFOGAD}, *, *)$$

$$(\text{START}, 0) \mapsto \text{„Kit érdekel?”}$$

$$(\text{ELŐL-TESZT-1-MOST}, \triangleleft) \mapsto \text{„Kit érdekel?”}$$

$$(\text{START}, \triangleleft) \mapsto \text{„Kit érdekel?”}$$

Első algoritmus/TG: A fej mozása, idő

Ha ω egy palindrom szó, akkor a fenti $\mathcal{A}_1$ algoritmus/gép futásának hosszát könnyen kiszámíthatjuk.

A fej mozgását könnyű leírni. Egy „csillapodó ingamozgás" lesz.

Tétel

$$TIME(\mathcal{A}_1, \omega) = \mathcal{O}(|\omega|^2)$$

A konstansokat nem számoltuk ki. Lényegtelen is, az állapotok számának növelésével a futási idő csökkenthető. Például használhatnánk a HÁTUL-TESZT-000, HÁTUL-TESZT-001, HÁTUL-TESZT-010, HÁTUL-TESZT-011, HÁTUL-TESZT-100, HÁTUL-TESZT-101, HÁTUL-TESZT-110, HÁTUL-TESZT-111 állapotokat input bitek hármasainak együttes tesztelésére és kevesebbet kellene utaznia a fejnek, „gyorsabban csillapodna az inga".

Második modell: Standard egy munkaszalagos

A standard modellt használjuk, $\Gamma = \Sigma = \{0, 1\}$.

Második algoritmus/TG

Az inputot átmásoljuk a munkaszalagra. Ennek befejezése, amikor az input vége karaktert olvassuk.

Ebből mindkét szem balra egyet lép (a munka szem/kéz az átmásolt sorozat utolsó karaktere felett lesz), továbbá az INPUTFEJ-ELŐRE állapotba jut a gép.

Ekkor a munkaszalag felett a szem/kéz mozdulatlan, az inputszalag feletti szem folyamatosan jobbra mozog. Egész addig, amíg a $\triangleright$ jelet nem látja.

Innen TESZT állapotba jut a gép az input szem eggyel jobbra mozgatása után (ekkor az input szem az input első karaktere fölé kerül, közben az output szem az átmásolt input utolsó karaktere felett maradt végig).

Második algoritmus/TG (folytatás)

A TESZT állapotban mindig ellenőrzi a gép, hogy a két szem ugyanazt látja-e. Ha valamikor ez nem teljesül, akkor ELVET állapotba jut, különben az input szem egyet jobbra, a munka szem egyet balra mozdul.

Ez addig történik, amíg az input szem az input végét jelző karaktert nem látja (ekkor szükségszerű, hogy a munka szem a munkaszalag kezdetét jelző karakter felett legyen.

Ha ez megtörténik, akkor a gép ELFOGAD állapotba kerül.

Második algoritmus/TG: Az állapothalmaz

Az állapothalmaz legyen

$S = \{$START, MÁSOLOK, INPUTFEJ-ELŐRE, TESZT, ELFOGAD, ELV

Második algoritmus/TG: Az átmeneti függvény

$$(START, \triangleright, \triangleright) \mapsto (\text{MÁSOLOK}, J, *, J)$$

$$(\text{MÁSOLOK}, 0, \smile) \mapsto (\text{MÁSOLOK}, J, 0, J)$$

$$(\text{MÁSOLOK}, 1, \smile) \mapsto (\text{mÁSOLOK}, J, 1, J)$$

$$(\text{MÁSOLOK}, \triangleleft, \smile) \mapsto (\text{INPUTFEJ-ELŐRE}, B, *, B)$$

$$(\text{INPUTFEJ-ELŐRE}, 0, 0) \mapsto (\text{INPUTFEJ-ELŐRE}, B, 0, .)$$

$$(\text{INPUTFEJ-ELŐRE}, \triangleright, 0) \mapsto (\text{TESZT}, B, 0, .),$$

Második algoritmus/TG: Az átmeneti függvény

$$(\text{TESZT}, 1, 1) \mapsto (\text{TESZT}, J, *, B),$$

$$(\text{TESZT}, 1, 0) \mapsto (\text{ELVET}, *, *, *),$$

$$(\text{TESZT}, \lhd, \rhd) \mapsto (\text{ELFOGAD}, ., *, .).$$

Második algoritmus/TG: Az idő

Észrevétel

Minden ω inputra a futás hossza legfeljebb $3|\omega| + 3 = \mathcal{O}(|\omega|)$.

