Turing-gépek

Kiegészítő anyag az Algoritmuselmélet tárgyhoz

(a Rónyai–Ivanyos–Szabó: Algoritmusok könyv mellé)

Friedl Katalin
BME SZIT
friedl@cs.bme.hu

2017. augusztus 16.

A veremautomatáknál az, hogy a memória egy verem, komoly megkötésnek bizonyul. Most egy olyan számítási modell következik, ahol ezt a megkötést enyhítjük. Bár ez is egy elméleti modell, de mint látni fogjuk, bizonyos szempontból a lehető legáltalánosabb. A Turing-gépeknek ugyan sokféle, egymással egyenértékű változata van, mi itt csak egy determinisztikust és egy nemdeterminisztikust tárgyalunk. A definíció hasonló az eddigiekhez, csak itt a korábbi vermet szalag helyettesíti, amin ugyan ugrani nem lehet, csak egyesével lépegethetünk, de mozoghatunk előre és hátra is. Kezdjük a determinisztikussal:

Determinisztikus Turing-gép

1. Definíció. *Legyen $k \geq 1$ egy egész szám. Egy* k-szalagos Turing-gépet *egy* $M = (Q, \Sigma, \Gamma, q_0, *, F, \delta)$ *hetes ír le, ahol:*

- Q *egy véges, nem üres halmaz, a gép* állapot*ainak halmaza*
- Σ *egy véges, nem üres halmaz, a bemeneti* ábécé
- Γ *egy véges, nem üres halmaz, a* szalagábécé, $\Sigma \subset \Gamma$
- $q_0 \in Q$ *a* kezdő állapot
- $* \in \Gamma \setminus \Sigma$, *a szalagon az* üres jel
- $F \subseteq Q$ *az* elfogadó állapotok *halmaza*
- δ *az* átmeneti függvény,

$$\delta : (q, a_1, a_2, \cdots, a_k) \to (q', b_1, b_2, \cdots, b_k, D_1, D_2, \cdots, D_k),$$

ahol $q, q' \in Q$, $a_i, b_i \in \Gamma$ *és* $D_i \in \{B, J, H\}$ *(azaz* ***B****alra,* ***J****obbra vagy* ***H****elyben).*

Úgy kell elképzelni, hogy minden szalag egymás utáni mezőkből áll, amelyek mindegyike a Γ ábécé egy-egy elemét képes tárolni. A szalagnak van egy eleje, ez az első mező, azután jön a második mező, stb. A gép kezdetben a q_0 állapotban van. Az első szalag első néhány mezőjén a bemeneti szó található, amiben csak Σ-beli karakterek szerepelhetnek. Az első szalag többi része, és ha $k > 1$, akkor a többi szalag mindenhol a $*$ (üres hely) szimbólummal van feltöltve. Minden szalaghoz tartozik egy író/olvasó fej, és ezek a szalag első mezőjén állnak.

Ha egy adott helyzetben az író/olvasó fej alatt levő karakter az első szalagon a_1, a másodikon a_2, az i-ediken a_i, és a gép a q állapotban van, akkor egy lépésben, az átmeneti függvény

$$\delta(q, a_1, a_2, \ldots, a_k) = (q', b_1, b_2, \cdots, b_k, D_1, D_2, \cdots, D_k)$$

értéke szerint a gép a q' állapotba kerül, az i-edik szalagon az a_i karaktert átírja b_i-re és az i-edik szalagon a D_i alapján mozdul el a fej egyet Balra, Jobbra vagy Helyben marad.

A Turing-gép egy számítás során a kezdő helyzetből indulva az átmeneti függvénynek megfelelő lépések sorozatát hajtja végre. A gép működése akkor ér véget, ha nem tud lépni, azaz elakad a számítás, mert az adott helyzetre az átmeneti függvény nincs értelmezve. A gép akkor fogadja el a bemeneti szót, ha ez az elakadás F-beli állapotban történik.

Arról a Turing-gép megadásakor gondoskodnunk kell, hogy amikor egy szalag elején van a fej, akkor onnan ne lépjen balra (nem szabad „leesni" a szalagról). Ha mégis ez történne, akkor a számítás hibaüzenettel leáll.

Fontos megjegyezni, hogy semmi nem garantálja, hogy egy adott bemenettel a Turing-gép valaha is le fog állni. A következő lehetőségek vannak:

- A gép előbb-utóbb leáll elfogadó állapotban, ilyenkor a szót a gép *elfogadja*.
- A gép előbb-utóbb leáll nem elfogadó állapotban, ilyenkor *nem fogadja el* a szót.
- A gép az adott bemenet hatására soha sem áll meg. Természetesen ez sem elfogadó számítás.
- A gép hibaüzenettel leáll (mert balra lelépne egy szalagról). Ilyenkor nem fogadja el a bemenetet (függetlenül attól, melyik állapotban akadt így el).

