Veremautomaták

Kiegészítő anyag az Algoritmuselmélet tárgyhoz

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

Friedl Katalin
BME SZIT
friedl@cs.bme.hu

2022. február 24.

A veremautomata a véges automatának egy olyan kiterjesztése, amikor az állapotokon kívül az automata, egy vermet (LIFO tulajdonságú tárolót) is használ memóriaként. A veremautomata is elsősorban nyelvek felismerésére szolgál, és mint látni fogjuk, többet tud, mint a véges automata.

1. Nemdeterminisztikus veremautomata

A későbbi jelölések egyszerűsítésére, ha Σ egy ábécé, akkor vezessük be a

$$\Sigma_0 = \Sigma \cup \{\varepsilon\}$$

jelölést.

1. Definíció. *A* veremautomata *(vagy* PDA*) egy* $M = (Q, \Sigma, \Gamma, q_0, Z_0, F, \delta)$ *hetessel írható le, ahol*

- Q *egy véges, nem üres halmaz, az automata* állapotai*nak halmaza.*
- Σ *egy véges, nem üres halmaz, az automata (bemeneti)* ábécé*je.*
- Γ *egy véges, nem üres halmaz, a* verem ábécé*je vagyis a lehetséges verem-szimbólumok halmaza.*
- $q_0 \in Q$ *a* kezdő állapot*.*
- $Z_0 \in \Gamma$ *a verem* kezdő szimbóluma*.*
- $F \subseteq Q$ *az* elfogadó *állapotok halmaza.*
- δ *az* állapotátmeneti függvény*,* $\delta(q, a, A) \subseteq Q \times \Gamma^*$*, ahol* $q \in Q$*,* $a \in \Sigma_0$ *és* $A \in \Gamma_0$*.*

Mivel az állapotátmeneti függvény értéke egy halmaz, az itt definiált veremautomata *nemdeterminisztikus*. Hiányos is lehet, amennyiben $\delta(q, a, A) = \emptyset$ valamely (q, a, A) helyen.

A veremautomata *működése* egy adott $w \in \Sigma^*$ bemeneten: kezdéskor a PDA a q_0 kezdő állapotban van, a veremben egyetlen szimbólum, a Z_0 található. Minden lépésben az aktuális állapot, a w soron következő karaktere és a veremben levő legfelső szimbólum alapján változik az állapot és a verem tetejére berakunk egy, a Γ karaktereiből álló véges hosszú sorozatot. Pontosabban, ha az i-edik lépés előtt r_{i-1} állapotban van az automata, akkor a bemeneti w szóból 0 vagy 1 karaktert ($a_i \in \Sigma_0$) olvas, a verem tetejéről levesz 0 vagy 1 szimbólumot ($A_{i-1} \in \Gamma_0$). Az átmeneti függvény által adott $\delta(r_{i-1}, a_i, A_{i-1})$ halmaz egy eleme alapján folytatódik a számítás. Ha ez $(r_i, t_i) \in \delta(r_{i-1}, a_i, A_{i-1})$, akkor az automata az r_i állapotba kerül és a verem tetejére (az A_{i-1} helyére) kerül a $t_i \in \Gamma^*$ szimbólumsorozat. (Kezdéskor $r_0 = q_0$, $A_0 = Z_0$ vagy $A_0 = \varepsilon$)

Formálisan

2. Definíció. *Legyen $w = a_1 a_2 \dots a_m$, ahol minden $a_i \in \Sigma_0$. Az M veremautomata* számítása *a w szón egy olyan $r_0, r_1, \dots, r_m \in Q$ állapotsorozat és $s_0, s_1, \dots, s_m \in \Gamma^*$ sorozata a veremtartalmaknak, melyekre teljesül, hogy*

- *$r_0 = q_0$ és $s_0 = Z_0$,*
- *minden $1 \le i \le m$ esetén, ha $s_{i-1} = A_{i-1}\tau_{i-1}$, ahol $A_{i-1} \in \Gamma_0$, valamint $\tau_{i-1} \in \Gamma^*$, akkor*
  - *$(r_i, t_i) \in \delta(r_{i-1}, a_i, A_{i-1})$ és*
  - *$s_i = t_i \tau_{i-1}$*

A definícióban, és később is, a verem tartalmát leíró s szót úgy kell érteni, hogy s-ben a verem tartalma felülről lefelé szerepel, azaz s első karaktere az, ami a verem tetején van, az utolsó karaktere pedig, ami a verem alján.

