Bevezetés a számítástudományba
Feladatgyűjtemény

Hunyadvári László

1999

1. Rekurzív függvények

A rekurzív függvényeket a kiszámíthatóság fogalmának axiomatizálására vezetjük be. Csak a természetes számok körében értelmezett függvényekkel foglalkozunk. (A véges sorozatokkal való reprezentálhatóság miatt minden más értelmes tartomány erre leképezhető.)

Legyen $\mathbf{N} = \{0, 1, 2, ...\}$ a természetes számok halmaza. $f : \mathbf{N}^k \longrightarrow \mathbf{N}$ azt jelöli, hogy az f függvény k természetes számból állít elő 1 természetes számot (vagyis k változós egyértékű függvény). Az elemi függvények a következők:

- $0 : \mathbf{N}^0 \longrightarrow \mathbf{N}$, a 0 konstans,
- $S : \mathbf{N}^1 \longrightarrow \mathbf{N}$, az eggyel növelés,
- $U_i^n : \mathbf{N}^n \longrightarrow \mathbf{N}$ n dimenzióban az i. komponensre való vetítés.

Mindegyik totális (mindenütt definiált függvény).

A megengedett függvénykonstrukciós szabályok az alábbiak:

- $H(g, h_1, h_2, \ldots, h_m)$, a függvénykompozíció, ahol g m változós, $h_1, h_2, \ldots, h_m$ n változós függggvények, míg az eredmény n változós függvény (a $h_1, h_2, \ldots, h_m$ g-be való helyettesítése),
- $R(g, h)$, a primitív rekurzió, ahol g $m-1$ változós, h $m+1$ változós függvények, míg az eredmény m változós függvény. Az $f = R(g, h)$ függvényhez az alábbi kiszámítási eljárás tartozik:

  $$f(x_1, x_2, \ldots, x_{m-1}, 0) = g(x_1, x_2, \ldots, x_{m-1}) \tag{1.1}$$

  $$f(x_1, x_2, \ldots, x_{m-1}, S(x_m)) = h(x_1, x_2, \ldots, x_{m-1}, x_m, f(x_1, x_2, \ldots, x_{m-1}, x_m)) \tag{1.2}$$

- $M(g)$, a minimalizáció, ahol g $m+1$ változós függvény, míg az eredmény m változós. Az $f = M(g)$ függvényhez az alábbi kiszámítási eljárás tartozik:

  $$f(x_1, x_2, \ldots, x_m) = Min\ z\ \{g(x_1, x_2, \ldots, x_m, z) = 0\}, \tag{1.3}$$

  feltéve, hogy ilyen z létezik és arra teljesül még, hogy $g(x_1, x_2, \ldots, x_m, j)$ definiált minden $j = 0, 1, \ldots, z-1$ -re.

A H és R műveletek totális függvényekből totális függvényt készítenek, míg M-el totális függvényből is készíthető parciális függvény.

Jelölje $\mathcal{F}_{prirek}$, illetve $\mathcal{F}_{parrek}$ azon függvények osztályát, melyek tartalmazzák az elemi függvényeket és zártak a helyettesítés és a primitív rekurzió illetve a helyettesítés, a primitív rekurzió és a minimalizáció műveletére. Legyen $\mathcal{F}_{rek}$ az $\mathcal{F}_{parrek}$ totális függvényekből álló részosztálya. Világos, hogy $\mathcal{F}_{prirek} \subseteq \mathcal{F}_{rek} \subseteq \mathcal{F}_{parrek}$.

Gyakorlatok

A következő gyakorlatokban megadjuk egy függvény intuitív definicióját, formális leírását a szokásos matematikai jelölésekkel, valamint az elemi függvényekből illetve a már korábban leírt függvényekből való előállítását.

1.0-1. Identitás függvény
$I(x) = x$
$I = U_1^1$

1.0-2. A k konstans ($k = 0, 1, \ldots$)
$k = k - 1 \ + \ 1$
$H(S, k-1)$ (Ne feledjük, hogy itt $k-1$ és k konstansok (0-változós függvények) nevei és nem értékek.)
Megj.: Az 1 konstans az úgynevezett 0. lépcsőfüggvény.

1.0-3. Csökkentés 1-el, $D(x)$
$D(0) = 0$ és $D(S(x)) = x$
$R(0, I)$

1.0-4. k. lépcsőfüggvény, l_k. (A k-nál kisebb értékekre 0-t, míg egyébként 1-et vesz fel, a 0.-at már láttuk.)
$l_k(0) = 0$ és $l_k(S(x)) = l_{k-1}(x)$ ($k-1$ és k itt is nevek, k nem 0)
$R(0, H(l_{k-1}, U_1^2))$
Megj.: l_1 az úgynevezett előjel függvény, melyet sg-vel fogunk jelölni.

1.0-5. Az összeadás
$x + 0 = x$ és $x + S(y) = S(x + y)$
$R(I, H(S, U_3^3))$

1.0-6. Pozitív különbség, $\dot{-}$ (0, ha a második argumentum kisebb az elsőnél, a szokásos különbség egyébként)
$x \dot{-} 0 = x$ és $x \dot{-} S(y) = D(x \dot{-} y)$
$R(I, H(D, U_3^3))$
Megj.: Világos, hogy a különbség abszolút értékét képező $|-|$függvény előállítása
$H(+, \dot{-}, H(\dot{-}, U_2^2, U_1^2))$

1.0-7. Az ellentett előjel függvény, $\overline{sg}$
$\overline{sg}(x) = 1 \dot{-} sg(x)$
$H(\dot{-}, 1, sg)$

1.0-8. A szorzás
$x * 0 = 0$ és $x * S(y) = x * y + x$
$R(0, H(+, U_3^3, U_1^3))$

1.0-9. Egész osztás,$\div$, a $x \div y$ a legkisebb olyan z egész, melyre $(z+1)y > x$
$min\ z\ \overline{sg}(S(z) * y \dot{-} x)$
$M(H(\overline{sg}, H(\dot{-}, H(\star, H(S, U_3^3), U_2^3), U_1^3)))$
Megj.: Az egész osztás a 0 osztóra természetesen definiálatlan.
Tulajdonságokat, relációkat is leírhatunk, ha igazsághalmazuk karakterisztikus függvényével azonosítjuk őket.