Pontosabban

$$TIME(\omega; T) = \Theta(|\omega|)$$

Ez az eredmény a nagyságrend szempontjából éles.

Szünet

Az egyszalagon modell rossz

Tétel

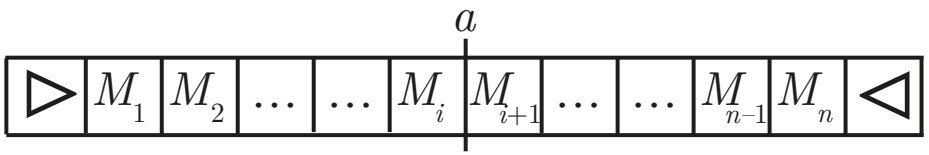
Ha T egy olyan Turing-gép, amely az egyszalagos modellben eldönti a *PALINDROM* nyelvet, akkor $\forall\, n$, $\exists\, \omega \in \Sigma^n$:

$$\mathit{TIME}(\omega, T) \geq \alpha_T |\omega|^2,$$

valamely α_T pozitív konstansra.

Bizonyítás: Fogalmak

Az inputszalag két szomszédos mezőjének közös határát *ajtónak* nevezzük. Ha az inputszalag mezőit úgy képzeljük el, mint végtelen egymásba nyíló szobasorozatot, akkor azt mondhatjuk, hogy a fej csak az ajtókon át tud közlekedni.



Az M_i és M_{i+1} mezőket elválasztó a ajtó.

Egy futás

Tekintsük egy T Turing-gép ω inputon való futását. Ekkor egy

$$\kappa_0(\omega) \to \kappa_1 \to \kappa_2 \to \ldots \to \kappa_\ell$$

konfigurációsorozatot kapunk, ahol

$$\ell := \min\{n \mid T \text{ állapota } \kappa_n\text{-ben } \textit{ELFOGAD} \text{ vagy } \textit{ELVET}\}$$

az időpont, amelyben eldöntjük, hogy ω eleme-e a *PALINDROM* nyelvnek.

Ez az időpont biztosan véges, hiszen a *PALINDROM* nyelv az eldönthető nyelvek osztályába tartozik (ezt igazolják a korábban említett T_1, T_2 algoritmusok).

Egy futás egy ajtónál állva nézve

Jelölés

Most vegyük azokat a κ_j, κ_{j+1} konfigurációkat, amelyekben az inputszem egyszer az M_i, másszor az M_{i+1} mezőt nézi. Jelölje s_j a mezőket elválasztó a ajtón történő átlépéskor a Turing-gép állapotát.
Ezen s_j állapotok sorozatát $\sigma(a, \omega)$ jelöli.

Szemléletesen úgy képzelhetjük a $\sigma(a, \omega)$ sorozatot, hogy az a ajtóban egy őr áll, amely a fej minden áthaladásakor feljegyzi annak állapotát.

Nyilvánvaló, hogy a $\sigma(a, \omega)$ sorozatból kiolvasható az ajtón való áthaladás iránya, hiszen a fej balról érkezik, ezért a páratlan sorszámú állapotok a balról-jobbra($\rightarrow$) történő átlépéskor, míg a páros sorszámú állapotok a jobbról-balra($\leftarrow$) történő átlépéskor kerülnek feljegyzésre.

Észrevétel

Jelölje $\omega \overset{a}{|}$ az ω input a ajtó előtti (tőle balra található) részét és $\overset{a}{|}\omega$ az ω input a ajtó utáni (tőle jobbra található) részét.

Természetesen a két rész kiadja a teljes ω inputot:
$\omega = \left(\omega \overset{a}{|}\right)\left(\overset{a}{|}\omega\right)$.

Észrevétel

Ha ismerjük az $\omega \overset{a}{|}$ inputrészletet és a $\sigma(a, \omega)$ állapotsorozatot, akkor meg tudjuk mondani, hogy a Turing-gép ,,hogyan működik", amikor a szem az a ajtó előtti mezőket pásztázza. Azt nem tudhatjuk, hogy a szem meddig volt a jobb oldalán, de amint átlépi az ajtót (és amíg a bal oldalon marad) képesek vagyunk T futását leírni. Természetesen ugyanez elmondható az $\overset{a}{|}\omega$ darabra is; ekkor az ajtó jobb oldalán ismerjük T lépéseit.