1. Megjegyzés. *A véges automatánál az m a w szó n hosszával volt egyenlő. Itt azonban lehetséges, hogy $m > n$, mivel előfordulhatnak olyan lépések, amikor a veremautomata átmenetében egy bemeneti karakter helyett ε szerepel, ilyenkor a veremautomata nem olvas a bemenetről.*

Mivel a verembe írt karaktersorozat lehet az üres is, $\delta(q, a, A) = (q', \varepsilon)$ típusú átmenetekkel a verem kiüríthető. Amikor üres a verem, akkor olyan $\delta(q, a, B)$ átmenetet nem lehet alkalmazni, ahol $B \neq \varepsilon$. Ha a veremautomata *konfigurációjára* bevezetjük a (q, x, s) jelölést, ahol q az aktuális állapot, x a bemenetből még hátra levő rész és s a verem tartalma, akkor az i-edik lépés leírható

$$(r_{i-1}, a_i a_{i+1} \dots a_m, A_{i-1}\tau_{i-1}) \Rightarrow (r_i, a_{i+1} \dots a_m, t_i \tau_{i-1})$$

alakban is. Itt a_i lehet ε, amikor nem történik tényleges továbblépés a bemeneten. Amikor $A_{i-1} = \varepsilon$, akkor a veremből nem veszünk ki semmit, csak írunk bele (pontosabban ha $t_i = \varepsilon$, akkor a verem nem változik, és ténylegesen írunk bele, ha $t_i \neq \varepsilon$).

3. Definíció. *Azt mondjuk, hogy az M veremautomata* elfogadja *a $w \in \Sigma^*$ szót, ha van olyan az $r_0, \cdots, r_m \in Q$ állapotsorozattal és a $s_0, \cdots, s_m \in \Gamma^*$ veremtartalmak sorozatával leírható számítása, hogy $r_m \in F$ teljesül.*

Az elfogadáshoz hozzá tartozik, hogy a w szó feldolgozása közben a számítás nem akad el, azaz a veremautomata az utolsó betűt is feldolgozza. Utána még végezhet $\delta(q, \varepsilon, A)$ típusú átmeneteket, csak az kell, hogy közben érintsen elfogadó állapotot is.

A konfigurációk segítségével az elfogadás úgy írható le, hogy a kezdeti helyzetből véges sok lépésben eljuthatunk egy elfogadóba:

$$(q_0, w, Z_0) \Rightarrow \cdots \Rightarrow (q, \varepsilon, s) \text{ ahol } q \in F, s \in \Gamma^* \text{ tetszőleges}$$

Ha a számítás elakad mielőtt egy ilyen konfigurációhoz ér, akkor nem elfogadó.

4. Definíció. *Az M veremautomata által* elfogadott nyelv *azoknak a Σ feletti szavaknak a halmaza, melyeket M elfogad. Jele $L(M)$.*

1. Példa. *A következő automata az $L = \{\mathtt{0}^n\mathtt{1}^n : n \geq 0\}$ nyelvet fogadja el. Az elv elég egyszerű, a vermet mint egy számlálót használjuk: amíg* `0` *karaktert olvasunk, berakunk egy-egy szimbólumot a verembe, az* `1` *karaktereknél pedig mindig kiveszünk egyet, és az aktuális állapot jelzi, hogy a beíró vagy a kivevő fázisban vagyunk.*

Legyen $\Gamma = \{Z, 0\}$, $Q = \{q_0, q_1, q_2, q_3\}$ és $M = (Q, \{\mathtt{0}, \mathtt{1}\}, \Gamma, q_0, Z, \{q_0, q_3\}, \delta)$, ahol

$$\begin{aligned}
\delta : (q_0, \mathtt{0}, Z) &\mapsto \{(q_1, 0Z)\} \\
(q_1, \mathtt{0}, \varepsilon) &\mapsto \{(q_1, 0)\} \\
(q_1, \mathtt{1}, 0) &\mapsto \{(q_2, \varepsilon)\} \\
(q_2, \mathtt{1}, 0) &\mapsto \{(q_2, \varepsilon)\} \\
(q_2, \varepsilon, Z) &\mapsto \{(q_3, Z)\}
\end{aligned}$$

a többi átmenet esetén δ értéke az üres halmaz.