1.0-10. Egyenlőség, $\chi_=$ értéke 1, ha az argumentumok egyenlőek, 0 egyébként.
$\chi_=(x,y) = \overline{sg}(|x-y|)$
$H(\overline{sg}, |-|)$
A következő gyakorlatokban csak a függvény intuitív leírását adjuk meg, a kidolgozást az olvasóra bízva.

1.0-11. Maradék függvény, $rm(x,y)$, az x y-al való egész osztásának maradéka

1.0-12. Négyzetgyök függvény, $[\sqrt{x}]$

1.0-13. Legnagyobb közös osztó

1.0-14. Legkisebb közös többszörös

1.0-15. 2 alapú logaritmus, $[\log_2(x)]$

1.0-16. Hatványfüggvény

1.0-17. Nagyobb, $\chi_>$ értéke 1, ha az első argumentum nagyobb a másodiknál, 0 egyébként

1.0-18. Prímszám, χ_{prim} értéke 1, ha az adott szám prímszám, 0 egyébként

2. Turing-gépek

Egy T Turing-gép alatt a következő ötöst értjük:

$$T = \langle Q, \Sigma, \delta, q_0, B \rangle, \tag{2.1}$$

ahol Q az állapotok véges halmaza, Σ az input jelek véges halmaza, δ az állapotfüggvény, $q_0 \in Q$ a kezdőállapot és $B \notin \Sigma$ az üres cella jele. $\delta : Q \times \Sigma \cup \{B\} \longrightarrow Q \times \Sigma \cup \{B\} \times \{L, R, S\}$ függvény.

A gép egy véges memórájú központi egységből (lehetséges állapotait tartalmazza a Q) és egy mindkét irányban potenciálisan végtelen, cellákra osztott szalagból áll. A szalag celláiba Σ-beli un. szalagjelek írhatók. A szalagon minden pillanatban (így a kezdés során is) csak véges sok cella nem üres, a többieket úgy képzeljük, hogy a B üres jel van bennük. A szalaghoz tartozik egy olvasó-író fej, mely a szalagon jobbra, balra cellánként mozgatható. Az olvasó-író a kezdéskor a cellákba balról-jobbra beírt Σ^* un. input szó bal szélső elemére mutat.

A gép diszkrét időskálában működik. Egy ütem során kiolvassa a központi egység aktuális állapotát és az olvasó fej alatti cella tartalmát, majd ezektől függően átmegy egy új állapotba, átírja a cella tartalmát, majd a fejet jobbra, balra egy cellával elmozgatja vagy helyben hagyja. Egy ilyen ütem működését a δ átmeneti függvény írja le. $\delta(q, x) = (q', y, D)$ -ben q az aktuális állapot, x a kiolvasott input, q' az új állapot, y a beírt új jel és $D = L, R, S$ azt jelenti, hogy a fejet rendre balra, jobbra kell mozgatni ill. helyben kell hagyni.

A gép megáll, terminál, ha az aktuális állapot és input szimbólum mellett a δ átmeneti függvény nem definiált. A számítás ilyenkor befejeződik, mégpedig a szalagon levő nem B jelek sorozata lesz az eredmény. Azt, hogy a gépet kezdéskor az $u \in \Sigma^*$ szóval indítjuk el és megállás után a $v \in \Sigma^*$ az eredmény, a $\chi_T(u) = v$ jelöléssel fejezzük ki, ahol χ_T a gép által megvalósított leképezés.

Standard Turing-gépnek nevezzük az olyan Turing-gépet, melynek input szalagján a megálláskor egybefüggő a nem B-k sorozata és ha van egyátalán nem B jel, akkor az olvasó fej ezek közül a bal szélsőre mutat. Nem nehéz tetszőleges Turing-géphez olyan standard Turing-gépet konstruálni, mely az eredeti bemenő abécé felett pontosan ugyanazt a leképezést valósítja meg. (Új állapotok segítségével tömöríteni kell, majd az olvasó fejet a bal szélső nem B-re mozgatni.)

Turing-gépeket úgy használunk $f : \mathbf{N}^k \longrightarrow \mathbf{N}$ alakú függvények kiszámítására, hogy a számokat unáris számrendszerben kódoljuk. Pl. az n számot az $\hat{n}$ $n+1$ darab | jelből álló sorozattal reprezentáljuk, míg az $n_1, n_2, \ldots n_k$ számsorozatot $\hat{n}_1 * \hat{n}_2 * \ldots \star \hat{n}_k$ -val, s a megfelelő kódszóval indítjuk el a számítást végző gépet, elvárva természetesen, hogy az eredmény egy szám kódja legyen.

Ha $\mathcal{F}_{turing}$ jelöli a Turing-gépek által kiszámítható számsorozat bemenetű és számértékű függvények halmazát, akkor fennáll a következő egyenlőség: $\mathcal{F}_{turing} = \mathcal{F}_{parrek}$.

Elfogadott az a hipotézis, hogy a Turing-gépek kiszámító ereje minden lehetséges kiszámítási modellel egyenértékű. Ez a CHURCH-tézis.

Gyakorlatok

A következő gyakorlatokban megadott függvényekhez keresünk őket kiszámító standard Turing-gépeket. Csak a Turing-programot (az átmeneti függvényt) adjuk meg produkciók formájában ($\delta(q,x) = (q', y, Z)$ helyett $(q,x) \longrightarrow (q', y, Z)$-t írunk.

2.0-1. A k változós 0 függvény

$(q_0, x) \longrightarrow (q_0, B, J)$ $x = |, *$ mozgatás az első B-ig
$(q_0, B) \longrightarrow (q_1, |, S)$

2.0-2. Az S eggyel növelő függvény

$(q_0, |) \longrightarrow (q_0, |, L)$
$(q_0, B) \longrightarrow (q_1, |, S)$

2.0-3. Az U_k^n projekciós függvény

$(q_i, \|)$	$\longrightarrow$	(q_i, B, R)	$i = 0, \ldots k-2$	
$(q_i, \star)$	$\longrightarrow$	(q_{i+1}, B, R)	$i = 0, \ldots k-2$	Törlés a k. argumentumig
$(q_{k-1}, \|)$	$\longrightarrow$	$(q_{k-1}, \|, R)$		
$(q_{k-1}, \star)$	$\longrightarrow$	(q_t, B, R)		Átállás a k. argumentum mögé
(q_t, x)	$\longrightarrow$	(q_t, B, R)	$x = \|, \star$	
(q_t, B)	$\longrightarrow$	(q_v, B, L)		Törlés a végéig, ha $k \neq n$
(q_v, B)	$\longrightarrow$	(q_v, B, L)		
$(q_v, \|)$	$\longrightarrow$	$(q_s, \|, S)$		Ráállás a k. argumentum végére
$(q_s, \|)$	$\longrightarrow$	$(q_s, \|, L)$		
(q_s, B)	$\longrightarrow$	(q_z, B, R)		Ráállás az eredmény elejére
(q_{k-1}, B)	$\longrightarrow$	(q_s, B, L)		Ráállás az $k = n$. argumentum végére