Az észrevétel következménye

Következmény

Legyenek $\omega, \omega' \in \Sigma^n$ tetszőleges inputok és a egy ajtó. Tegyük fel, hogy $\sigma(a, \omega) = \sigma(a, \omega')$ és a T Turing-gép futásának eredménye megegyezik a két inputon. Ekkor az $\widetilde{\omega} = \left(\omega \overset{a}{|}\right)\left(\overset{a}{|} \omega'\right)$ inputon is ugyanazt számolja ki T.

Valójában ennél többet is mondhatunk, hiszen az ajtón történő áthaladások számától függ, hogy melyik inputon (ω-n vagy ω'-n) fejeződik be a futás.

I_0

Definíció

Tegyük fel, hogy $3 \mid n$. Legyen

$$I_0 := \{\alpha 0^{\frac{n}{3}} \overleftarrow{\alpha} : \alpha \in \Sigma^{\frac{n}{3}}\} \subseteq \mathit{PALINDROM} \cap \Sigma^n.$$

$|I_0| = |\Sigma|^{\frac{n}{3}} = |\{0,1\}|^{\frac{n}{3}} = 2^{\frac{n}{3}}$.

Következmény

Legyenek ω, $\omega' \in I_0$ különböző szavak és a egy középső ajtó (azaz valamelyik a középső $n/3$ nullát elválasztó ajtó közül). Ekkor $\sigma(a,\omega) \neq \sigma(a,\omega')$.

Indirekt módon tegyük fel, hogy $\sigma(a,\omega) = \sigma(a,\omega')$. Ekkor az előző következmény alapján, ha mindkét inputon *ELFOGAD* állapottal áll le a Turing-gép, akkor az $\widetilde{\omega} = \left(\omega \overset{a}{|}\right)\left(\overset{a}{|} \omega'\right) = \alpha 0^{\frac{n}{3}} \overleftarrow{\alpha'}$ inputot is elfogadja. Ellentmondás.

Észrevétel

Észrevétel

Ha feltételezzük, hogy $|\Sigma| \geq 2$ és $|S| \geq 3$, akkor a t-nél rövidebb állapotsorozatok száma

$$1 + |S| + \cdots + |S|^{t-1} = \frac{|S|^t - 1}{|S| - 1} < |S|^t - 1 < |S|^t.$$

Következmény

Ha $|I_0| = |\Sigma|^{\frac{n}{3}} \geq |S|^t$, akkor $\exists \omega \in I_0$, hogy $\sigma(a, \omega)$ hossza legalább t, ahol a egy középső ajtó.

Indirekt módon tegyük fel, hogy nincs ilyen ω. Ekkor bármely $\omega \in I_0$ és a középső ajtó esetén $\sigma(a, \omega)$ hossza kisebb, mint t. Ez $|I_0| > |S|^t$ állapotsorozatot jelent. Azonban a t-nél rövidebb állapotsorozatok számára adott becslés alapján $|I_0| < |S|^t$. Ez ellentmondás, tehát az állításban szereplő ω input létezik.

Következmény

$|I_0| = |\Sigma|^{\frac{n}{3}} \geq |S|^t$ esetén $t \sim \beta_T \cdot n$.

Következmény

Legyen a egy tetszőleges középső ajtó. Ha $|I_0| \geq 2|S|^t$, akkor létezik legalább $|I_0|/2$ olyan szó I_0-ban, amelyre $|\sigma(a, \omega)| \geq t$ és mindegyik különböző állapotsorozatot ad, amikor kiszámítjuk. Ezek halmazát jelölje I_1. Tehát

$$I_1(a) = \{\omega \in I_0 : |\sigma(a, \omega)| \geq t\} \subseteq I_0.$$

Az előbbi megjegyzés alapján $t \sim \gamma_T \cdot n$, alkalmas γ_T konstansra.