Tehát csak az első eset elfogadó, amikor egy elfogadó állapotban akad el (és ez nem a szalagról "leesés" miatt történik).

1. Megjegyzés. *A Turing-gép definíciója a determinisztikus, hiányos automatákra emlékeztet. Fontos de formai megkötés, hogy most az elfogadás feltétele más: a véges automatáknál és veremautomatáknál megköveteltük, hogy elfogadó számításnál a bemeneti szót végig kell olvasni, nem szabad, hogy közben a számítás elakadjon. Most viszont a végigolvasás nem szükséges, és elfogadni csak elakadás esetén lehetséges.*

Lehetne az elfogadási feltételt a korábbiakhoz hasonlóan megadni, ezzel egy ekvivalens modellt kapnánk, de a jelen forma az általánosan elterjedt és egyszerűbben is használható.

2. Megjegyzés. *A Turing-gépben a korábbiakhoz képest a több szalag írásán olvasásán túl az is újdonság, hogy a fej a bemeneten mindkét irányban mozoghat. Megmutatható az oda-vissza mozgás a véges automaták esetében is megengedhető, ezzel nem bővül a felismerhető nyelvek osztálya, így a DVA és a Turing-gép közötti lényegi különbség, hogy míg a DVA csak olvas, a Turing-gép írni is tud.*

Megmutatható, hogy az eltérések ellenére a Turing-gép általánosítása mind a véges automatának, mind a veremautomatának: mindkét automatát könnyen lehet szimulálni Turing-géppel, az állapotokat a Turing-gép állapotaiban őrizzük. A veremautomata esetén a verem tartalmát egy szalagon tároljuk, a szalagot a veremhez hasonlóan kezeljük. A véges automatánál pedig lényegében nem írunk a szalagokra.

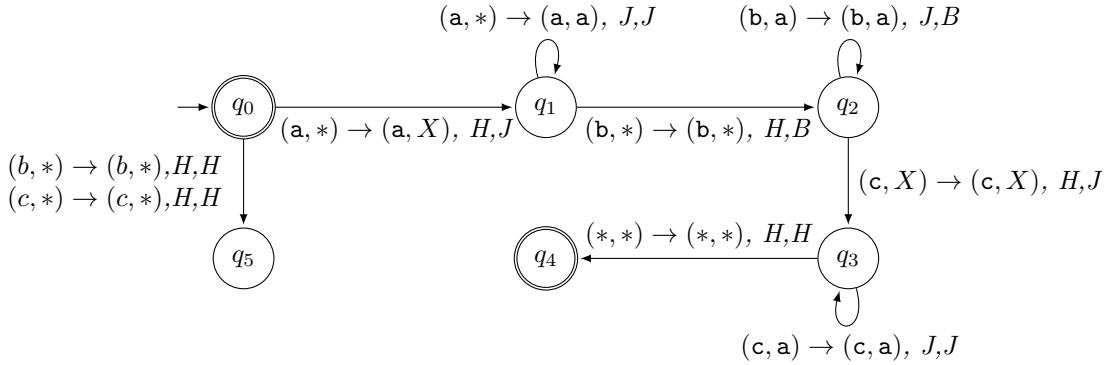
2. Definíció. *Az M Turing-gép által elfogadott nyelv:*

$$L(M) = \{w \in \Sigma : M \text{ elfogadja a } w \text{ szót}\}$$

A Turing-gépeket is lehet gráffal ábrázolni, ilyenkor az átmeneteket jelző nyilakon szerepel, hogy milyen olvasott karakter hatására milyen karaktert ír a gép, illetve milyen irányba lép tovább. Ez a forma azonban, különösen több szalag esetén már közel sem olyan áttekinthető, mint a DVA-nál volt.

Annak illusztrálására, hogy a Turing-gép többet tud, mint a veremautomata (és a DVA), megmutatjuk, hogy a nem CF $\{\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n\}$ nyelvhez van Turing-gép.

1. Példa. *Az alábbi 2-szalagos Turing-gép az $\{\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n : n \geq 0\}$ nyelvet fogadja el. Az automata ötlete az, hogy a bemeneten található* a *betűket átmásoljuk a 2. szalagra, a 2. szalagon visszafelé menve hasonlíthatjuk a* b *betűk számát az* a *betűkhöz, majd előrefelé haladva a* c *betűkhöz.*



Kicsit pontosabban:

- *q_0 állapotban ha az üres szó a bemenet, akkor azt elfogadja a gép. Ha* b *vagy* c *betűvel kezdődik, akkor átlép a q_5-be, ahol a számítás véget ér (és nem fogadja el a szót). Amennyiben a bemenet* a *betűvel kezdődik, akkor a második szalag elejére teszünk egy X-et, hogy visszafelé jőve, tudjuk majd, hol van az első mező. Az 1. szalagon nem változik semmi.*