2.0-4. A $\chi(u) = u \star u$ másoló függvény ($u \in \{x, y\}^\star$)

(q_0, z)	$\longrightarrow$	(q_0, z, R)	$z = x, y$	mozgatás az első B-ig
(q_0, B)	$\longrightarrow$	$(q_1, \star, L)$		$\star$ kiírása
(q_1, z)	$\longrightarrow$	(q_1, z, L)	$z = x, y$	
(q_1, B)	$\longrightarrow$	(q_2, B, R)		
(q_2, z)	$\longrightarrow$	(a_z, B, R)	$z = x, y$	A másolandó jel megjegyzése
(a_z, z')	$\longrightarrow$	(a_z, z', R)	$z' = x, y, \star$	
(a_z, B)	$\longrightarrow$	(b_z, z, L)	$z = x, y$	A megjegyzett jel kiírása
(b_z, z')	$\longrightarrow$	(b_z, z', L)	$z' = x, y, \star$	
(b_z, B)	$\longrightarrow$	(q_2, z, R)	$z = x, y,$	A megjegyzett jel visszaírása
$(q_2, \star)$	$\longrightarrow$	$(q_3, \star, L)$	$z = x, y$	
(q_3, z)	$\longrightarrow$	(q_3, z, L)	$z = x, y$	Haladás visszafelé
(q_3, B)	$\longrightarrow$	(q_3, B, R)	$z = x, y$	Ráállás az eredményre

2.0-5. Legyenek T_1 és T_2 Σ bemenetű standard Turing-gépek. Készítsük el a $\chi_{T_2}(\chi_{T_1})(u)$ helyettesítést megvalósító standard Turing-gépet. Legyen Q_1 ill. Q_2 a T_1 ill. T_2 állapothalmaza (feltehető, hogy metszetük üres), melyeknek elemei 1-el ill. 2-vel vannak felső cimkézve. Az új T gép állapothalmaza az eredeti gépek állapothalmazának uniója, kibővítve egy új, q_0-al jelölt kezdőállapottal. T programja a következő:

(q_0, z)	$\longrightarrow$	(q_0^1, z, S)	$z \in \Sigma \cup B$	Átmenet az első gép kezdőállapotába
T_1	programja			
(q, z)	$\longrightarrow$	(q_0^2, z, S)	minden párra, ahol T_1 megáll	Átmenet a második gép kezdőállapotába
T_2	programja			

A következő gyakorlatok standard Turing-gépeinek megszerkesztését az olvasóra bízzuk:

2.0-6. Az n változós k konstans függvény

2.0-7. A D 1-el csökkentő függvény

2.0-8. Az sg előjel függvény

2.0-9. Az összeadás

2.0-10. A $\dot{-}$ pozitív különbség függvény

2.0-11. A szorzás

2.0-12. Az egész osztás

2.0-13. Az egyenlőség függvény

2.0-14. A legnagyobb közös osztó

2.0-15. A 2 alapú egész logaritmus

2.0-16. $\varphi(u \star v) = uv$

2.0-17. $\varphi(w) = u$ ha $w = uu$

2.0-18. $\varphi(w) = w\,|^{|w|}$

2.0-19. $\varphi(u \star v) = w$ ha u és v hossza egyenlő és $w = u_1 v_1 u_2 v_2 \ldots u_{|u|} v_{|v|}$

2.0-20. $\varphi(u) = u^{-1}$

3. Formális nyelvek

Legyen Σ valamilyen véges szimbólumhalmaz. Az ilyen Σ halmazokat a továbbiakban ábécének nevezzük. A Σ ábécé felett képezhető véges sorozatok összességét Σ^*-al jelöljük, elemeit (Σ feletti) szavaknak, míg részhalmazait (Σ feletti) nyelveknek nevezzük. Legyenek u, v szavak és L, K nyelvek. $u \mid u \mid$-al jelölt hossza az u, mint sorozat hossza. u és v uv-vel jelölt konkatenációja az egymás után írásukkal kapott sorozat. u $i(\geq 0)$ példányának konkatenációját u^i-vel jelöljük ($u^0 = \varepsilon$, az ún. üres szó). Az összes lehetséges szorzatok halmazaként definiálhatjuk a LK konkatenációt és az L^i hatványt is (a 0, hatvány itt is ε). Nyelveken még használjuk az $L^\star$-al jelölt lezárási műveletet, mely az L^i hatványok uniója.

A nyelvek általában végtelen objektumok, ezért számítógépes feldolgozásukhoz szükséges, hogy valamilyen véges leírással egyértelműen jellemezni tudjuk őket. Ilyen leírás pl. a formális nyelvtan, a későbbiekben ismertetemdő véges automata, veremautomata, ill. a már látott Turing-gép is.

A $G =< T, N, \mathcal{P}, S >$ négyest formális nyelvtannak hívjuk, ahol T, N ábécék ($T \cap N = \emptyset$), a terminális jelek és a nyelvtani jelek ábécéi, $\mathcal{P}$ $p \longrightarrow q$ alakú ún. átirási szabályok véges halmaza, ahol $p, q \in (T \cup \Sigma)^\star$ és p tartalmaz legalább egy N-beli jelet. Az $\alpha \in (T \cup \Sigma)^\star$ szóból **közvetlenül** levezethető a G nyelvtanban a $\beta \in (T \cup \Sigma)^\star$ szó, ha létezik $g_1, g_2 \in (T \cup \Sigma)^*$ és $p \longrightarrow q \in \mathcal{P}$ szabály, hogy $\alpha = g_1 p g_2$ és $\beta = g_1 q g_2$ (jelölésben: $\alpha \underset{G}{\longrightarrow} \beta$). Az $\alpha \in (T \cup \Sigma)^\star$ szóból **közvetetten** levezethető a G nyelvtanban a $\beta \in (T \cup \Sigma)^\star$ szó, ha van olyan $k \in \mathbf{N}$ és vannak olyan $\delta_0, \delta_1, \ldots, \delta_k \in (T \cup \Sigma)^*$ szavak, hogy $\alpha = \delta_0, \beta = \delta_k$ és minden $i \in [0, k-1]$ esetén $\delta_i \underset{G}{\longrightarrow} \delta_{i+1}$ (jelölésben: $\alpha \overset{\star}{\underset{G}{\longrightarrow}} \beta$).

