ALGORITMUSOK ÉS BONYOLULTSÁGELMÉLET

Matematika MSc hallgatók számára

6. Előadás

Előadó: Hajnal Péter
Jegyzetelő: Szarvák Gábor 2011. március 8.

1. További példák

Példa. Legyen $L = $ 3-SZÍNEZHETŐSÉG $= \{\lceil G \rceil : \chi(G) \leq 3\}$. Ekkor 3-SZÍNEZHETŐSÉG $\in \mathcal{NP}$.

Valóban legyen T a következő tanúszalagos nem-determinisztikus Turing-gép: A tanúszalag tartalma legyen $\lceil c \rceil$, egy $c : V(G) \to \{1, 2, 3\}$ színező függvény kódja. A gép tegye a következőt. Ellenőrzi, hogy a tanúszalag tartalma tényleg 3 színezést kódol és minden összekötött csúcshoz különböző színeket rendel-e. Ha minden teszt stimmel, akkor ELFOGAD, különben ELVET.

Ez a gép bizonyítja állításunkat. Ez a rövid, egyszerű mondat a következő részállításokat „rejti”:

(i) $\lceil G \rceil \in L$ esetén van ELFOGADÓ futás (azaz „jó” tanúszalag tartalom).

(ii) $\lceil G \rceil \notin L$ esetén minden futás ELVET-ő.

(iii) A gép időigénye $|\lceil G \rceil|^{\mathcal{O}(1)}$.

Mindhárom állítás nagyon egyszerű.

Példa. Legyen $L = $ NEM-3-SZÍNEZHETŐSÉG, azaz az előző nyelv komplementere.

Emlékeztető (Hajós-tétel). $\chi(G) > 3$ akkor és csak akkor, ha van olyan G_0, G_1, …, G_ℓ gráfsorozat, hogy $G_\ell = G$, továbbá minden i-re G_i a következő lehetőségek valamelyikét teljesíti

- $G \simeq K_4$,
- valamely $j < i$ esetén G_i a G_j gráfból csúcs- vagy élhozzáadással kapható,
- valamely $j < i$ esetén G_i a G_j gráfból össze nem kötött csúcsok azonosításával kapható,
- valamely $j, j' < i$ esetén G_i a G_j és $G_{j'}$ gráfból Hajós-operációval kapható.

Leírunk egy T tanúszalagos nem-determinisztikus Turing-gépet:

A tanúszalag tartalma legyen egy G_1, G_2, …, G_ℓ gráfsorozat. A gép ellenőrzi, hogy a tanúszalag tartalma tényleg egy Hajós tételbeli bizonyító sororzat-e. Ha igen, akkor ELFOGAD (garantálja a NEM-3-SZÍNEZHETŐSÉG nyelvhez tartozást), különben ELVET.

A Hajós-tétel éppen azt állítja, hogy ez a Turing-gép a NEM-3-SZÍNEZHETŐSÉG nyelvet fogadja el.

A fenti gép bizonyítja, hogy NEM-3-SZÍNEZHETŐSÉG$\in \mathcal{NP}$? (Azaz igazolja-e, hogy 3-SZÍNEZHETŐSÉG $\in co\mathcal{NP}$?) NEM. A következő rejtett állításoknak kellene igaznak lenniük:

(i) $\lceil G \rceil \in L$ esetén van ELFOGADÓ futás (azaz „jó" tanúszalag tartalom).

(ii) $\lceil G \rceil \notin L$ esetén minden futás ELVET-ő.

(iii) A gép időigénye $|\lceil G \rceil|^{\mathcal{O}(1)}$.

Az első két állítás igaz, sőt (mint már mondtuk) éppen a Hajós-tétel. A harmadik állítást azonban nem tudjuk. Ami könnyen látható, az az hogy

(iii)' T-nek időigénye polinomiális $(\lceil G \rceil, \tau)$ hosszában, ahol τ a tanúszalag tartalma.

A futás $|\lceil G \rceil|$-ben is polinomiális lenne, ha polinomiális becslést tudnánk adni a Hajós-bizonyítás hosszára. Ilyen tétel egyelőre nincs.