- *a q_1 a másolás, amíg* `a` *betű jön az 1.szalagon, a 2-ra is írunk egy* `a` *betűt. Az első* `b`*-nél átmegyünk a q_2 állapotba. (Ha pl.* `c` *betű jön, akkor a számítás elakad, és mivel nem elfogadó állapotban vagyunk, nem fogadjuk el a szót.)*

- *a q_2 állapotban az első szalagon továbbra is jobbra, de a másodikon balra haladunk amíg* `b` *betű van az első és* `a` *betű a második szalagon. Akkor tudunk átlépni a q_3 állapotba, ha egyszerre érünk az első* `c` *betűhöz, illetve a 2. szalag elejét jelző X szimbólumhoz, azaz ha az* `a` *és* `b` *betűk száma megegyezik.*

- *a q_3 a* `c` *betűk és a 2. szalagon levő* `a` *betűk számának összehasonlítására szolgál, a 2. szalagon megint előrefelé megyünk.*

- *az elfogadó q_4 állapotba akkor fogunk átlépni, ha a mindkét szalagon egyszerre érjük el az első üres jelet.*

A diagonális és a megállási nyelv

Turing-géppel se lehet minden nyelvet elfogadni. Ennek megmutatásához szükségünk lesz arra, hogy észrevegyük, egy Turing-gépet le lehet írni véges sok karakterrel (lényegében az átmeneti függvényt kell megadni). Egy ilyen leírás felírható 0/1 sorozattal is.Tehát egy Turing-gép leírása (de akár egy véges automata vagy egy veremautomata leírására is) tekinthető a $\{\mathtt{0},\mathtt{1}\}^*$ egy elemének, ezt hívjuk az adott Turing-gép kódjának is. Természetesen nem minden szó fog Turing-gépet leírni. Ha $w \in \{\mathtt{0},\mathtt{1}\}^*$ egy Turing-gép kódja, akkor jelölje a hozzá tartozó Turing-gépet M_w.

A következő nyelvnél egy 0/1 sorozat kétféle szerepben jelenik meg: egyrészt, mint egy Turing-gép leírása, másrészt mint egy lehetséges bemenete a Turing-gépnek.

3. Definíció. *Az L_d* diagonális nyelv *az olyan $w \in \{\mathtt{0},\mathtt{1}\}^*$ szavakból áll, amelyek Turing-gépek kódjai és az M_w Turing-gép nem fogadja el a w szót, azaz*

$$L_d = \{w \in \{\mathtt{0},\mathtt{1}\}^* : \exists M_w \text{ és } w \notin L(M_w)\}.$$

1. Tétel. *Nincs olyan M Turing-gép, amelyik az L_d diagonális nyelvet fogadja el.*

Bizonyítás: Indirekt módon bizonyítunk. Tegyük fel, hogy van ilyen M Turing-gép, legyen ennek a kódja x. Kérdés, hogy x benne van-e a diagonális nyelvben.

Próbáljuk ki, mit jelent, ha $x \in L_d$. Ekkor a diagonális nyelv definíciója szerint $x \notin L(M_x)$. Másrészt az x-et úgy választottuk, hogy $M_x = M$, és M választása miatt $L(M_x) = L(M) = L_d$. Ezzel ellentmondásra jutottunk, hiszen a feltevés miatt $x \in L_d$ és x választása miatt $x \notin L(M_x) = L_d$.

Próbáljuk ki a másik esetet is, azaz amikor $x \notin L_d$. Az előzőhöz hasonlóan ebből meg az következik, hogy $x \in L(M_x)$, ami megint csak ellentmondás. Mivel

több lehetőség nincs, így az eredeti feltevésünk, hogy M a diagonális nyelvet fogadja el a hibás. Tehát valóban nincs a diagonális nyelvhez Turing-gép. □

Egy másik ,,problémás" kategória az, amikor ugyan van Turing-gép a nyelvhez, de olyan nincs, ami minden bemeneten meg is áll.

4. Definíció. *A* megállási nyelv *azokból a (Turing-gép, bemenet) párokból áll, melyekre az adott Turing-gép megáll az adott bemeneten:*

$$L_h = \{w\#x : \exists M_w \text{ és megáll az } x \text{ bemeneten } \}$$

2. Tétel. *Az L_h nyelvhez van Turing-gép, de nincs olyan M Turing-gép, ami minden bemeneten véges sok lépés után megáll és az L_h nyelvet fogadja el.*

Bizonyítás vázlat: Azt itt nem bizonyítjuk, hogy van Turing-gép, csak azt vázoljuk, miért nem lehet olyan, ami minden bemeneten megáll. Megmutatjuk, hogy ha lenne ilyen M Turing-gép, akkor olyan M' is lenne, ami a diagonális nyelvet ismeri fel. Ez utóbbiról pedig már tudjuk, hogy nem létezhet. Tehát legyen M olyan, hogy minden bemeneten megáll és $L(M) = L_h$. Ez lényegében azt jelenti, hogy tesztelni tudjuk, hogy például az M_w gép megállna-e ha a w szót kapja bemenetként.