A G-által **generált nyelv** (jelölésben $L(G)$) a G-ben az S kezdőszimbólunból levezethető terminális szavakból áll. ($L(G) = \{u; u \in T^\star$ és $S \overset{\star}{\underset{G}{\longrightarrow}} u\}$).

A nyelvtanokat szabályaikra tett további kikötésekkel a $0, 1, 2, 3$, osztályokba sorolhatjuk, ami magukon a nyelveken is természetes módon indukál egy osztályozást.

0.típus, itt nincs további megkötés, azaz minden nyelvtan egyben 0-ás típusú is.

1.típus, ahol két szabályforma megengedett:

1. $a_1 A a_2 \longrightarrow a_1 q a_2$, ahol $a_1, a_2 \in (T \cup N)^*$ a környezet, $A \in N$, és $q \in (T \cup N)^+$ (azaz $q \neq \epsilon$). Ez a környezetfüggő szabályforma.

2. $S \longrightarrow \varepsilon$, ugyanis az 1. pont alapján nem lehetne ϵ-t levezetni. Megkötés: ha van ilyen szabály, akkor S nem fordulhat elő más szabály jobb oldalán. Ekkor S csupán ε levezetésére szolgál.

2.típus: Itt is két szabályforma lehetséges:

1. $A \longrightarrow q$, ahol $A \in N$, és $q \in (T \cup N)^+$. Ez a környezetfüggetlen szabályforma.

2. hasonlóan, mint az 1. típus 2. pontja.

Megjegyzés: Ez a típus az 1. típus megszorítása: a_1, a_2 mindig ϵ-nal egyenlő.

3.típus: Itt három szabályforma megengedett:

1. $A \longrightarrow aB$, ahol $A, B \in N$, és $a \in T$.

2. $A \longrightarrow q$, ahol $A \in N$ és $q \in T$.

3. $S \longrightarrow \epsilon$, hasonlóan, mint az 1. típus 2. pontja.

A különböző típusú nyelvtanok osztályokat alkotnak, melyeket rendre $\mathcal{G}_i$-vel jelölünk. Tehát: $\mathcal{G}_i$:= { i. típusú nyelvtanok }.

Viágos, hogy $\mathcal{G}_3 \subseteq \mathcal{G}_2 \subseteq \mathcal{G}_1 \subseteq \mathcal{G}_0$.

Hasonló módon a nyelveket is osztályozhatjuk: $\mathcal{L}_i := \{L \mid L$ nyelv, és van olyan $G \in \mathcal{G}_i$, amelyre $L(G) = L\}$.

Az előző nyelvtanhierarchia alapján $\mathcal{L}_3 \subseteq \mathcal{L}_2 \subseteq \mathcal{L}_1 \subseteq \mathcal{L}_0$. Ez az úgynevezett Chomsky-féle hierarchiatétel.

Szokás definiálni az 1.,2.,3. típusok kiterjesztését is, mely a generált nyelvek osztályát nem változtatja meg. A kiterjesztett 1. típusban az első szabályalak $p \longrightarrow q$, ahol $\mid p \mid \leq \mid q \mid$. A kitejesztett 2. típusnál az első szabályalakban $q = \varepsilon$ is megengedett, a második szabályalakra pedig nincs szükség. A kitejesztett 3. típusnál az első és második szabályalakban $a, q \in T^\star$ lehet és a harmadik szabályalakra nincs szükség.

Definiálhatjuk egy L nyelv szígorú típusát is, mely a legnagyobb olyan i, melyre $L \in \mathcal{L}_i$. Ennek megállapításához ad segítséget az alábbi két tétel:

Kis Bar-Hillel lemma: Ha egy L nyelv 3. típusú, akkor $\exists\ n > 0$ egész, hogy valahányszor $u \in L$ és $\mid u \mid \geq n$ akkor u felírható az $u = xyz$ alakban úgy, hogy $\mid xy \mid \leq n$, $\mid y \mid > 0$ és $\forall\ i = 0, 1, 2 \ldots$ mellett $xy^iz \in L$.

Nagy Bar-Hillel lemma: Ha egy L nyelv 2. típusú, akkor $\exists\ p, q > 0$ egészek, hogy valahányszor $u \in L$ és $\mid u \mid > p$ akkor u felírható az $u = xyzvw$ alakban úgy, hogy $\mid yzv \mid \leq q$, $\mid yv \mid > 0$ és $\forall\ i = 0, 1, 2 \ldots$ mellett $xy^izv^iw \in L$.

A nyelvtanokat a gyakorlatban széles körben alkalmazzák. Egyik fontos terület, ahol alapvető jelentőségük van: a programozási nyelvek szintaxisának leírása. Megmutatható, hogy a programozási nyelvek alapszimbólumai 3. típusú, a lényegi szintaxis elemei 2. típusú nyelvtannal írhatók le, míg a szemantikai elemzés során (pl. deklarációk feldolgozása) 1. típusú nyelvtanok hasznosíthatók.

Gyakorlatok

A következő gyakorlatokban nyelvekhez készítünk őket generáló nyelvtanokat. Csak a szabályrendszert adjuk meg azzal a konvencióval, hogy a terminális jelek mindig a magyar ábécé elejéről való kisbetűk vagy számjegyek, a nyelvtaniak pedig a magyar ábécé elejéről való nagybetűk, és mindig a legelső szabály baloldala a kezdőszimbólum.

3.0-1. Készítsünk nyelvtant, mely azokat az $u \in \{a, b\}^\star$ szavakat fogadja el, melyek a-val kezdődnek és b-vel végződnek.

Jelölje A azt, hogy egy a-t kell generálnunk, C, hogy a-t vagy b-t generálhatunk, míg B, hogy a záró b-t kell előállítani.