A $w =$ `000111` *szón az automata egy számítása a következő:*

$$(q_0, \mathtt{000111}, Z) \Rightarrow (q_1, \mathtt{00111}, 0Z) \Rightarrow (q_1, \mathtt{0111}, 00Z) \Rightarrow (q_1, \mathtt{111}, 000Z) \Rightarrow$$
$$(q_2, \mathtt{11}, 00Z) \Rightarrow (q_2, \mathtt{1}, 0Z) \Rightarrow (q_2, \varepsilon, Z) \Rightarrow (q_3, \varepsilon, Z)$$

Tehát ezt a szót az automata elfogadja.

Nézzük meg általában, hogyan működik és miért is jó ez a veremautomata! Az első átmenet szerint, amennyiben az első karakter `0`*, a verem alját jelölő Z tetejére rakunk egy 0 karaktert és átkerülünk a q_1 állapotba. Mivel a q_0 állapothoz nincs olyan átmenetünk, ami az* `1` *karaktert kezelné, ezért ha* `1`*-gyel kezdődik a szó, akkor a számítás elakad, és így definíció szerint nem lesz elfogadó. A q_1 állapotban minden további* `0` *karakternél berakunk egy 0-t a verembe. Az első*

`1` *érkezésekor vége ennek a fázisnak, átlépünk* q_2*-be. Innentől amíg* `1` *karakterek jönnek, addig kiveszünk egy-egy 0-t a veremből. Ha ilyenkor megint egy* `0` *következik, akkor a számítás elakad, ezért nem fogadja el a szót.*

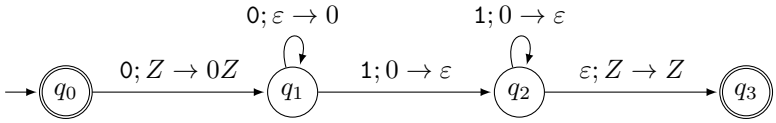
Látszik, hogy egy $\mathtt{0}^n\mathtt{1}^n$ *alakú szón, ha* $n \geq 1$*, az utolsó karakter elolvasása után a PDA a* q_2 *állapotban lesz, és a veremben ilyenkor csak a* Z *szimbólum marad, tehát az utolsó átmenetet használva eljut a* q_3 *elfogadó állapotba. Az* $n = 0$ *esetben a nulla hosszú számítás elfogadó, mert* $q_0 \in F$*.*

Egy 0^n1^k *alakú szó esetén, amikor* $n > k \geq 0$*, akkor a* q_2 *állapotban nem lehet az összes* 0 *karaktert törölni a veremből, nincs mód átkerülni a* q_3 *állapotba. Ha viszont* $n < k$*, akkor, mivel a veremből a* 0 *szimbólumok elfogynak, a szó vége előtt átkerülünk a* q_3 *állapotba, ahol a számítás elakad, és ezért nem lesz elfogadó.*

Azt már láttuk, hogy ha a szó nem megfelelő alakú, azaz az első karaktere `1`*, vagy ugyan* `0` *karakterrel kezdődik, de néhány* `1` *karakter után megint* `0` *jön, akkor a számítása elakad, tehát nem fogadja el a szót.*

2. Megjegyzés. *A* $\delta(q_0, \mathtt{0}, Z) = \{(q_1, 0Z)\}$ *átmenet helyett használhattuk volna a* $\delta(q_0, \mathtt{0}, \varepsilon) = \{(q_1, 0)\}$ *átmenetet is, és hasonlóan a* $\delta(q_1, \mathtt{0}, \varepsilon) = \{(q_1, 0)\}$ *helyett a* $\delta(q_1, \mathtt{0}, 0) = \{(q_1, 00)\}$ *átmenetet.*

A veremautomatákat is lehet gráffal ábrázolni, a nyilakon a szó megfelelő karakterétől elválasztva jelezzük a veremben történő változást. Az előző példa ebben a formában



2. Veremautomaták és CF nyelvtanok

Nem nehéz megmutatni, hogy a környezetfüggetlen nyelvtanok felismerhetők veremautomatával. Ennél lényegesen bonyolultabb a másik irány, hogyan lehet egy veremautomatából környezetfüggetlen nyelvtant készíteni.