Sőt általános a vélekedés/sejtés, hogy NEM-3-SZÍNEZHETŐSÉG$\notin \mathcal{NP}$. Ez azt is jelenti, hogy a fenti gép nem lehet polinomiális, azaz van olyan nem 3-színezhető gráf, amely szuperpolinomiális Hajós-bizonyítást követel.

2. Teljes problémák

Emlékeztető. Legyen $L \in_T \mathcal{NSPACE}(s(n))$, ahol $s(n)$ szép tárfüggvény. Ekkor ω inputra kiszámolható $\left(\vec{G}_{\omega,T}, A, Z\right)$ hármas kódja $\mathcal{O}(\log(n+1) + s(n) + 1)$ tárban (így $\cup_{\alpha\in\mathbb{N}}\alpha^{\log(n+1)+s(n)}$ időben) úgy, hogy $\omega \in L$ akkor és csak akkor teljesüljön, ha $(\overrightarrow{G}, A, Z) \in$ ELÉRHETŐSÉG. Azaz a redukciók nyelvén: tetszőleges $L \in \mathcal{NL}$ esetén $L \preceq_{\mathcal{L}}$ ELÉRHETŐSÉG.

Továbbá tudjuk, hogy ELÉRHETŐSÉG $\in \mathcal{NL}$.

A fent felidézett eredmények egy értelmezése: az ELÉRHETŐSÉG megragadja a teljes $\mathcal{NL}$ nyelvosztály erejét. Így „ha az ELÉRHETŐSÉG-ről tudtunk valami okosat mondani (például Savitch-algoritmus), akkor az egész $\mathcal{NL}$-ről is tudunk valamit mondani".

1. Következmény (Savitch-tétel). *(i)* $\mathcal{NL} \subseteq \mathcal{SPACE}(\log^2 n)$.

(ii) $\mathcal{NPSPACE} \subseteq \mathcal{PSPACE}$, *azaz* $\mathcal{PSPACE} = \mathcal{NPSPACE}$.

Bizonyítás. (i) Legyen L egy tetszőleges nyelv $\mathcal{NL}$-ben. Vezessük vissza az ELÉRHETŐSÉG nyelvre. A visszavezetés által legyárott inputon futtassuk a Savitch-algoritmust. Az eredő algoritmusból kiküszöbölhető a legyárott input leírása (lásd például az $\mathcal{L}$ redukció tranzitivitásának indoklását). Így az eredő algoritmus (amely egy tetszőleges választott $\mathcal{NL}$-beli nyelvet dönt el) $\mathcal{O}(\log^2 n)$ tárigényű.

(ii) Vegyünk egy tetszőleges $L \in \mathcal{NPSPACE}$ nyelvet. $\mathcal{PSPACE}$ algoritmussal kell megoldani. Ehhez a fenti algoritmust kell csak megismételni. A redukcióban nyert gráf leírásához polinomiális tár kell. A Savitch-algoritmus a kód hosszában négyzetes tárigényű. Polinom négyzete is polinom, így a tétel adódik. ■

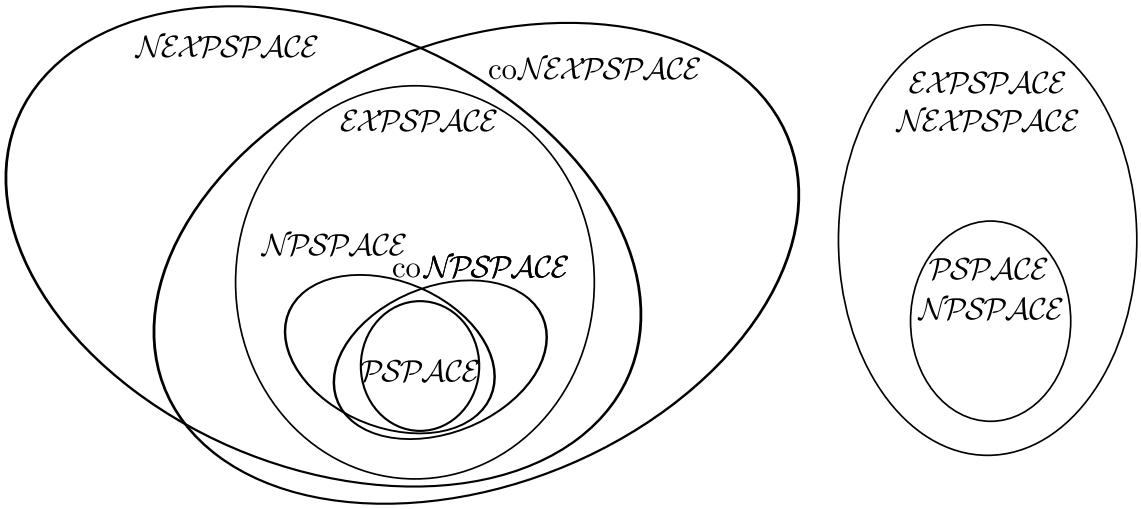
Természetesen a polinomiális tárkorláttal azért tudtunk kényelmesen dolgozni, mert polinom négyzete is polinom. Általában is megfogalmazható a következő tétel:

2. Következmény $^+$**.** *Legyen S szép tárfüggvények egy osztálya, ami zárt a négyzetre emelésre. Ekkor $\mathcal{NPSACE}(\cup_{s\in S}s(n)) = \mathcal{PSPACE}(\cup_{s\in S}s(n))$. Speciálisan $\mathcal{NPSPACE}(\cup_{s\in S}s(n))$ zárt a komplementálásra, azaz*

$$\mathcal{NPSPACE}(\cup_{s\in S}s(n)) = co\mathcal{NPSPACE}(\cup_{s\in S}s(n)).$$

Így speciálisan $\mathcal{EXPSPACE} = \mathcal{NEXPSPACE}$.

Az 1. ábran látható, hogyan egyszerűsödik az eddig megismert nyelvosztályok viszonya az eddigi megállapításaink alapján.



1. ábra.

A fenti okoskodás nagyon fontos. Hogy a gondolatmenetet kihasználjuk egy nagyon fontos fogalmat vezetünk be:

Definíció. $\widehat{L}$ nyelv *teljes* a $\mathcal{C}$ osztályban az $\mathcal{R}$ bonyolultságú rekurzióra, ha:

(i) $\widehat{L} \in \mathcal{C}$,

(ii) minden $L \in \mathcal{C}$ esetén $L \preceq_{\mathcal{R}} \widehat{L}$.

Egy speciális esetet kiemelünk.

Definíció. $\widehat{L}$ nyelv $\mathcal{NP}$*-teljes*, ha

(i) $\widehat{L} \in \mathcal{NP}$,

(ii) minden $L \in \mathcal{NP}$ esetén $L \preceq_{\mathcal{P}} \widehat{L}$.

Definíció. $\widehat{L}$ *nehéz* a $\mathcal{C}$ osztályra $\mathcal{R}$ bonyolultságú redukcióval, ha minden $L \in \mathcal{C}$ esetén $L \preceq_{\mathcal{R}} \widehat{L}$.

Azaz a nehézség a teljesség fogalma az (i) feltétel nélkül. Azaz nem követeljük meg az osztályhoz tartozást csak az osztály elemeinek visszavezethetőségét rá.

Korábbi eredményeinket az alábbi észrevétel az új fogalmak segítségével tömören fogalmaza meg.

Észrevétel. ELÉRHETŐSÉG $\mathcal{NL}$-teljes az $\mathcal{L}$-redukcióra.

Az alábbi észrevétel egyszerű, de fontos.

Észrevétel. Ha az L nyelv $\mathcal{NP}$-teljes, és $L \in \mathcal{P}$, akkor $\mathcal{NP} \subseteq \mathcal{P}$, azaz $\mathcal{P} = \mathcal{NP}$.

Valóban legyen K egy tetszőleges $\mathcal{NP}$-beli nyelv. Legyen R egy redukáló algoritmus, ami $K \preceq_{\mathcal{P}} L$-et igazolja. Legyen M egy $L \in \mathcal{P}$-t igazoló algoritmus, azaz az L nyelvhez tartozás problémáját polinom időben eldöntő algoritmus. Ekkor K polinom időben eldönthető. ω inputon számoljuk ki $R(\omega)$-t majd futtasuk az L-et eldöntő M algoritmust $R(\omega)$-n. Ezzel a K-hoz tartozást döntjük el. Futási időnk $|R(\omega)|$ egy polinomja, ami egyben $|\omega|$ egy polinomja, hiszen a polinomok zártak a kompozícióra.