$A \longrightarrow aC$
$C \longrightarrow aC$
$C \longrightarrow bC$
$C \longrightarrow B$
$B \longrightarrow b$

A nyelvtan kiterjesztett 3. típusú, ezért a nyelv 3. típusú.

3.0-2.

Készítsünk nyelvtant, mely az $u = a^n b^n$ $n \geq 1$ szavakat fogadja el.

Ha van egy megfelelő u szavunk, az vagy ab vagy egy a, b párba zárt hasonló tulajdonságú szó. Ennek alapján az alábbi rekurzív szabályrendszer adódik:

$S \longrightarrow ab$
$S \longrightarrow aSb$

A nyelvtan 2. típusú, ezért a nyelv 2. típusú. Mivel láthatóan nem teljesül rá a kis Bar-Hillel lemma, ezért szigorúan 2. típusú is.

3.0-3.

Készítsünk nyelvtant, mely az $u = a^n b^n c^n$ $n \geq 1$ szavakat fogadja el.

Az előző nyelvtant az $a = a$ és $b = bc$ szereposztással alkalmazva $a^n(bc)^n$ alakú szavakat kaphatunk. Csak az összes b kötelező középre mozgatását kell megoldanunk néhány környezet függő szabállyal (b helyett B-t kell a csere szabályok miatt használnunk, különben nem lenne nyelvtani jel a baloldalon).

$S \longrightarrow aBc$
$S \longrightarrow aSBc$
$cB \longrightarrow Bc$
$aB \longrightarrow ab$
$bB \longrightarrow bb$

A nyelvtan kiterjesztett 1. típusú, ezért a nyelv 1. típusú. Mivel láthatóan nem teljesül rá a nagy Bar-Hillel lemma, ezért szigorúan 1. típusú is.

3.0-4.

Készítsünk nyelvtant, mely az olyan a, b-ből álló szavakat fogadja el, melyekben páros számú a és páratlan számú b van.

Jelölje az $A_{i,j} (i, j \in \{0, 1\}$ nyelvtani jel azt, hogy a szó eddig generált részében az a-k száma i, míg b-k száma j paritású.

$A_{i,j} \longrightarrow aA_{i+1,j}$
$A_{i,j} \longrightarrow bA_{i,j+1}$
$A_{0,1} \longrightarrow \varepsilon$

Itt a kezdőszimbólum $A_{0,0}$, $i, j \in \{0, 1\}$ és az összeadás moduló 2. A nyelvtan kiterjesztett 3. típusú, ezért a nyelv 3. típusú.

3.0-5.

Készítsünk nyelvtant, mely az olyan a-kból álló szavakat fogadja el, melyek hossza nemnulla négyzetszám.

A nyelvtan a négyzetszámokra vonatkozó következő előállítást használja :

$(n)^2 = (n-1)^2 - 1 + 2(n-1) + 2$. $n \geq 2$ esetén legeneráljuk az $Lx^{n-2}yxy^{n-2}R$ szót, majd az összes Y-t az elejére és X-et a végére visszük, miközben $(n-1)^2 - 1$-szer kell megcserélni őket egymással, behozva ugyanennyi a-t. A többi a rendre az $n-1$ darab X, az $n-1$ darab Y, valamint az 1-1 L és R-ből származik. Az $n = 1$ esetet az első szabály intézi el.

$S \longrightarrow a$
$S \longrightarrow LVR$
$V \longrightarrow XVY$
$V \longrightarrow YX$
$XY \longrightarrow YaX$
$aY \longrightarrow Ya$
$Xa \longrightarrow aX$
$LY \longrightarrow La$
$XR \longrightarrow aR$
$L \longrightarrow a$
$R \longrightarrow a$

A nyelvtan kiterjesztett 1. típusú, ezért a nyelv 1. típusú. Mivel láthatóan nem teljesül rá a nagy Bar-Hillel lemma, ezért szigorúan 1. típusú is.

3.0-6. Készítsünk nyelvtant a 4-el osztható bináris számok nyelvéhez.

3.0-7. Készítsünk nyelvtant azokhoz az a, b, c-t tartalmazó szavakhoz, melyekben a c-k száma 5-el osztva 2-t ad maradékul.

3.0-8. Készítsünk nyelvtant azon 4-es számrendszerben felírt számokhoz, melyek 3-al oszthatók.

3.0-9. Készítsünk nyelvtant ahhoz az a, b betűk feletti nyelvhez, melynek szavai ugyanannyi a-t és b-t tartalmaznak.

3.0-10. Készítsünk nyelvtant ahhoz az a, b betűk feletti nyelvhez, melynek szavai palindrómák (megegyeznek a megfordításukkal).

3.0-11. Készítsünk nyelvtant ahhoz az a, b betűk feletti nyelvhez, melynek szavai négyzetek (v^2 alakúak, ahol v a, b feletti szó).

3.0-12. Készítsünk nyelvtant ahhoz az a, b, c betűk feletti nyelvhez, melynek szavai ugyanannyi a-t és b-t és c-t tartalmaznak.

3.0-13.
Készítsünk nyelvtant, mely az olyan a-kból álló szavakat fogadja el, melyek hossza 2 hatvány.

A következő gyakorlatokban a nyelvtant adjuk meg, abból kell a generált nyelvet kitalálni.

3.0-14. Milyen nyelvet generál az alábbi szabályrendszer?
$S \longrightarrow aA$
$S \longrightarrow bB$
$S \longrightarrow cC$
$A \longrightarrow aA$
$A \longrightarrow bB$
$B \longrightarrow bB$
$B \longrightarrow cC$
$C \longrightarrow cC$
$C \longrightarrow aA$
$A \longrightarrow \varepsilon$
$B \longrightarrow \varepsilon$
$C \longrightarrow \varepsilon$

3.0-15. Milyen nyelvet generál az alábbi szabályrendszer?

$$
\begin{array}{lcl}
S & \longrightarrow & aB \\
S & \longrightarrow & bA \\
A & \longrightarrow & aS \\
A & \longrightarrow & a \\
A & \longrightarrow & bAA \\
B & \longrightarrow & bS \\
B & \longrightarrow & b \\
B & \longrightarrow & aBB
\end{array}
$$

4. Véges automaták

A véges determinisztikus automaták a Turing-gépek olyan leegyszerűsített változatai, ahol a szalagot csak olvasni lehet és a fej is csak jobbra léphet. Mivel a szalag nem írható, ezért a gép csak igen és nem választ tud adni attól függően, hogy az input elolvasása után végállapotba került vagy sem. Formálisan egy A véges determinisztikus automata alatt a következő ötöst értjük:

$$A = \langle Q, T, \delta, q_0, F \rangle, \tag{4.1}$$

ahol Q az állapotok véges halmaza, T az input jelek véges halmaza, δ az állapotfüggvény, $q_0 \in Q$ a kezdőállapot és $F \subseteq Q$ a végállapotok halmaza. $\delta : Q \times T \longrightarrow Q$ függvény.