Amennyiben tudjuk, hogy megállna, akkor akár szimulálhatjuk is (higgyük itt el, ez megoldható a Turing-gép leírása alapján) és véges sok lépés után megkapjuk, hogy elfogadja-e ezt a szót. Ha igen, azaz $w \in L(M_w)$, akkor az M' gép utasítsa el a bemenetet, ha pedig $w \notin L(M_w)$, akkor fogadja el.

Ezzel szemben, ha az derül ki, hogy az M_w gép nem állna meg a w szón, akkor nincs szükség arra, hogy szimuláljuk a működését, hiszen, ha nem áll meg, akkor nem is fogadhatja el. Azaz ilyenkor az M' gép álljon meg elfogadó állapotban.

Így tehát egy olyan M' Turing-gépet adhatunk meg, ami minden bemeneten véges sok lépésben megáll, és éppen a diagonális nyelvet fogadja el, ami viszont ellentmond az előző tételnek. □

A Turing-gépekre gondolhatunk, mint valamilyen nyelven írt programokra, a Turing-gépnek adott szóra pedig, mint a program bemenetére. Ezzel az analógiával élve az előző tétel azt mutatja, hogy nem lehet olyan ellenőrző programot írni, ami véges lépésben bármilyen programról és ahhoz tartozó bemenetről eldönti, hogy a program megáll-e azon a bemeneten.

Bár első látásra úgy tűnhet, az előbb leírt analógia nem túlzó. Általában a Turing-gépek erejét és korlátait mutatja az alábbi

Church–Turing-tézis

1. Egy L nyelvhez akkor és csak akkor van ezt elfogadó Turing-gép, ha van olyan (nem feltétlenül véges) eljárás, ami pontosan L szavait fogadja el.

2. Egy L nyelvhez akkor és csak akkor van olyan, az L nyelvet elfogadó Turing-gép, ami minden bemeneten véges sok lépés után megáll, ha van olyan (mindig véges lépésből álló) algoritmus, ami tetszőleges x bemenet esetén eldönti, hogy x benne van-e az L nyelvben.

A tézisben szereplő eljárás, algoritmus fogalmakra nincsen definíciónk, ezért a fenti tézis tételnek nem tekinthető. A Church–Turing-tézis azt a tapasztalati tényt rögzíti, hogy eddig senki sem tudott olyan számítási modellt, olyan algoritmus fogalmat kitalálni, amivel több nyelvet lehetett volna felismerni, mint Turing-gépek segítségével.

Röviden tehát azt mondja ki a Church–Turing-tézis, hogy számítási modellek terén a Turing-gép egy lehetséges legjobb, legerősebb eszköz, amivel rendelkezünk. Ha a jövőben változik, mit tekintünk algoritmusnak, akkor ezt a tézist újra kell majd gondolni.

Nemdeterminisztikus Turing-gép

A nemdeterminisztikusság már ismert fogalom, itt is hasonlóan működik. Nemdeterminisztikus Turing-gép esetén az átmeneti függvény értéke egy halmaz, a gép akkor fogad el egy szót, ha van olyan számítási út, amelyik elfogadó állapotban ér véget.

Azt azért érdemes kiemelni, hogy egy bementhez tartozó számítási fának itt lehetnek véges és végtelen hosszú útjai is.

A véges automatákhoz hasonlóan itt is meg lehet szabadulni a nemdeterminizmustól,

3. Tétel. *Minden nemdeterminisztikus Turing-gép szimulálható determinisztikus Turing-géppel.*

Bizonyítás vázlat: Legyen M a nemdeterminisztikus Turing-gép, amihez egy M' determinisztikusat akarunk megadni. Az ötlet az, hogy M' egy tetszőleges x bemenetre az M számítási fáján egy szélességi bejárást végez (pontosabban föntről lefelé haladva generálja ezt a számítási fát). Ha talál egy elfogadó levelet (azaz, ahol M elfogadó állapotban elakadna), akkor M' megáll elfogadó állapotban. Ha a bejárás véget ér és nem talált ilyen levelet, akkor megáll egy nem elfogadó állapotban. Ha meg a fa végtelen nagy és nincs benne elfogadó levél, akkor M' nem fog megállni (ami definíció szerint azt jelenti, hogy nem fogadja el a bemenetet). □

3. Megjegyzés. *Vegyük észre, hogy a mélységi bejárás erre a célra nem használható, mert az nem tér vissza, ha a számítási fában van egy végtelen hosszú út.*

Polinomiális idő

A továbbiakban olyan Turing-gépekkel foglalkozunk, amelyek minden bemeneten véges sok lépés után megállnak. Ilyenkor az a fontos kérdés, hogy hány lépésben állnak meg. Ezt a lépésszámot érdemes a bemenet hosszához viszonyítani, hiszen nagyobb feladaton jogos a több lépés.