Bizonyítás

Azok az időpontok nyilván nem relevánsak, amikor a fej nem mozdul, ezért az alábbi alsó becslés adható

$$\sum_{\omega \in I_0} TIME(\omega, T) \geq \sum_{\omega \in I_0} \sum_{a \text{ középső ajtó}} |\{t : t\text{-ben a fej átlép } a\text{-n}\}|$$

Az összegben megjelenő halmaz számossága $\sigma(a, \omega)$ definíciója alapján $|\sigma(a, \omega)|$,

$$= \sum_{\omega \in I_0} \sum_{a \text{ középső ajtó}} |\sigma(a, \omega)| = \sum_{a \text{ középső ajtó}} \sum_{\omega \in I_0} |\sigma(a, \omega)|$$

$$\geq \sum_{a \text{ középső ajtó}} \sum_{\omega \in I_1} t = \sum_{a \text{ középső ajtó}} |I_1| \cdot t \geq \sum_{a \text{ középső ajtó}} \frac{|I_0|}{2} \cdot t$$

$$= \frac{n}{3} \cdot \frac{|I_0|}{2} \cdot t = \frac{\gamma_T}{6} \cdot n^2 \cdot |I_0|$$

Bizonyítás: Vége

$|I_0|$-val való osztás után

$$\frac{1}{|I_0|} \sum_{\omega \in I_0} TIME(\omega, T) \geq \frac{\gamma_T}{6} \cdot n^2$$

adódik.

Tehát azt kaptuk, hogy az átlagos futásidő négyzetesen függ az input hosszától.

A γ_T konstans függ a Turing-géptől, értéke csökkenthető (az ára nagyobb állapothalmaz, nagyobb munkaszalag ábécé), de a négyzetes jelleg megmarad.

Szünet

Technikai problémák

Egy probléma formalizálása megkövetel egy véges ábécé-t.

Nincs olyan halmaz, amely az összes véges halmazt tartalmazná.

Ez a halmazelméleti tény problémát jelent, amelyen könnyű túljutnunk: egy megállapodást teszünk, amely alapján egy tetszőleges n-elemű Σ ábácé használata helyett mindig egy standard n elemű ábécével dolgozunk.

Ez a megállapodás nem jelent megszorítást.

Technikai problémák (folytatás)

Egy Turing-gép leírásához kell

(1) egy véges k természetes szám, amely „megmondja” hány munkaszalagunk van,

(2) egy véges S állapothalmaz,

(3) Γ véges munka-ábécé,

(4) egy δ átmeneti függvény, amely véges értelmezésitartománya/értékkészlete csak a korábbi választásainktól függ.

S és Γ választása megint korlátozható elemszámuk, egy természetes szám választására

Észrevétel

Megszámlálhatóan sok Turing-gép definiálható.

Következmény

Tétel

Adott nem-üres Σ esetén

(i) megszámlálhatóan végtelen sok $L \subset \Sigma^*$ eldönthető nyelv létezik,

(ii) megszámlálhatóan végtelen sok $L \subset \Sigma^*$ felsorolható nyelv létezik,

(iii) kontinuum számosságú nem felsorolható (így nem is eldönthető) nyelv létezik.

Következmény

Van nem felsorolható (és így nem kiszámítható nyelv).

Sokkal érdekesebb egy matematikailag értelmes, természetes nyelvre rámutatnunk és arról belátnunk, például hogy nem eldönthető. Az ilyen tételek már nem olyan egyszerűek.

Turing-gépek kódolása

Egy algoritmus futásához kell $\omega \in \Sigma^*$ input és egy Turing-gép. Azaz két megszámlálhatóan végtelen halmaz egy-egy eleme.

Egy kis jártasság után „érezhetjük", hogy minden megszámlálhatóan végtelen halmaz kódolható: természetes számok, egész számok, racionális számok algebrai számok, gráfok (véges gráfok izomorfiatípusai), gráfok racionális élsúlyokkal, racionális együtthatójú polinomok, racionális számokból álló mátrixok és így tovább.

Egy kis ugrás, hogy lássuk, hogy a Turing-gépek is kódolhatók. Állapodjunk meg egy speciális kódolásban amely egy fix ábécé-t, mondjuk $\Sigma_0 = \{0, 1\}$-et használ. Ezt a kódolást nem rögzítjük le, a későbbi diszkusszióban nem térünk ki olyan technikai részletekre, ahol ez lényeges lenne.

Az érdeklődő hallgató választhat egy számára megfelelő megállapodást.

Turing tétele

Jelölés

Legyen $\lceil T \rceil$ a T Turing-gép kódja. Legyen $\lceil \omega \rceil$ egy $\omega \in \Sigma^n$ kódja.