A δ függvény egy ütem működését írja le. $\delta(q,t) = r$ azt jelenti, hogy a q állapotban a t bemenő jelet olvasva a gép átmegy az új r állapotba. A többlépéses működések leírására a δ függvényt kiterjesztjük $Q \times T^\star$-ra. A kiterjesztett δ-t használva formálisan is definiálhatjuk az automata által elfogadott nyelvet. $L(A) = \{u; u \in T^\star, \delta(q_0, u) \in F\}$

Jelölje $\mathcal{L}_{VDA}$ a véges determinisztikus automaták által elfogadott nyelvek osztályát. Bizonyítható, hogy $\mathcal{L}_{VDA} = \mathcal{L}_3$.

Gyakorlatok

A most következő gyakorlatokban egy adott nyelvhez kell őt elfogadó véges determinisztikus automatát készíteni. Az átmeneti függvényt adjuk meg táblázatos formában, a kezdőállapotot $\rightarrow$, a végállapotokat $\leftarrow$ jelöli.

4.0-1. A 4-gyel osztható bináris számok nyelve. Az állaprotban megjegyezzük az input beolvasott részének utolsó két jegyét.

$q \backslash x$	0	1
$\rightarrow q_\varepsilon$	q_0	q_1
$\leftarrow q_0$	q_{00}	q_{01}
q_1	q_{10}	q_{11}
$\leftarrow q_{00}$	q_{00}	q_{01}
q_{01}	q_{10}	q_{11}
q_{10}	q_{00}	q_{01}
q_{11}	q_{10}	q_{11}

4.0-2. A 3-mal osztva 1 maradékot adó 4-áris számok nyelve. Tudjuk, hogy egy 4-áris szám 3-mal akkor osztható, ha a jegyeinek összege 3-mal osztható. Ezért az állapotban megjegyezzük az input beolvasott részének maradékát moduló 3.

$q\backslash x$	0	1	2	3
$\rightarrow q_0$	q_0	q_1	q_2	q_0
$\leftarrow q_1$	q_1	q_2	q_0	q_1
q_2	q_2	q_0	q_1	q_2

4.0-3. Olyan a, b, c jeleket tartalmazó nem üres szavak, ahol a után nem jöhet c, b után nem jöhet a és c után nem jöhet b. Az állapotban megjegyezzük az input beolvasott részének utolsó betűjét.

$q\backslash x$	a	b	c
$\rightarrow q_\varepsilon$	q_a	q_b	q_c
$\leftarrow q_a$	q_a	q_b	q_h
$\leftarrow q_b$	q_h	q_b	q_c
$\leftarrow q_c$	q_a	q_b	q_h
q_h	q_h	q_h	q_h

4.0-4. Legyen T egy ábécé és $m \in T^\star$ egy szava, az ún. minta. $L : \{u; u \in T^\star$ és $u = pmr$ valamely $p, r \in T^\star\}$ az a nyelv, mely a mintát tartalmazó szavakból áll.

Az automatát a Knuth-Morris-Pratt-féle mintafelismerő algoritmus alapján készítjük el. A q_i állapot azt jelöli, hogy az input beolvasott részének vége hány jelre egyezik a minta elejével ($i = 1, 2, \ldots, \mid m \mid$). $q_{\mid m \mid}$ tehát azt jelenti, hogy valahol a minta előfordult, ezért ez az állapot a továbbiakban nem változik, s ő lesz a végállapot is. A többi állapot változásában az alább definiált un. eltolási függvény játssza a fő szerepet.

$f(q_i, x) = \underset{v \in Post(m_1 m_2 \ldots m_i x),\ Pre(m)}{Max \mid v \mid}$,

ahol $Post(w)$, $Pre(w)$ a w szó összes vég- ill. kezdőszeleteinek halmazát jelöli. Ennek alapján:

$\delta(q_i, x) = q_{f(i,x)}$ $i = 1, 2, \ldots, \mid m \mid -1,\ x \in T$

$\delta(q_{\mid m \mid}, x) = q_{\mid m \mid}$ $x \in T$

4.0-5. Legyen $L \subseteq T^*$ tetszőleges véges nyelv. (Konkrét példa lehet ilyen nyelvre valamely programozási nyelv alapszavainak összessége).

Az L-et elfogadó automata állapotai lényegében az L-beli szavak prefixei, vagyis a $Pre(L)$ halmaz, s lesz még egy hiba állapot a rossz szavaknak.

$\delta(q_w, x) = q_{wx}$ ha $wx \in Pre(L)$

$\delta(q_w, x) = q_h$ ha $wx \notin Pre(L)$

$\delta(q_h, x) = q_h$ ha $x \in T$

Kezdőállapot q_ε, $F = \{q_w;\ w \in L\}$.

4.0-6. A 8-cal osztható bináris számok nyelve.

4.0-7. A 4-gyel osztható 5-ös számrendszerben felírt számok nyelve.

4.0-8. A T ábécé feletti, két egymás utáni a betűt nem tartalmazó szavak nyelve.

4.0-9. A T ábécé feletti, két egyforma betűt egymás után nem tartalmazó szavak nyelve.

4.0-10. Legyen adott egy A T input ábécéjű véges, determinisztikus automata. Konstruáljunk véges, determinisztikus automatát a $T^* - L(A)$ nyelvhez.

4.0-11. Legyenek adottak az A_1, A_2 T input ábécéjű véges, determinisztikus automaták. Konstruáljunk véges, determinisztikus automatát az $L(A_1) \bigcup L(A_2)$ nyelvhez.

4.0-12. Legyenek valamely programozási nyelv azonosítói kisbetűvel kezdődő, kisbetűkkel, decimális számjegyekkel és _ jellel folytatódó, tetszőleges hosszú sorozatok. Készítsünk véges, determinisztikus automatát, mely felismeri őket.