1. Tétel. *Minden* L *környezetfüggetlen nyelvhez van olyan* M *veremautomata, melyre* $L(M) = L$*.*

Bizonyítás vázlat: Legyen $G = (V, \Sigma, S, P)$ egy környezetfüggetlen nyelvtan ami generálja az L nyelvet.

Az az ötlet, hogy a veremben készítjük a levezetést és amikor ezzel sikerül előállítani a bemeneti szó következő karakterét, akkor ezt a karaktert kivesszük a veremből és a szón is továbblépünk a következő karakterre. Ehhez legyen $\Gamma = V \cup \Sigma \cup \{Z\}$, ahol $Z_0 = Z$ és $Q = \{q_0, q, q_e\}$, q_e az elfogadó állapot.

Először a nyelvtan kezdő változóját berakjuk a verembe: $\delta(q_0, \varepsilon, Z) = \{(q, SZ)\}$.

A q állapotban, ha a verem tetején a nyelvtan egy A változója van, akkor helyettesítsük ezt egy A-hoz tartozó szabály jobb oldalával, azaz $\delta(q, \varepsilon, A)$ az olyan (q, α) párokból áll, melyekre a nyelvtanban van $A \to \alpha$ szabály.

Amikor a q állapotban a verem tetején egy $a \in \Sigma$ karakter áll és ez megegyezik a szó következő karakterével, akkor kidobjuk a veremből. Amennyiben nem a megfelelő karakter jelenik meg a veremben, akkor a számítás elakad, $\delta(q, a, a) = \{(q, \varepsilon)\}$.

Az elfogadáshoz kell még egy $\delta(q, \varepsilon, Z) = \{(q_e, Z)\}$ átmenet, amivel átlépünk az elfogadó állapotba.

Teljes indukcióval belátható, hogy ez a konstrukció jó. □

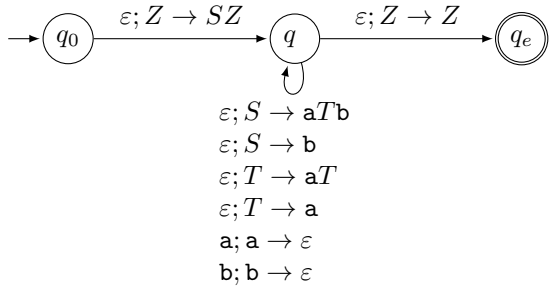
Vegyük észre, hogy a konstrukció erősen kihasználja a veremautomata nemdeterminizmusát.

2. Példa. *Legyen a nyelvtan* $S \to \mathtt{a}T\mathtt{b} \mid \mathtt{b} \qquad T \to \mathtt{a}T \mid \mathtt{a}$

Ehhez az előbb vázolt PDA átmeneti szabályai:

$$\begin{aligned}
\delta(q_0, \varepsilon, Z) &= \{(q, SZ)\} \\
\delta(q, \varepsilon, S) &= \{(q, \mathtt{a}T\mathtt{b}),\ (q, \mathtt{b})\} \\
\delta(q, \varepsilon, T) &= \{(q, \mathtt{a}T),\ (q, \mathtt{a})\} \\
\delta(q, \mathtt{a}, \mathtt{a}) &= \{(q, \varepsilon)\} \\
\delta(q, \mathtt{b}, \mathtt{b}) &= \{(q, \varepsilon)\} \\
\delta(q, \varepsilon, Z) &= \{(q_e, Z)\},
\end{aligned}$$