$\lceil T \rceil$ és $\lceil \omega \rceil$ a megállapodással a fejünkben minden információt tartalmaz, ami ahhoz szükséges, hogy T futását szimuláljuk az ω inputon. Ez Turing-géppel is megvalósítható:

Tétel (Alan Turing)

Feltesszük, hogy megállapodtunk egy fent vázolt kódolási rendszerben. Ekkor létezik egy U Turing-gép, amely $\lceil T \rceil, \lceil \omega \rceil$ inputon szimulálja T-t ω-n. Speciálisan futása ELFOGAD állapotba jut, ha T ezt teszi ω-n, ELVET állapotba jut, ha T ezt teszi ω-n, végül végtelen, ha T is végtelen futásba kerül az ω inputon.

Univerzális Turing-gép

A tételbeli Turing-gépet (a megállapodások után, azokat közzétéve) *univerzális Turing-gépnek* nevezzük.

Ez a fogalom játéknak tűnhet, azonban rendkívül fontos szerepet játszik az algoritmusok, bonyolultságelmélet, számítástechnika történetében.

Az, hogy az algoritmusok 0-1 sorozattal írhatók le, amit egy eszköz memóriájában tároljunk, amit az eszköz értelmez utasítások sorozataként az a modern számítógép fogalmának megszületése.

Ezt Neuman István nevéhez fűzik, de Turing munkássága nélkül nem tehette volna meg ennek kifejtését.

Talán hasznos adnunk egy szótárat, ami a bonyolutságelmélet alapfogalmait köti a mindennapos szóhasználatunkkal

A szótár

Bonyolultságelmélet	Hétköznapi matematika
Turing-gép	Algoritmus
Megállapodás, hogyan kódoljunk egy Turing-gépet	Egy programozási nyelv
$\lceil T \rceil$	Egy program (konkrét algoritmus kódja)
Univerzális Turing-gép	Egy programozási nyelvben leírt tetszőleges algoritmus interpretálására képes számítógép

Megállási probléma

A fenti előkészületek után megadunk egy eldönthetetlen nyelvet, ami a Turing-gépek definíciójával kapcsolatos. Be is bizonyítjuk eldönthetetlenségét. A példa Turing nevéhez fűződik, az első kiszámíthatatlansági eredmény.

Definíció: Megállási probléma

A megállási problémában adott egy T Turing-gép és egy ω input. El kell döntenünk, hogy T leáll-e ω-n.

$$\text{MEGÁLLÁS} = \{\lceil T, \omega \rceil : T \text{ leáll } \omega\text{-n, azaz STOP vagyis ELVET/ELFOGAD állapotba kerül}\}.$$

Turing tétele

Turing-tétel

(i) MEGÁLLÁS $\in \mathcal{S}$,

(ii) MEGÁLLÁS $\notin \mathcal{D}$.

Azaz MEGÁLLÁS felsorolható, de nem eldönthető.

A tétel első részét az univerzális Turing-gép bizonyítja: A szimuláló gép leállását úgy kell módosítani, hogy ne STOP állapotba jusson, hanem a felsorolhatóságot bizonyító Turing-gép ELFOGAD állapotába.

Így az elfogadandó inputokra a gép elfogad, míg a nem elfogadandó inputokat úgy jelzi ahogy kell: a szimulálással, ami egy végtelen futás.

Bizonyítás

A második állítás bizonyítása indirekten történik, azaz tegyük fel, hogy létezik I Turing-gép, amely eldönti a MEGÁLLÁS nyelvet. Továbbiakban az indirekt feltevés I gépére alapítva egy kissé módosított gépet írunk le.

A következő (könnyen, de technikai módon megoldható) feltevéssel élünk: A Turing-gépeket azonsítjuk a természetes számokkal. Azaz minden i természetes szám egy T Turing-gép kódja ($i = \lceil T \rceil$). Továbbá i-ből dekódolható T.

Hasonlóan azonosítjuk Σ^*-ot és $\mathbb{N}$-et. $j = \lceil \omega \rceil$ esetén j-ből dekódolható ω.

Bizonyítás (folytatás)

Képzeljünk el egy $\mathbb{N} \times \mathbb{N}$ típusú táblázatot, amely (i, j) pozíciójában $(i = \lceil T \rceil, j = \lceil \omega \rceil)$ ∞ áll, ha T az ω-n végtelen ciklusba kerül és 0 különben (T az ω-n leáll). Indirekt feltevésünk az, hogy van olyan I Turing-gép, amely „kiszámolja" ezt a táblázatot.