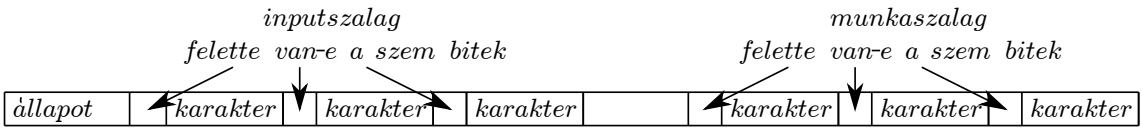
Emlékeztető. $L \in_T \mathcal{TIME}(t(n))/\mathcal{NTIME}(t(n))$ esetén mindig feltesszük, hogy $t(n)$ szép idő függvény, azaz Turing-géppel megvalósítható egy óra, ami n hosszú inputra „$t(n)$ idő után üt".

$\omega \in \Sigma^*$ inputon T futása a

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

konfigurációsorozat, ahol $\kappa_0(\omega)$ az ω-hoz tartozó kiinduló konfiguráció, κ_{i+1} a κ_i konfiguráció rákövetkezője, és κ_ℓ az első olyan konfiguráció, ahol az állapot ELFOGAD vagy ELVET.

Észrevétel 1. κ konfigurációk kódolhatók bitsorozatokkal.



2. ábra.

A fenti ábra (megjegyzéseivel) magáért beszél. A fej pozícióját is kell kódolnunk. Ezt „szétszórtuk", minden mező raktunk egy bitet. Így nem miden adott hosszúságú sorozat kódol konfigurációt. Az állapotot és karaktereket kódoló blokkkok hossza függ $|S|$, $|\Sigma|$ és $|\Gamma|$ értékétől. Minden estre konstans sok bit szükséges (ami a Turing-géptől függ). S elemeinek kódolására $\lceil \log_2 |S| \rceil$ bit legendő. Ha $|S|$ nem kettőhatvány, akkor lesznek olyan 0-1 bitsorozatok, amik a megfelelő pozíciókban állnak, de nem kódolnak állapotot. Ennek ellenére könnyű tervezni egy olyan tesztelő hálózatot, ami egy adott (megfelelő hosszú) kódról eldönti, hogy konfigurációt kódol-e.

Megjegyzés (FONTOS). A megállapodát úgy választhatjuk adott n hosszú ω input esetén $\lceil \omega \rceil$ hossza $\alpha_T \cdot n$ legyen. Ha T időigénye legfeljebb $t(n)$, akkor konfigurációk kódjának hossza $\beta_T \cdot n$ legyen. A továbbiakban n mindig rögzített és ennek megfelelően a megfelelő kódok hossza mindig ismert.

Észrevétel 2. $\lceil \kappa_{i-1} \rceil \to \lceil \kappa_i \rceil$ egyszerű hozzárendelés/függvény.

Az egyszerűség alatt azt értjük, hogy a függvény a következő egyszerű, kombinatorikus modellben, hatékonyan kiszámolható.

Definíció (Hálózat). Egy irányított gráf, amely

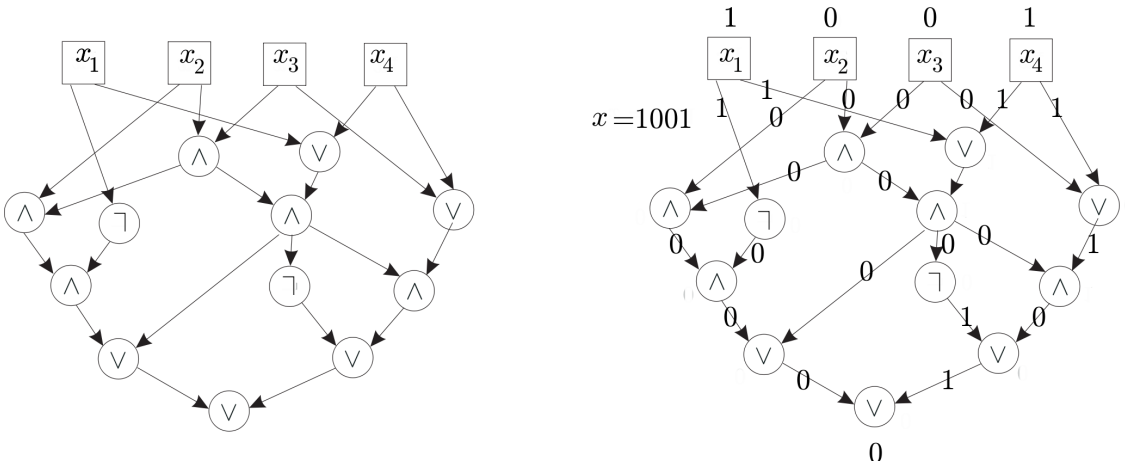
(i) irányított kör nélküli gráf (ez ekvivalens azzal, hogy lerajzolható úgy, hogy minden él két végpontja különböző magasságban legyen, és irányítása a magasabb végpont felöl az alacsonyabb felé mutasson),