ahol q_0 *a kezdő és* q_e *az elfogadó állapot. Ez könnyen fel is rajzolható:*



Az `aaaab` *szón egy lehetséges számítás:*

$$(q_0, \mathtt{aaaab}, Z) \Rightarrow (q, \mathtt{aaaab}, SZ) \Rightarrow (q, \underline{\mathtt{a}}\mathtt{aaab}, \underline{\mathtt{a}}T\mathtt{b}Z) \Rightarrow (q, \mathtt{aaab}, T\mathtt{b}Z) \Rightarrow$$
$$(q, \underline{\mathtt{a}}\mathtt{aab}, \underline{\mathtt{a}}T\mathtt{b}Z) \Rightarrow (q, \mathtt{aab}, T\mathtt{b}Z) \Rightarrow (q, \underline{\mathtt{a}}\mathtt{ab}, \underline{\mathtt{a}}T\mathtt{b}Z) \Rightarrow (q, \mathtt{ab}, T\mathtt{b}Z) \Rightarrow$$
$$(q, \underline{\mathtt{a}}\mathtt{b}, \underline{\mathtt{a}}\mathtt{b}Z) \Rightarrow (q, \underline{\mathtt{b}}, \underline{\mathtt{b}}Z) \Rightarrow (q, \varepsilon, Z) \Rightarrow (q_e, \varepsilon, Z)$$

A konfiguráció-sorozatban az aláhúzott karakter jelzi, amikor a szó soron következő karaktere megegyezik a verem tetején levővel. A többi esetben a verem tetején levő változót helyettesítjük egy megfelelő nyelvtani szabály alapján, lényegében egy helyes levezetést szimulálva. Az is látszik, hogy ha ilyenkor másik szabályt választottunk volna, akkor a számítás nem jutna el sikeresen az elfogadó állapothoz.

2. Tétel. *Ha az L nyelvhez van ezt elfogadó veremautomata, akkor az L nyelv környezetfüggetlen.*

Ennek bizonyítása eléggé összetett, ezt itt nem is vázoljuk.

3. Megjegyzés. *Mivel egy véges automata felfogható veremautomatának is (a verembe soha nem ír és nem is olvas onnan), a reguláris nyelvekhez is van veremautomata, tehát van környezetfüggetlen nyelvtan is.*

3. Tétel. *Az $\{\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n : n \geq 0\}$ nyelv nem környezetfüggetlen.*

Ezt itt nem bizonyítjuk, de intuícióként azt lehet látni, hogy egy veremautomata ugyan el tudja tárolni az `a` betűket, de annak ellenőrzése során, hogy `b` betűből ugyanannyi van, ezek kikerülnek a veremből és akkor már nincs mihez viszonyítani a `c` betűk számát. (Ez persze nagyon messze van egy bizonyítástól!)

Míg a palindromok nyelve környezetfüggetlen (hogy működhet egy jó veremautomata?), megint csak „érezhető", hogy az $L = \{ww : w \in \Sigma^*\}$ nyelv nem az. A problémát itt az okozza, hogy a veremből (LIFO) a berakáshoz képest fordított sorrendben érjük el a szimbólumokat, a palindromok esetében pont ez kell, ennél az L nyelvnél pedig az eredeti és nem a fordított sorrendben lenne szükségünk a szó első felére.

3. Determinisztikus veremautomata

Természetesen lehet a veremautomatának determinisztikus változatát is definiálni, és az alkalmazások szempontjából ez a hasznosabb változat.

A determinisztikus veremautomatában minden esetben legfeljebb egy lépési lehetőség van. (Ez a véges automatáknál használt terminológia szerint igazából a hiányos automata.) Az ε-átmeneteket ilyenkor is megtartjuk, csak ezek sem kerülhetnek konfliktusba a ténylegesen olvasó (író) lépésekkel, tehát pl. ha van $\delta(q, \varepsilon, A)$ átmenet, akkor ugyanerre a q állapotra és A veremtetőre egyetlen $a \in \Sigma$ karakterre sem lehet a $\delta(q, a, A)$ átmenet is definiálva.

Az első példa veremautomatája egy determinisztikus veremautomata, de az a PDA, amit CF nyelvtanból definiáltunk, nem determinisztikus.

A következő eredmény indokolja, hogy veremautomatának a nemdeterminisztikus változatot hívjuk.

4. Tétel. *Van olyan L nyelv, amelyhez létezik M veremautomata, hogy $L = L(M)$, de nem létezik ilyen determinisztikus veremautomata.*

A palindromok nyelve például ilyen, de ennek a bizonyítsa nem túl egyszerű. Ezért helyette egy olyan nyelvet tekintünk, amire a bizonyítás ötletes, de nem túl bonyolult.