Az ellentmondást Cantor átlós módszere adja.

Az E Turing-gép

(Átló): Beolvas egy i inputot és kiszámolja a fenti táblázat i indexű ($\lceil T_i \rceil = i$, $\lceil \omega_i \rceil = i$) átlós elemét.

(Fordítás1): Ha az i indexű átlós elem ∞ (azaz T_i nem áll le ω_i-n), akkor E gépünk STOP állapotba kerül.

(Fordítás2): Ha ez 0 (azaz T_i leáll ω_i-n), akkor E gépünk jobbra-balra „lépeget", végtelen ciklusba kerül.

E éppen a kiolvasott információval ellentétes viselkedést végez.

Bizonyítás (folytatás)

E egy TG, azaz valamilyen k-ra $\lceil E \rceil = k$.

Mit csinál E a k inputot olvasva?

A definíció alapján „kibontja k"-t mint Turing-gép és ekkor magát/E-t találja.

E hogyan működik ω-n ($\lceil \omega \rceil = k$)? Akár leáll, akár végtelenségig fut E definíciója ellentmondáshoz vezet.

A bizonyítás lényege hasonlít Cantor bizonyítására, hogy $[0, 1]$ nem felsorolható/megszámlálhatóan végtelen halmaz (átlós módszer). Csak most valós számok helyett gépek, illetve tizedesvessző utáni pozíciók helyett inputok kódjai szerepelnek.

Szünet

$\mathcal{D}$-n kívül

Említettük, hogy a bonyolultságelmélet témája a $\mathcal{D}$-beli nyelvek vizsgálata, összehasonlításuk, bonyolultság szerint struktúrálásuk. A $\mathcal{D}$-n kívüli nyelvek is igen aktívan vizsgáltak. Kutatásuk módszerei és motivációja inkább a matematikai logikához köthető.

Egy új probléma esetén az első kérdés (döntési problémák esetén), hogy $\mathcal{D}$-hez tartozik-e. Nagyon sok matematikailag fontos, központi kérdés esetén kiderült, hogy nem $\mathcal{D}$-beli kérdésről van szó.

Egy ilyen matematikai tétel jelentése az, hogy amíg a Church-tézis jól leírja a kiszámíthatóság fogalmát, addíg tudjuk, hogy a probléma általánosságban számítógéppel NEM kezelhető.

Természetesen speciális inputokra, különböző feltételek mellett elképzelhető a kiszámíthatóság. Ilyen problémáknál a matematikai kutatásoknak ebbe az irányba kell tartaniuk.

Hilbert X. Problémája

Hilbert X. Problémája

Legyen

$$DIOPHANTOSZ = \{\lceil p(x) \rceil : p \in \mathbb{Z}[x_1, x_2, \ldots, x_n], \\ p\text{-nek van egész gyöke}\}.$$

Hilbert problémájának modern értelmezése, hogy *DIOPHANTOSZ* nyelv $\mathcal{D}$-hez tartozik-e. A probléma kitűzésének idejében $\mathcal{D}$ fogalma még nem született meg. A klasszikus nyelven a probléma az, hogy van-e olyan algoritmus, ami egy adott egész együtthatós polinomról eldönti, hogy van-e egész gyöke.

Hilbert X. Problémája: A tétel

Két lehetőség volt. Vagy valaki ad egy algoritmust, ami megoldja Hilbert problémáját (azaz DIOPHANTOSZ$\in \mathcal{D}$), a metematikusok közössége pedig megérti, ellenőrzi és elfogadja az algoritmust.

A másik lehetőség: nincs ilyen algoritmus. Ebben az esetben ezt bizonyítani kell. Ez nem megy $\mathcal{D}$ definíciója nélkül. Kiderült, hogy a második lehetőség az igazság.

Matijaszevics (1970)

$$DIOPHANTOSZ \notin \mathcal{D}.$$

Hilbert X. Problémája: A történet

Hilbert X. problémájának megoldásának története:

1900 Hilbert előadja a problémát,

1935 Church megfogalmazza a Church-tézist,

1936 Turing bevezeti a Turing-gép fogalmát, a Church-tézist széleskörben elfogadják,

1950- a diophantikus halmazok bevezetése és vizsgálata Davies és Robinson vezetésével,

1970 Matijaszevics megteszi az utolsó (legnehezebb) lépéseket, bebizonyítja, hogy DIOPHANTOSZ nem tartozik $\mathcal{D}$-hez.