4.0-13. Készítsünk véges, determinisztikus automatát a féllogaritmikus alakban
(előjelesegész.egész$E$előjeleslegfeljebbkétjegyűegész, pl. $-45.2E-2$)
felírt valós számok felismerésére.

5. Veremautomaták

A veremautomaták a véges determinisztikus automaták kétirányú továbbfejlesztései. Van bennük egyrészt egy veremmemóriaként használt, potenciálisan végtelen kiegészítő tár, melyben a feldolgozáshoz szükséges információkat tárolhatjuk. Másrészről a veremautomaták nemdeterminisztikusak, ami azt jelenti, hogy egy adott helyzetben több lehetséges működés közül is választhatnak. Egy input szót akkor fogad el a veremautomata, ha a kezdőállapotból indítva van olyan működés, melynek során a szó teljes végigolvasása után végállapotba kerül. Formálisan egy V veremautomata a következő hetes:

$$V = \langle Q, T, \Sigma, \delta, q_0, \sigma_0, F \rangle, \tag{5.1}$$

ahol Q az állapotok véges halmaza, T az input ábécé, Σ a veremábécé, δ az állapotfüggvény, $q_0 \in Q$ a kezdőállapot és $\sigma_0 \in \Sigma$ a verem kezdőjele, $F \subseteq Q$ a végállapotok halmaza. $\delta : \ Q \times (T \cup \{\varepsilon\} \times \Sigma \longrightarrow Q \times \Sigma^*$ nemdeterminisztikus függvény.

A veremautomata diszkrét időskálában működik. Egy ütem során kiolvassa a központi egység aktuális állapotát, az aktuális input szimbólumot és a verem tetejét, majd ezektől függően átmegy egy új állapotba, átírja a verem tetejét és az input fejet eggyel jobbra lépteti. Lehetőség van arra is, hogy egy ütemben ne történjen a szalagról olvasás, ilyenkor úgy tekintjük, hogy az olvasott szimbólum az ε. Egy ilyen ütem működését a δ átmeneti függvény írja le. $\delta(q, x, \sigma) \ni (q', \beta)$ -ban q az aktuális állapot, x a kiolvasott input vagy ε, ha az inputról nem olvastunk, σ a verem tetején lévő szimbólum, q' a lehetséges működésben kialakuló új állapot, míg β a lehetséges működésben a tető helyére írt jelsorozat. A veremautomata pillanatnyi leírása, konfigurációja alatt a következő hármast értjük $[q, v, \alpha]$, ahol q az aktuális állapot, v az input még nem elolvasott része (bal szélére mutat az olvasó), α a verem tartalma (bal széle a tető). A k konfigurációból a k' elérhető - jelölésben $k \xrightarrow[V]{*} k'$ - ha létezik olyan többlépéses működés, mely k-ból indul és k'-ben ér véget. A V által elfogadott nyelv formálisan a következő: $L(V) = \{u; u \in T^* \text{ és } \exists\ f \in F, \beta \in \Sigma^*\ [q_0 u, \sigma_0] \xrightarrow[V]{*} [f, \varepsilon, \beta]\}$

Egy V veremautomatát determinisztikusnak nevezünk, ha tetszőleges k konfigurációjából egy ütemben legfeljebb egy k' konfiguráció érhető el.

Jelölje $\mathcal{L}_V$ a nemdeterminisztikus veremautomaták által elfogadott nyelvek osztályát. Bizonyítható, hogy $\mathcal{L}_V = \mathcal{L}_2$.

Gyakorlatok

A most következő gyakorlatokban egy adott nyelvhez kell őt elfogadó nemdeterminisztikus - vagy ha ez lehetséges determinisztikus - veremautomatát készíteni. Az átmeneti függvényt a Turing-gépekhez hasonlóan szabályrendszerként adjuk meg. Mindig # a verem kezdőjele és q_0 a végállapot.

5.0-1. Készítsünk veremautomatát, mely az $u = a^n b^n \ n > 0$ szavakat fogadja el.

$(q_0, a, \#)$	$\longrightarrow$	$(q_0, a\#)$	az a-k behelyezése a verembe az első b-ig
(q_0, a, a)	$\longrightarrow$	(q_0, aa)	
(q_0, b, a)	$\longrightarrow$	(q_1, ε)	az a-k számának egyeztetése a b-k számával
(q_1, b, a)	$\longrightarrow$	(q_1, ε)	az a-k számának egyeztetése a b-k számával
$(q_1, \varepsilon, \#)$	$\longrightarrow$	$(q_2, \#)$	ha egyezett, átmenet a q_2 végállapotba

5.0-2. Készítsünk veremautomatát, mely a [,] feletti nemüres helyes zárójelezéseket fogadja el.

$(q_0, [, \#)$	$\longrightarrow$	$(q_0, [\#)$	az [-k behelyezése a verembe
$(q_0, [, [\ )$	$\longrightarrow$	$(q_0, [[\ )$	
$(q_0,], [\ )$	$\longrightarrow$	(q_0, ε)	a]-t egyeztetjük a veremmel
$(q_0, \varepsilon, \#)$	$\longrightarrow$	$(q_1, \#)$	ha egyezett, átmenet a q_1 végállapotba

5.0-3. Készítsünk veremautomatát, mely a egy T ábécé feletti palindróma (megegyezik a megfordításával) szavakat fogadja el.

$(q_0, t, \#)$	$\longrightarrow$	$(q_0, t\#)$	$t \in T$	berakás a verembe a közepéig
(q_0, t, t')	$\longrightarrow$	(q_0, tt')	$t, t' \in T$	
(q_0, ε, t')	$\longrightarrow$	(q_1, t')	$t \in T \cup \{\varepsilon\}, t' \in T$	nemdeterminisztikus áttérés a vége egyeztetésére
(q_1, t, t)	$\longrightarrow$	(q_1, ε)	$t \in T$	egyeztetés a veremmel
$(q_1, \varepsilon, \#)$	$\longrightarrow$	$(q_2, \#)$		ha egyezett,
&			átmenet a q_2 végállapotba	

5.0-4. Készítsünk veremautomatát, mely az olyan $+, \star$ műveleteket, [,] zárójeleket tartalmazó helyes kifejezéseket fogadja el, ahol az argumentumok helyén mindenütt a áll.