5. Definíció. *Egy M determinisztikus Turing-gépre azt mondjuk, hogy $f(n)$* időkorlátos, *ha minden x bemenetre teljesül, hogy az x bemeneten M legfeljebb $f(|x|)$ lépést tesz.*

Azaz $f(n)$ egy felső becslés a Turing-gép futási idejére az n hosszú szavakon.

6. Definíció. *Egy M nemdeterminisztikus Turing-gépre azt mondjuk, hogy $f(n)$* időkorlátos*, ha minden x bemenetre teljesül, hogy az x bemeneten M legfeljebb $f(|x|)$ lépést tesz.*

Azaz $f(n)$ egy felső becslés a Turing-gép futási idejére az n hosszú szavakon, bármelyik számítási ágat is nézzük, tehát felső becslés a számítási fa magasságára.

7. Definíció. *Az M Turing-gép* polinom időkorlátos*, ha $f(n)$ időkorlátos valamilyen $f(n)$ polinomra (azaz valamilyen c konstansra a lépésszám $O(n^c)$).*

A nyelveket osztályozhatjuk az alapján, hogy milyen gyors Turing-gép található rájuk.

A két, talán legfontosabb osztály a P és az NP.

8. Definíció. *A* P *azoknak a nyelveknek az osztálya, amelyekhez van polinom időkorlátos determinisztikus Turing-gép. Az* NP *pedig azoké, amelyekhez van polinom időkorlátos nemdeterminisztikus Turing-gép.*

2. Példa. *Például a* P *osztályba tartozik az $\{\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n : n \geq 0\}$ nyelv, hiszen a korábban megadott Turing-gép nem csak hogy polinom, de a bemenet hosszában lineáris lépést használ.*

A fenti nyelvosztályok azért is érdekesek, mert robusztusak abban az értelemben, hogy a nyelvosztályba tartozó nyelvek halmaza elég tág körben független attól, hogy milyen gépmodellt használunk az osztály definiálására. Ha például megkötjük, hogy csak egyetlen szalagja lehet a Turing-gépnek akkor is ugyanezekhez a nyelvhalmazokhoz jutnánk.

Általában fáradságos munka egy algoritmust Turing-gépre átírni. A P osztályba tartozáshoz tipikusan elég azt meggondolni, hogy van egy polinom sok lépést használó algoritmus annak eldöntésére, hogy egy szó beletartozik a nyelvbe. Ha igen, akkor ebből polinom időkorlátos Turing-gép is készíthető.

Ilyen alapon a már korábban tanult hatékony algoritmusokhoz tartozó nyelvek P-beliek.

3. Példa. *A* P *osztályba tartoznak például az alábbiak*

- *Az összefüggő gráfok nyelve. A nyelv szavai az irányítatlan, összefüggő gráfok szomszédossági mátrixai. Egy N csúcsú gráfnál ennek hossza N^2, tehát a bemenet hossza is N^2. A gráf összefüggősége szélességi bejárással $O(N^2)$ lépésben eldönthető (tehát ez valójában egy lineáris algoritmus). Az eljárás Turing-géppel is megvalósítható polinomiális időben, ezért ez a nyelv a* P *osztályban van.*

- *A páros gráfok nyelve. Erre is van polinomiális algoritmus, pl. a szélességi bejárás segítségével.*

- *Teljes párosítással rendelkező páros gráfok nyelve, mert a magyar módszer polinomiális.*
- *Azok a (G, s, t, k) alakú szavak, ahol G egy súlyozott gráf, s, t két csúcsa, k egy szám és G-ben van s és t között legfeljebb k súlyú út.*

Az NP osztályba tartozáshoz polinom időkorlátos nemdeterminisztikus Turing-gépet kell mutatnunk a nyelvhez. Hogy ezt is át lehessen fordítani a szokásosabb algoritmikus gondolkodásra, szükségünk lesz az NP egy másik jellemzésére.

4. Tétel (Tanú tétel). *Egy L nyelvre akkor és csak akkor teljesül, hogy $L \in$ NP, ha található olyan $c_1, c_2 > 0$ konstans és szópárokból álló L_1 nyelv, melyre $L_1 \in$ P és*

$$L = \{x : \text{ van olyan } y, \text{ hogy } |y| \leq c_1|x|^{c_2} \text{ és } (x, y) \in L_1\}$$

Egy fenti tulajdonságú y szót az x *tanújának* hívunk. A feltétel szerint az L_1 nyelvhez van polinom idejű Turing-gép. Ezt (vagy a neki megfelelő algoritmust) szokás az L-hez tartozó *hatékony tanúsítvány*nak hívni, hiszen megfelelő y segítségével tanúsítja, hogy $x \in L$.