Természetesen DIOPHANTOSZ$\in \mathcal{S}$ (miért?).

Egy változó illetve lineáris eset könnyen megoldható. A kvadratikus kétváltozós eset is megoldható, de már komoly számelméleti vizsgálatok szükségesek.

Csoportok szóproblémája

SZÓPROBLÉMA inputja tartalmaz egy G csoportot. G-re multiplikatív írásmódot használva hívatkozunk. Mielőtt leírnánk a teljes problémát tisztáznunk kell, hogyan kódolhatunk csoportokat?

Egy lehetséges megoldást ad a kombinatorikus csoportelmélet. Legyen G egy csoport egy B generátorhalmazzal. Ekkor B elemeiből kifejezéseket épithetünk fel, amik a csoport egy-egy elemét írják le. Ha $B = \{a, b, c\}$, akkor $abbaca^{-1}ba^{-1}$ egy ilyen kifejezés. 1, az előző betűkészletből felírt üres szorzat is egy kifejezés, ami a csoport egységelemét írja le. Tehát a kifejezéseink, szakzsargonnal *szavaink*, B elemeiből és B elemeinek inverzéből szorzásokkal felépített kifejezések.

Persze különböző szavak írhatják le ugyazt az elemet. A csoportszámtan garantálja, hogy $aa^{-1}b$ és b ugyanazt az elemet írja le.

Csoportok szóproblémája: Szabadon generált csoportok

Egy szó elemi egyszerűsítése az xx^{-1}, illetve $x^{-1}x$ egymásutáni két karakter kihúzása.

Ha egy $w_1, w_2, w_3, \ldots, w_n$ szósorozatban bármely két egymásutáni szó közül egyik a másik elemi egyszerűsítése, akkor a sorozat bármely két eleme ugyanazt a csoportelemet írja le. Azt mondjuk w_1 és w_n ekvivalens.

Ez egy ekvivalenciareláció a B-ből felírható csoportkifejezések halmazán. Az ekvivalenciaosztályok között könnyű szorzást, inverzet, egységosztályt definiálni. Így egy csoporthoz jutunk. Ez a B generátorhalmazhoz tartozó „legbővebb" generált csoport. A neve a B által szabadon generált csoport.

A B által szabadon generált csoport esetén könnyű tervezni egy algoritmust, amely két adott szóról eldönti, hogy ugyanazt a csoportbeli elemet írják-e le.

Csoportok szóproblémája: Végesen prezentált csoportok

Jóval általánosabb csoportok is leírhatók a fenti módszer általánosításával.

Adjunk meg elemi egyszerűsítésekkel (és persze elemi bonyolításokkal) nem levezethető szóegyenlőségeket. Ha ilyen összefüggések egy halmazát adjuk meg, akkor ehhez is tartozik egy csoport: az elemi egyszerűsítés/elemi bonyolítás fogalmát ki kell terjeszteni az egyenlőség egyik oldalán szereplő kifejezés átírásával a másik oldalon szereplő kifejezésre.

Így ha adott egy B halmaz és T egyenlőségek egy halmaza (ezek bal és jobb oldalán egy-egy szó szerepel), akkor egy $G = \langle B; T \rangle$ csoportot írtunk le.

Amennyiben B és T véges az így leírt csoportok a végesen prezentált csoportok.

Csoportok szóproblémája: Példák végesen prezentált csoportokra

Példa

$\langle a, b; ab = ba \rangle$ egy csoport.

Könnyen ellenőrizhető, hogy ez $(\mathbb{Z}, +) \times (\mathbb{Z}, +)$.

Példa

$\langle a, b; a^n = b^2 = abab = 1 \rangle$ egy csoport.

Könnyen ellenőrizhető, hogy ez D_n.

Csoportok szóproblémája: A tétel

Ezekután a problémánk: Legyen adva egy B véges generátorhalmaz, egy véges T összefüggés halmaz (igy adva van egy $G = G(B; T)$ végesen prezentált csoport). Adott még két B-re épített szó. Döntsük el, hogy azonos csoportbeli elemet írnak-e le.