(ii) Minden csúcs befoka $0, 1$ vagy 2, A 0 befokú csúcsok az *input-csúcsok*, a többi csúcsok *kapuk*. A csúcsok kifokára nincs feltétel.

(iii) Minden csúcsnak van egy címkéje: Input-csúcsok címkéje $\{x_1, x_2, \ldots, x_n\} \cup \{0, 1\}$-ből kerülnek ki. Az 1 befokú kapuk címkéje $\neg$. A 2 befokú kapuk címkéje $\vee$ vagy $\wedge$.

(iv) Egy vagy több speciális csúcs ki van jelölve. Nevük az *output-csúcsok*.

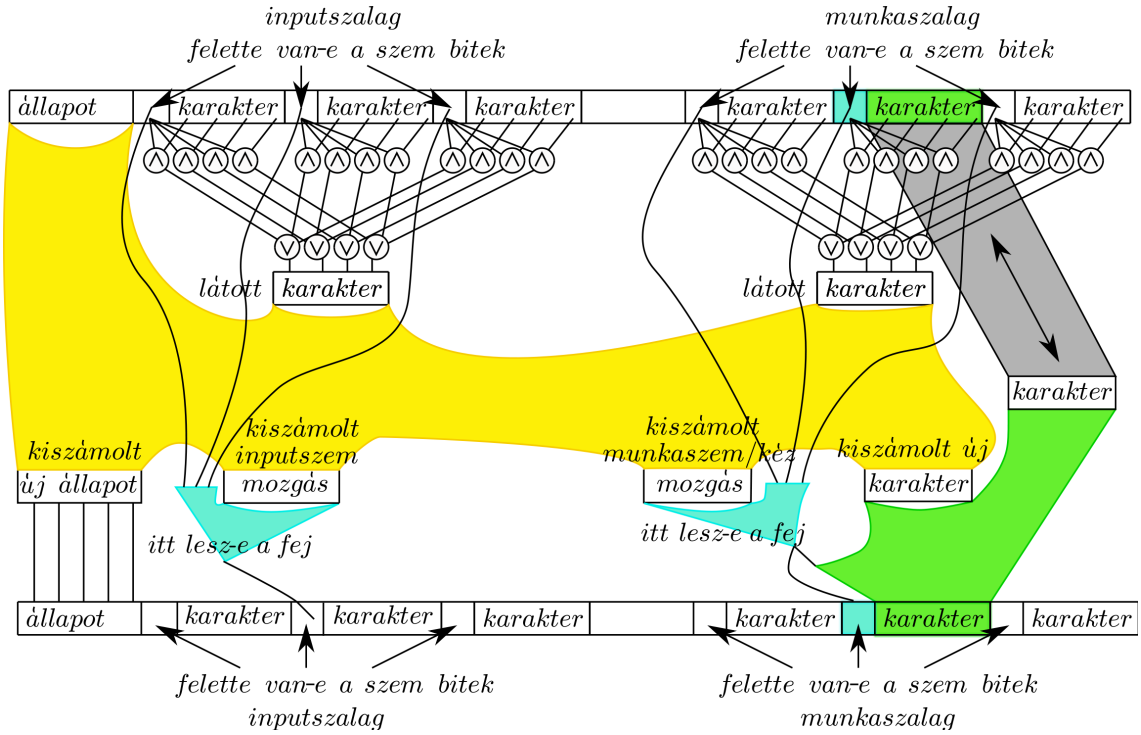
A fenti egy statikus leírása a hálózatnak. Úgy kell rá gondolni, mint egy program kódjára.