Bizonyítás vázlat: Először tegyük fel, hogy $L \in$ NP. Ez azt jelenti, hogy van olyan polinom időkorlátos nemdeterminisztikus M Turing-gép, melyre $L(M) = L$. Tehát van olyan k szám, hogy minden n hosszú bemeneten a számítási utak hossza $O(n^k)$. Így, ha $x \in L$, akkor van az M-nek egy olyan számítási ága, ami elfogadó állapottal ér véget, és a hossza legfeljebb $c|x|^k$. Egy ilyen utat le lehet írni azzal, hogy megmondjuk, mikor melyik ágon menjünk tovább, ami lépésenként konstans sok bittel megtehető, hiszen a számítási fa minden pontban csak az átmeneti függvény által meghatározott konstans sok felé ágazhat. Egy elfogadó számítási út leírásának hossza $O(|x|^k)$, ezért ez a leírás jó lesz tanúnak ($c_2 = k$).

Az L_1 nyelv tehát álljon az olyan (x, y) párokból, hogy ha x-et mint az M gép bemenetét tekintjük, akkor y egy olyan ágat ír le a számítási fában, ami elfogadással ér véget. Ez az y a fentiek miatt teljesíti a hosszára tett feltételt. Azt meg, hogy ez valóban egy lehetséges elfogadó út, a megfelelő számítási lépések végrehajtásával az y hosszával arányos lépésben lehet ellenőrizni. Vegyük észre hogy ha $x \notin L$, akkor nem lesz olyan y, amire $(x, y) \in L_1$.

A másik irányhoz induljunk ki abból, hogy ha egy adott x-hez tekintjük az összes legfeljebb $c_1 \cdot |x|^{c_2}$ hosszú y sorozatot, és minden ilyen y-ra az (x, y) páron lefuttatjuk az L_1 nyelvet felismerő M' Turing-gépet, akkor minden egyes pár esetén ez polinomiális az idő. Igaz, az összes idő exponenciális lesz, mert exponenciálisan sok y van. Azonban kereshetjük a jó y-t nemdeterminisztikusan: az y generálása legyen a nemdeterminisztikus rész, utána már az M' determinisztikus, és akkor fogadja el a gép az x bemenetet, ha M' elfogadja az (x, y) párt. Az így összerakott Turing-gép az L nyelvet fogadja el, a nemdetermnisztikus szakasz és az utána levő rész is polinom idejű. □

4. Példa. NP-*beliek az alábbi nyelvek:*

- *A Hamilton-kört tartalmazó gráfok* HAM *nyelve:* HAM$\in$ NP, mert L_1-nek választható az olyan szópárok halmaza, ahol a pár első tagja egy gráf leírása, a második tag pedig a gráf csúcsainak egy olyan permutációja, ami Hamilton-kört alkot. (A csúcssorozat leírása történhet a $\log n$ hosszú bitsorozatok egymás után írásával.)

  Világos, hogy $L_1 \in$ P, mert azt könnyű eldönteni egy adott gráfról és adott számsorozatról, hogy ez valóban egy Hamilton-kör: ellenőrizni kell, hogy minden csúcs pontosan egyszer szerepelt és hogy a szomszédos csúcsok között, valamint az első és az utolsó csúcs között fut él a gráfban. Ez egy n csúcsú gráf esetén $O(n)$ lépéses algoritmussal (és Turing-géppel is polinom időben) megtehető.

  Az is világos, hogy csak akkor van egy x gráfleíráshoz jó pár, ha a gráfban van Hamilton-kör. Azt kell még belátnunk, hogy a tanú mérete polinomiális a gráf méretéhez képest, de ez is teljesül, hiszen a gráf leírása n^2, a tanú leírása pedig csak $O(n \cdot \log n)$ méretű.

- *A pozitív összetett számok bináris alakjaiból álló* ÖSSZETETT *nyelv:* Az L_1 nyelv az olyan (m, t) binárisan felírt pozitív egészekből álljon, amelyeknél $1 < t < m$ és m osztható t-vel, azaz egy t valódi osztó a tanú. Mivel az oszthatóság ellenőrzése polinom lépésben megvalósítható, ezért $L_1 \in P$ és természetesen t hossza legfeljebb annyi, mint m hossza.

- *A 3 színnel kiszínezhető gráfok* 3SZÍN *nyelve:* 3SZÍN $\in$ NP, mert L_1-nek választható az olyan szópárok halmaza, ahol a pár első tagja egy gráf leírása, a második tag pedig sorban a csúcsok színe.

  $L_1 \in$ P, mert könnyű eldönteni egy adott gráfról és egy adott színezésről, hogy ez a gráfnak egy jó színezése, csak azt kell ellenőriznünk minden élre, hogy a végpontjai különböző színt kapnak, és hogy összesen legfeljebb 3 színt használtunk, Ez pedig polinom időben (pl. $O(n^2)$ lépésű algoritmussal) megtehető.