$(q_0, [, \#)$	$\longrightarrow$	$(q_0, [\#)$		a [-k behelyezése a verembe
$(q_0, [, [)$	$\longrightarrow$	$(q_0, [[)$		
$(q_0, a, \#)$	$\longrightarrow$	$(q_1, \#)$		operandus jött
$(q_0, a, [)$	$\longrightarrow$	$(q_1, [)$		
$(q_1,], [)$	$\longrightarrow$	(q_1, ε)		a]-t egyeztetjük a veremmel
$(q_1, \triangleleft, \sigma)$	$\longrightarrow$	(q_0, σ)	$\triangleleft = +, \star,\ \sigma = [, \#$	műveleti jel jött
$(q_1, \varepsilon, \#)$	$\longrightarrow$	$(q_2, \#)$		helyes kifejezésre
				átmenet a q_2 végállapotba

5.0-5. Készítsünk veremautomatát, mely a $[,], \{, \}$ zárójelpárok feletti nemüres helyes zárójelezéseket fogadja el.

5.0-6. Készítsünk veremautomatát, mely a $u = a^n b^{2n} \ n > 0$ nyelvet fogadja el.

5.0-7. Készítsünk veremautomatát, mely az a, b ábécé feletti olyan nemüres szavakat fogadja el, melyekben az a és b betűk száma egyenlő.

Az előbb definiált közönséges veremautomatához hasonlóan definiálhatjuk azokat a veremautomatákat, melyekben több vermet lehet tárolásra használni. Ha egy gépben $r \geq 0$ verem van, akkor r-veremről beszélünk. Egy input szót itt is akkor fogadunk el, ha a kezdőállapotból az adott szóval, mint inputtal elindítjuk, akkor van olyan átmenet, melynek során a szó teljes elolvasása után végállapotba kerülünk.

Jelölve $\mathcal{L}_{rV}$-vel a nemdeterminisztikus r-veremautomaták által elfogadott nyelvek osztályát, bizonyítható, hogy $\mathcal{L}_{rV} = \mathcal{L}_0 \ \forall \ r \geq 2$.

5.0-8. Készítsünk 2-veremautomatát, mely az T ábécé feletti nemüres dadogós (vv alakú valamely $v \in T^*$-re) szavakat fogadja el.

$(q_0, t, \#, \#)$	$\longrightarrow$	$(q_0, t\#, \#)$	$t \in T$	berakás az 1. verembe a szóközepig
$(q_0, t, t', \#)$	$\longrightarrow$	$(q_0, tt', \#)$	$t, t' \in T$	
$(q_0, \varepsilon, t', \#)$	$\longrightarrow$	$(q_1, t', \#)$	$t' \in T$	nemdeterminisztikus áttérés a vége egyeztetésére
$(q_1, \varepsilon, t, t')$	$\longrightarrow$	(q_1, ε, tt')	$t \in T, t' \in T \cup \{\#\}$	megfordítás a 2. verembe
$(q_1, \varepsilon, \#, t')$	$\longrightarrow$	$(q_2, \#, t')$	$t' \in T \cup \{\#\}$	megfordítás a 2. veremben
$(q_2, t, \#, t)$	$\longrightarrow$	$(q_2, \#, \varepsilon)$	$t \in T$	a vége egyeztetése a 2. veremmel
$(q_2, \varepsilon, \#, \#)$	$\longrightarrow$	$(q_3, \#, \#)$		ha egyezett, átmenet a q_3 végállapotba

Világos, hogy az így elkészített gép nemdeterminisztikus.

5.0-9. Készítsünk elfogadó 2-vermet az $u = a^n b^n c^n$ $n > 0$ alakú szavakból álló nyelvhez.

5.0-10. Készítsünk elfogadó 2-vermet az $u = a^{n^2}$ $n > 0$ alakú szavakból álló nyelvhez.

5.0-11. Készítsünk elfogadó 2-vermet az $u = a^{2^n}$ $n \geq 0$ alakú szavakból álló nyelvhez.

A Turing-gépek is használhatók nyelvek elfogadására úgy, hogy megállásukkor az igen ill. a nem kódjának valamelyike van a szalagon. Ennél egyszerűbb és az előzővel bizonyíthatóan megegyező hatóerejű, ha a Turing-gépeknél is bevezetjük a végállapotok fogalmát. A Turing-gép egy input szót elfogad, ha az őt tartalmazó kezdőhelyzetből indulva végállapottal terminál.

Jelölve $\mathcal{L}_{turing}$-gal a Turing-gépek által elfogadott nyelvek osztályát, bizonyítható, hogy $\mathcal{L}_{turing} = \mathcal{L}_0$.

5.0-12. Készítsünk Turing-gépet, mely a T ábécé feletti nemüres dadogós ($v \star v$ alakú valamely $v \in T^*$-re) szavakat fogadja el.

A megoldás az első v-ben lévő betűket megjegyezi az állapotkomponensben, majd B-re cseréli őket. Elszalad jobbra a *-ok utáni első jelre, azt összeveti a megjegyzett jellel. Egyezőség esetén átcseréli *-ra, visszamegy az első B-ig és majd onnan jobbra lépve újra egyezetet. Vége akkor van, amikor az első v elfogyott, vagyis az egyeztetést kezdeni akaró q_0 állapot *-ra mutat. Ha ettől a *-tól jobbra csupa * van, akkor a szó jó volt.

(q_0, z)	$\longrightarrow$	(a_z, B, R)	$z \in T$	a hasonlítandó jel megjegyzése
(a_z, z')	$\longrightarrow$	(a_z, z', R)	$z, z' \in T$	jobbra lépés a *-ig
$(a_z, \star)$	$\longrightarrow$	$(b_z, \star, R)$	$z \in T$	
$(b_z, \star)$	$\longrightarrow$	$(b_z, \star, R)$	$z \in T$	a * sorozat átlépése
(b_z, z)	$\longrightarrow$	$(q_v, \star, L)$	$z \in T$	egyeztetés
(q_v, z')	$\longrightarrow$	(q_v, z', L)	$z' \in T \cup \star$	vissza az elejére
(q_v, B)	$\longrightarrow$	(q_0, B, R)		a következő egyezetetendőre lépés
$(q_1, \star)$	$\longrightarrow$	$(q_1, \star, R)$		nincs mit egyeztetni, *-ok átlépése
(q_1, B)	$\longrightarrow$	(q_2, B, S)		átmenet a q_2 végállapotba