3. ábra. Bal oldalon a hálózat statikus képe, a jobb oldalon a dinamikus kép az $x = (1, 0, 0, 1)$ inputon. Az időbeliséghez „fentről lefelé kellolvasni a képet”.

Van egy dinamikus szemlélet is (mint a program-kód esetén a futás). Ekkor az input változóknak konkrét bit értékeket adunk. Így a hálózat „tetején lévő csúcsok” egy bit értéket kaptak. Az egyes csúcsok értékei az éleken az irányításnak megfelelően (megállapodásunk szerint lefelé) haladnak. A kapukban a befutó bitek a kapuk címkéje alapján (ami egy logikai jel) értelemszerűen összetevődnek. A most leírt rekurzió minden csúcshoz egy kiszámolt értéket/bitet rendel. A hálózat számolásának végeredménye az output-csúcsokhoz rendelt bitek.

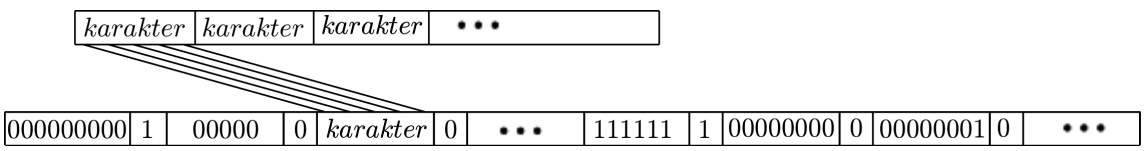
Most lássuk, hogy hogyan számolható ki ebben a modellben egy konfigurációt kódoló bitsorozatból a rákövetkező konfigurációt kódoló bitsorozat. A konstrukciónk egyszerű, de sok esetlegességet, megállapodást követel. Egy formális leírás helyett egy példán szemléltetjük milyen ötletek vezethetnek el egy megoldáshoz.



4. ábra. A sárga területben egy összetettebb számolás történik: konstans sok bitból számolunk ki konstans sok bitet (a konstanok függnek a Turing-géptől). Konkrét kidolgozása az átmenetifüggvénytől függ. Nem okozhat problémát akkor sem, ha az egyes bitek függését a nyilvánvaló DNF formula alapján írjuk fel. Ekkor is konstans sok kapuval dolgozva megoldjuk a feladatunkat. A világos kék területen számolódik kis a fej pozícióját leíró egyik bit. Ez attól függ, hogy a fej ott volt-e vagy valamelyik szomszédba állt, illetve merre írja eló mozgását az átmeneti függvény. A hálózatunk ezen részét konkrétan felírhatnánk, de ez a munka felesleges az előző megjegyzésünk alapján. Ez a kék rész minden fej-pozíció-bithez ott van. Az átláthatóság kedvéért rajzoltuk fel csak egyszer az input- és egyszer a munkaszalaghoz. A zöld területen számoljuk ki a munkaszalag új tartalmát. Minden munkaszalag mezőhöz tartozik egy ilyen zöld blokk (egyszerűség kedvéért csak egyet tüntettünk fel). Az új karaktertől, a régitől és attól az információtól függ, hogy a fej ott áll-e. Ez a rész is könnyen megvalósítható lenne, ha tudnánk Γ elemeinek kódolásához használt bitek ℓ számát ismernénk. A zöld területen az $f(\epsilon, k_0, k_1) = k_\epsilon$ függvényt számoljuk ki, ami $1 + 2\ell$ bitből számol ki ℓ-et.

Észrevétel 3. $\lceil\omega\rceil \to \lceil\kappa_0(\omega)\rceil$ egy egyszerű hozzárendelés/függvény.

Ez a korábbiaknál egyszerűbb észrevétel. Ismét egy ábrára utalunk.