  Az is világos, hogy pontosan akkor van egy x gráfhoz jó tanú, ha a gráfnak van jó színezése. A tanú mérete polinomiális a gráf méretéhez képest, hiszen csúcsonként konstans bittel, azaz $\Theta(n)$ hosszban a színezés megadható.

9. Definíció. *Egy L nyelv $\overline{L}$* komplementere *azokból a szavakból áll, amelyek nincsenek L-ben, azaz $\overline{L} = \{x : x \notin L\}$.*

5. Példa. ÖSSZETETT $= \overline{\text{PRÍM}}$*, ahol* PRÍM *a prímszámok bináris alakjából álló nyelvet jelöli.*

10. Definíció. *Jelölje* co NP *az* NP*-beli nyelvek* komplementereiből *álló halmazt, azaz* co NP $= \{L : \overline{L} \in \text{NP}\}$.

4. Megjegyzés. *Vigyázat, a* co NP *osztály nem az* NP *osztály komplementere! Ezért is más a jelölés.*

Vannak olyan nyelvek, amelyek egyik osztályba sem tartoznak bele (pl. a diagonális nyelv, amihez nincs is Turing-gép), és hamarosan látni fogjuk, hogy vannak olyan nyelvek, amelyek mindkét osztályba beletartoznak.

Szemléletesen, míg az NP nyelveknél a nyelvbe tartozásra van hatékony tanúsítvány, addig a co NP nyelveknél a nyelvbe nem tartozásra. Így van ez a PRÍM $\in$ co NP nyelv esetén is, mert azt, hogy a szám nem prím egy valódi osztó tanúsítja.

5. Tétel. P $\subseteq$ NP *és* P $\subseteq$ co NP

Bizonyítás: Ha $L \in$ P, akkor van olyan polinom időkorlátos M determinisztikus Turing-gép, amire $L(M) = L$. Ez az M tekinthető nemdeterminisztikusnak is, tehát van nemdeterminisztikus polinom időkorlátos gép is, azaz $L \in$ NP.

Másrészt, ha $L \in P$, akkor $\overline{L} \in P$ szintén teljesül, hiszen csak meg kell cserélni, melyik állapot elfogadó, melyik nem. Az előzőek szerint $\overline{L} \in$ NP, amiből a definíció szerint adódik, hogy $L \in$ co NP $\square$

5. Megjegyzés. *A 3. Tétel bizonyításából kiolvasható, hogy egy $p(n)$ polinom időkorlátos nemdeterminisztikus gépből exponenciális ($O(c^{p(n)})$) időkorlátos determinisztikus gépet készíthetünk.*

Szemléletesen, bár kissé pongyolán úgy is mondhatjuk, hogy egy nyelv akkor van P-ben, ha tetszőleges x esetén a nyelvbe tartozást gyorsan el tudjuk dönteni, míg egy nyelv akkor van NP-ben, ha megsejtve (ajándékba kapva, megálmodva, valami manó megsúgja, stb.) egy bizonyítékot a nyelvbe tartozásra, a bizonyíték gyorsan leellenőrizhető.

A számítástudomány egyik alapvető kérdése, hogy P és NP megegyezik vagy nem. Ha P = NP igaz lenne, ez azt jelentené, hogy ugyanolyan nehéz kitalálni egy jó bizonyítékot, mint ellenőrizni azt. Ez hihetetlennek tűnik, de nem ismert bizonyítás arra, hogy P $\neq$ NP (ahogyan arra sem, hogy P = NP igaz lenne).

Karp-redukció

Az NP-beli nyelvek vizsgálatakor hasznos lesz a következő fogalom. A definíció nem csak NP-beli nyelvekre alkalmazható, bár mi főleg ebben a körben fogjuk használni.

11. Definíció (Karp-redukció). *Legyen $L_1, L_2 \subseteq \Sigma^*$ két nyelv. Az L_1 nyelv* K*arp-redukálható (vagy* p*olinomiálisan visszavezethető) az L_2-nyelvre, ha létezik olyan $f : \Sigma^* \to \Sigma^*$ polinom időben számolható függvény, melyre $x \in L_1 \Leftrightarrow f(x) \in L_2$.*

Jelölése $L_1 \prec L_2$.

A jelölés arra utal, hogy, mint majd látni fogjuk, az L_2 nyelv legalább olyan nehéz, mint az L_1.

6. Megjegyzés. *Azt, hogy egy függvény polinom időben számolható, értsük úgy, hogy van rá polinomiális algoritmus, ami adott x-re kiszámolja az $f(x)$ értékét. Formálisan ezt is Turing-gépekkel lehet definiálni, a Turing-gépnek egy olyan változata kell hozzá, aminél nem azt nézzük, hogy elfogad-e, hanem azt, hogy megálláskor mi van az egyik, rögzített szalagján, mondjuk a második szalag tartalma a számítás eredménye.*

6. Példa. *A* 3SZÍN *nyelvhez hasonlóan definiálhatjuk a* 4SZÍN *nyelvet is: ez azokból az irányítatlan gráfokból (gráfok leírásából) áll, amelyek kiszínezhetők 4 színnel. Megmutatjuk, hogy* $3\text{SZÍN} \prec 4\text{SZÍN}$.