Definíció

$$\text{SZÓPROBLÉMA} = \{\lceil B, T; w_1 = w_2 \rceil : \text{ a } \langle B; T \rangle \text{ csoportban a } w_1 \text{ és } w_2 \text{ csoportelemek megegyeznek}\}$$

Tétel (Novikov (1955), Boone (1958))

A probléma eldönthetetlen,

$$\text{SZÓPROBLÉMA} \notin \mathcal{D}.$$

Homeomorfizmus

HOMEOMORF inputja két topológikus tér. Azt kell eldöntenünk, hogy homeomorfak-e.

Ismét a lényeges kérdés: Hogyan kódolunk topológikus tereket? A legegyszerűbb megoldás topologikus terek egy tág osztályának leírására a rekurzió: Egyszerű, jól ismertnek vett topológikus terekből egyszerű operációkkal „felépítünk" további, bonyolultabbakat.

Talán a legkombinatorikusabb lehetőség, ha szimplexekből indulunk ki. Szimplexek a pontok, szakaszok, háromszögek, tetraéderek. Ezek pontosan a legfeljebb három-dimenziós szimplexek. Minden d természetes szám esetén definiálható egy d-dimenziós szimplex, például a $\mathbb{R}^d$ origója és e_i standard báziselemeinek konvex burka.

A felépítéshez használható operáció lehet az azonos dimenziós lapok menti ragasztás.

Homeomorfizmus (folytatás)

A Könnyű igazolni, hogy csak a kiinduló szimplexek dimenziója és a ragasztásnál használt lapok ismerete elég a leírt topológikus tér homomorfiatípusának ismeretéhez.

Ennek leírásához a szimplexeket és lapjaikat azonosítjuk csúcsaik halmazával. A szimpliciális komplexus egy halmazrendszer lesz egy véges V halmaz felett. A szimpliciális komplexus egyetlen tulajdonsággal jellemezhető: minden hozzátartozó halmaz összes részhalmaza is hozzátartozik (egy szimplex csúcsainak tetszőleges részhalmaza egy jól meghatározott lapja — ami szintén egy szimplex — csúcshalmaza).

Homeomorfizmus: A tétel

Tétel

A SZIMPLICIÁLIS-KOMPLEXUSOK-HOMEOMORFIZMUSA probléma nem eldönthető. Azaz

$$\text{HOMEOMORF} \notin \mathcal{D}.$$

Post dominó problémája

A POST problémában adott Σ véges ábécé. Az input egy dominó készlet: Véges sok dominótípus, ahol egy típus egy alsó és egy felső minta, ami egy-egy Σ^*-beli szó. Minden típusból végtelen sok dominónk áll rendelkezésünkre. Azt kell eldönteni, hogy ki tudunk-e rakni dominóinkból egy (véges) sort úgy, hogy az alsó és felső minták összeolvasva (konkatenálva) ugyanaz a szót adják.

Leírásunk elemi volt. A probléma ehelyett a félcsoportok nyelvén is megfogalmazható. Az irodalomban legtöbbször félcsoportokra vonatkozó problémaként ismertetik ezt a nyelvet.

A probléma nem eldönthető.

Tétel (Post)

$$\text{POST} \notin \mathcal{D}.$$

Csempézési problémák

Egy négyzetet két átlójával osszunk négy negyednégyzetre. Mindegyik negyedez színezzünk ki egy színnel. Az így kapott négyzetet nevezzük csempetípusnak.

Osszuk fel a síkot egymással páhuzamos vízszintes és függőleges egyenesekkel csempényi nagyságú négyzetekre.

Csempézési probléma

Adott véges sok csempetípus. Mindegyikből végtelen sok csempénk van.

Kicsempézhető-e a sík (a fenti felosztás négyzeteibe rakható-e csempe) úgy, hogy az éllel találkozó csempéknél a megfelelő két csempenegyed színe ugyanaz legyen?

Csempézési problémák: Wang-csempék

Wang Csempézési probléma

Adott véges sok csempetípus. Mindegyikből egy-egy egymás mellé helyezve a síkon. Mindegyik típusból végtelen sok csempénk van.

Kicsempézhető-e a sík (a fenti felosztás négyzeteibe rakható-e csempe) úgy, hogy az egyes csempék lerakása a lerakott típusokbók eltolással kapható, továbbá az éllel találkozó csempéknél a megfelelő két csempenegyed színe ugyanaz legyen?

Csempézési problémák: A tétel

Tétel

$$CSEMPE \notin \mathcal{D}.$$

Tétel

$$WANG\text{-}CSEMPE \notin \mathcal{D}.$$

Vége van!

Köszönöm a figyelmet!