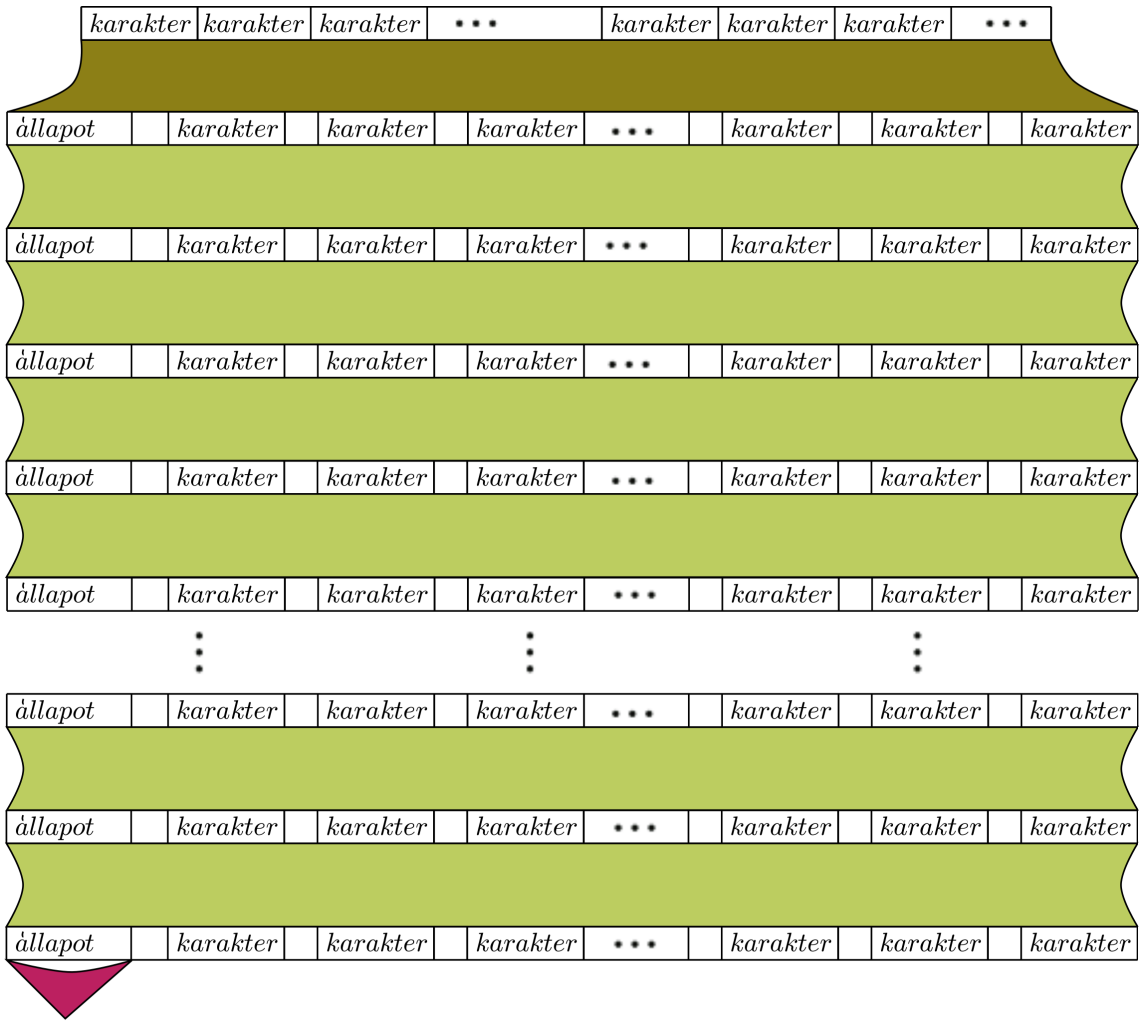
5. ábra.

Feltettük, hogy a START állapot kódja $00\ldots0$ (hossza $\lceil\log_2|S|\rceil$, példánkban 9).

Az inputszalagon $\triangleright$ kódja $00\ldots0$, a $\triangleleft$ kódja $11\ldots1$ (hosszaik $\lceil \log_2 |\Sigma| \rceil$,példánkban 5). Az munkaszalagon $\triangleleft$ kódja $0\ldots00$, a szűzkarakter kódja $0\ldots01$, (hossza $\lceil \log_2 |\Gamma| \rceil$ 8). Egy kissé bonyolúltabb ábrára van szükség, ha a tanúszalag tartalmával (nemdeterminisztikus eset) is számolunk. Ennek elképzelése mindenki számára egyszerűen megtehető.