Ehhez egy megfelelő f függvényt kell megadnunk.

- *Ha az x nem egy gráfot ír le (pl. nem négyzetszám hosszú bitsorozat, amikor szomszédossági mátrixot várunk), akkor legyen $f(x) = x$. Ilyenkor $x \notin$ 3SZÍN és $f(x) \notin$ 4SZÍN.*
- *Egy G gráfhoz legyen G' az a gráf, amit úgy kapunk, hogy G-hez hozzáveszünk egy új csúcsot, amit összekötünk minden régivel. Legyen $f(G) = G'$.*

Ez az f függvény polinom időben kiszámolható, hiszen a G' (szomszédossági mátrixa) polinom lépésben elkészíthető G-ből. Másrészt világos, hogy G pontosan akkor színezhető ki 3 színnel, ha G' kiszínezhető 4 színnel.

NP-teljesség

12. Definíció. *Az L nyelv* NP-teljes, *ha $L \in$ NP és minden $L' \in$ NP esetén $L' \prec L$.*

Az NP-teljes nyelvek tekinthetők a legnehezebbeknek az NP osztályban, hiszen minden NP-beli nyelv visszavezethető rájuk.

Történetileg az első NP-teljes nyelv logikai, Boole-formulákból áll.

Egy *Boole-formula* 0 és 1 logikai konstansokból (jelentésük ,,hamis" és ,,igaz"), $x_1, \ldots, x_n$ logikai változókból és ezek $\overline{x_1}, \ldots, \overline{x_n}$ negáltjaiból, illetve az $\wedge$ (,,és") és a $\vee$ (,,vagy") műveleti jelekkel és zárójelekkel elkészített formula. Egy formula kielégíthető, ha tudunk úgy értéket adni a változóknak, hogy a formula értéke 1 legyen.

Egy Boole-formula *konjunktív normál formájú* vagy röviden *CNF*, ha az alábbi alakú:

$$(y_{i_1} \vee y_{i_2} \vee y_{i_3} \ldots) \wedge (y_{i_j} \vee y_{i_{j+1}} \vee y_{i_{j+2}} \ldots) \wedge \ldots$$

ahol minden y vagy egy változót vagy egy változó negáltját jelöli.

A 3CNF formula olyan CNF formula, ahol minden zárójelben legfeljebb 3 tag szerepel.

13. Definíció. *A kielégíthető formulák nyelve*

$$\text{SAT} = \{\varphi(x_1, \ldots, x_n) : \text{ van olyan behelyettesítés, hogy } \varphi(b_1, \ldots, b_n) = 1\}$$

A kielégíthető 3CNF formulák nyelve

$$3\text{SAT} = \{\varphi(x_1, \ldots, x_n) : \varphi \in SAT \text{ és a } \varphi \text{ formula 3CNF alakú}\}$$

7. Megjegyzés. *Ezt persze úgy kell érteni, hogy valamilyen módon a formulákat pl. 0-1 sorozatokká kódoljuk, és a megfelelő kódokból áll a nyelv.*

6. Tétel (Cook, Levin). *A* SAT *nyelv* NP*-teljes.*

Bizonyítás vázlat: Azt nem nehéz megmutatni, hogy a nyelv NP-beli, hiszen egy jó behelyettesítés jó tanú, aminek hossza és az ellenőrzéséhez szükséges idő is polinom.

A bizonyítás nehéz része, hogy minden NP-beli nyelv visszavezethető rá. Legyen $L \in$ NP tetszőleges. Ekkor van egy polinom időkorlátos nemdeterminisztikus M Turing-gép, amire $L(M) = L$. Ha x az M egy lehetséges bemenete, akkor ehhez a Karp-redukció egy formulát rendel, ami pontosan akkor kielégíthető, ha $x \in L$. Az alap ötlet az, hogy a formula lényegében az M működését írja le az x-en, és akkor kielégíthető, ha van egy elfogadó számítási út. Ennek a részleteit nem ismertetjük, csak ízelítőül például lesznek z_{iq} típusú változók, amelyeknek jelentése, hogy az i-edik lépés után M a q állapotban van. Ezekre teljesülni kell, hogy minden szóba jövő i-re pontosan egy olyan q van, amire $z_{iq} = 1$. Hasonlóan lesznek változók amelyek a szalagok tartalmát, illetve a fejek helyzetét írják le. Az átmeneti függvény alapján felírhatók a szabályok, hogyan változhatnak ezek az i-edik lépésről az $(i + 1)$-edikre. □

Mivel a tételbeli formula felírható 3CNF alakban is, ezért

7. Tétel. *A* 3SAT *nyelv* NP*-teljes.*

. . . . . .