5. Tétel. *Az $L = \{\mathtt{a}^n\mathtt{b}^n : n \geq 0\} \cup \{\mathtt{a}^n\mathtt{b}^{2n} : n \geq 0\}$ nyelv környezetfüggetlen, de nincs hozzá determinisztikus veremautomata.*

Bizonyítás vázlat: A nemdeterminizmust kihasználva nem nehéz az L nyelvhez veremautomatát készíteni, de pl. az

$$S \to X \mid Y \qquad X \to \mathtt{a}X\mathtt{b} \mid \varepsilon \qquad Y \to \mathtt{a}Y\mathtt{bb} \mid \varepsilon$$

nyelvtan is mutatja, hogy L környezetfüggetlen.

Tegyük most fel, hogy van hozzá determinisztikus veremautomata. Vegyünk ebből két példányt, legyenek ezek M_1 és M_2. Definiáljuk a következő veremautomatát: az állapotok a két automata állapotai lesznek, a kezdő állapot az M_1 kezdő állapota, az elfogadók M_2 elfogadó állapotai. Az átmeneteket változtassuk meg úgy, hogy M_1 eredeti elfogadó állapotaiból az összes átmenet ne M_1-be, hanem M_2 megfelelő állapotaiba mutasson. Továbbá ezeknél és az M_2-beli átmeneteknél ahol a bemeneti ábécéből a `b` betű szerepelt, azt cseréljük le `c` betűre (az `a` betű marad).

Nem nehéz látni, hogy ez az automata elfogadja az $\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n$ alakú szavakat, hiszen az $\mathtt{a}^n\mathtt{b}^n$ rész után egy olyan állapotba kell érjen, ami az eredeti M_1 determinisztikus veremautomatának elfogadó állapota volt. Innen a konstrukció miatt átkerülünk az M_2 részbe, ami elfogadja, ha $\mathtt{c}^n$ jön, hiszen M_1 elfogadná, ha $\mathtt{b}^n$ lenne a folytatás. Másrészt meggondolható, hogy semmi más szót nem fogad el az új PDA, azaz megadtunk egy veremautomatát az $\{\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n : n \geq 0\}$ nyelvhez, ami pedig nem környezetfüggetlen, azaz veremautomata sincs rá. Ezzel tehát ellentmondásra jutottunk, nem igaz a feltevés, hogy L-hez van determinisztikus veremautomata.

□

4. Alkalmazás

Bár a determinisztikus veremautomaták az alkalmazások szempontjából nagyon fontosak, látjuk, hogy ezek a környezetfüggetlen nyelveknek csak egy valódi részhalmazát képesek felismerni. Ezért általában nem elég környezetfüggetlen nyelvtant használni, célszerű egy olyat választani, amihez van determinisztikus veremautomata. Annak érdekében, hogy egy szó levezetési fája egyszerűbben generálható legyen, további megkötéseket is szokás alkalmazni.

Az eddigiek alkalmazásának egy fontos területe a fordítóprogramok készítése. Egy fordítóprogram több fő részre osztható, bár ezek nem mindig különülnek el egymástól.

A *lexikai elemzés (tokenizálás)* során megtörténik pl. a programozási nyelv kulcsszavainak felismerése (mintaillesztés), a változók, számok, műveleti jelek, stb. azonosítása. Ezek a feladatok, mint láttuk, véges automatákkal elvégezhetők.

A *szintaktikai elemzés* (szintaxis elemzés) feladata a struktúrák, zárójelezések (begin – end párok), aritmetikai, logikai kifejezések, stb. felépítése. Ehhez használható determinisztikus veremautomata, aminek segítségével lényegében egy levezetési fa (szintaxisfa) készülhet. Egy nyelv megengedhet CF-ből kivezető lehetőségeket is, de ezeket külön kell kezelni. (Ilyen lehet például a

függvényhívás, amikor minden alkalommal ugyanannyi, ugyanolyan típusú paraméterrel kell hívni a függvényt.)

Ez után következhet a *szemantikai elemzés* pl. a típusok ellenőrzése, majd az optimalizálás és a kód generálása.

. . . . . .