Észrevétel 4. $t(n)$ szép, ezért feltehető, hogy T futása pontosan $t(n)$ idejű.

Ekkor az előző hálózatok összerakhatók az egész futást leíró hálózattá, amelyben egyetlen kapu által kiszámított bit jelzi, hogy a megfelelő futás ELFOGAD vagy ELVET állapottal zárul.



6. ábra.

A hálózat alján lévő tesztelés egyszerű lesz, ha az ELFOGAD állpotot a csupa 1 bitet tartalmazó sorozatttal kódoljuk. Ekkor „csak” ÉS kapukkal kapcsoljuk össze az állapotot kódoló biteket. Más megállpodás esetén sincs komoly probléma a megvalósításal (idegen szóval implementációval).

Észrevétel. Hálózatok kódolhatók. Egy H hálózat leírható $\lceil H \rceil$ kóddal, amely hossza $\mathcal{O}\left(s(H)^{\mathcal{O}(1)}\right)$, ahol $s(H)$ a hálózat mérete, a csúcsainak száma.

Összefoglalva:

3. Tétel. *Legyen T egy Turing-gép (determinisztikus vagy nem-determinisztikus). Legyen L a Turing-gép által elfogadott nyelv. Legyen $n \in \mathbb{N}$ egy input-méret.*

Definiáltunk

$$\widehat{H}_{T,n}(x_1, \ldots, x_N), \text{ illetve } \widehat{H}_{T,n}(x_1, \ldots, x_N, t_1, t_2, \ldots, t_M)$$

hálózatot, ahol $x = (x_1, \ldots, x_N)$ az ω input $b(\omega)$ kódja, illetve $(x, t) = (x_1, \ldots, x_N, t_1, t_2, \ldots, t_M)$ az input és tanúszalag tartalmának $b(\tau)$ kódja. $N = \mathcal{O}(n)$, $M = \mathcal{O}(t(n))$, ahol $t(n)$ egy becslés az időigényre n hosszú inputok esetén.

Ez a hozzárendelés olyan, hogy

(i) A determinisztikus esetben $\omega \in L$ akkor és csak akkor, ha $H_T(b(\omega))) = 1$. A nem-determinisztikus esetben pedig (ω, τ)-n akkor és csak akkor elfogadó a futás ha $\widehat{H}_{T,n}(b(\omega), b(\tau)) = 1$. Azaz $\omega \in L$ akkor és csak akkor, ha van olyan t bitsorozat, hogy $\widehat{H}_{T,n}(b(\omega), t) = 1$.

(ii) $\widehat{H}_{T,n}$ csúcsainak száma kisebb mint $\mathcal{O}(t^3(n))$, azaz a hálózat egy csúcsát $\mathcal{O}(\log t(n))$ bittel azonosíthatjuk.

(iii) 1^n -ből $\mathcal{O}(\log t(n))$ tárban kiszámolható $\lceil \widehat{H}_{T,n} \rceil$.

Definíció. HÁLÓZAT-KIÉRTÉKELÉS$= \{(\lceil H \rceil, \sigma) : H(\sigma) = 1\}$.

4. Következmény. *HÁLÓZAT-KIÉRTÉKELÉS $\mathcal{P}$-teljes $\mathcal{L}$-redukcióra.*

Bizonyítás. Tegyük fel, hogy $L \in_T \mathcal{P}$. Ekkor ω esetén számoljuk ki az ω-t kódoló $b(\omega)$ bitsorozatot és a T-hez és $|\omega|$ input-mérethez tartozó H hálózatot. A számításunk $\mathcal{L}$-beli. A korábbiak alapján $(\lceil H \rceil, b(\omega)) \in$HÁLÓZAT-KIÉRTÉKELÉS akkor és csak akkor, ha $\omega \in L$, ahogy bizonyítani kellett.

A legegyszerűbb algoritmus is mutatja, hogy HÁLÓZAT-KIÉRTÉKELÉS nyelv polinomiális időben eldönthető. ■

A fenti gondolatmenetnek további fontos következményei adódnak, amikre a jövő órán mutatunk rá.