

Programozás módszertan

Nyelvek, formális nyelvek

Pere László (pipas@linux.pte.hu)

PÉCSI TUDOMÁNYEGYETEM TERMÉSZETTUDOMÁNYI KAR
INFORMATIKA ÉS ÁLTALÁNOS TECHNIKA TANSZÉK

PÉCSI TUDOMÁNYEGYETEM TERMÉSZETTUDOMÁNYI KAR
SZÁMÍTÁSTECHNIKAI SZOLGÁLTATÓ KÖZPONT

MAGYAR TUDOMÁNYOS AKADÉMIA
SZÁMÍTÁSTECHNIKAI ÉS AUTOMATIZÁLÁSI KUTATÓINTÉZET
ELEARNING OSZTÁLY

Nyelvek

A nyelv

A nyelv az emberi elme legfejlettebb terméke, amely az értelemmel, intelligenciával együtt fejlődött ki.

Az intelligencia és a nyelv nagy valószínűség szerint nem választhatók el egymástól, a nyelv nem csak a terméke, hanem része is az intelligenciának.

A nyelv a kapcsolattartás (kommunikáció) legfejlettebb formája, amely végtelen rugalmassággal képes kifejezni gondolatainkat.

A nyelv

„Mostantól kezdve *nyelv*nek tekintem a mondatok valamely (véges vagy végtelen) halmazát; minden egyes mondat véges hosszúságú, és elemek véges halmazából épül fel.”

NOAM CHOMSKY: *Mondattani szerkezetek*,
Osiris-Századvég (1995), ISBN: 963 379 0441

Gépi nyelvek

Mivel a nyelv a kapcsolattartás legfejlettebb formája, nyilvánvalóan nem mondhatunk le a használatáról, amikor ember/gép kapcsolattartást (*human-computer interaction*) hozunk létre.

A számítógéppel való kapcsolattartásra használt *gépi nyelvek* legfontosabb tulajdonsága, hogy értelmezésük egyértelmű.

Az egyértelműség definíciója korántsem egyszerű, nyilvánvalóan nem csak arról van szó, hogy a szavaknak egyetlen jelentése van.

„The old man thinks he is in love with his daughter.”

Formális nyelvek

Noam Chomsky munkássága nyomán képesek vagyunk matematikai, formális eszközökkel kezelni a nyelveket. A formális eszközökkel definiált nyelveket formális nyelveknek nevezzük.

A dolog pikantériája az, hogy a matematika is egyfajta nyelvet használ, a „formális” szó jelentése tehát körülbelül „nyelvi eszközökkel kezelt”.

A matematikai nyelvészet, a formális nyelvek elmélete a számítástechnika legalapvetőbb alapköve.

Programozási nyelvek

„A program egy komputáció specifikációja. A programozási nyelv a program denotációjának eszköze.”

MACLENNAN, B.: *Principles of Programming Languages*, Holt-Saunders, 1987.

A programozási nyelvek tehát gépi nyelvek, amelyek komputáció leírására készültek.

Programozási nyelvek

A program a komputációt írja le, de ez nem jelenti feltétlenül azt, hogy megadja az algoritmust, azt, hogy a komputációt milyen elemi lépésekben kell elvégezni (bár a legtöbb esetben erről van szó).

Léteznek olyan programozási nyelvek (pl. Prolog), amelyek értelmezőprogramjába az algoritmus be van építve, de a program ezekben az esetekben is meghatározza a komputáció folyamatát.

A formális nyelvek elmélete

A nyelvelemélet

A formális nyelvek elmélete külön tananyag, amelynek csak a gyakorlat szempontjából elengedhetetlenül fontos alapelemeit vizsgáljuk meg.

Ezek az alapok lehetővé teszik számunkra, hogy megértsük a programozási nyelvek leírásának eszközeit, megismerkedhessünk a programok kezelésének alapkérdéseivel.

A nyelv

Definíció: Legyen Σ egy ábc (véges jelkészlet halmaz). Jelölje Σ^* az ábc-ből alkotott véges, de nem korlátos sorozatok halmazát. Akkor

$$\mathcal{L} \subset \Sigma^*$$

halmazt nyelvnek nevezzük.

A nyelv tehát az adott ábc-ből képezhető sorozatok egy halmaza. Ez tulajdonképpen megegyezik a már említett definícióval.

(Megjegyezzük, hogy a sorozat hossza lehet 0 is, azaz $\lambda \in \Sigma^*$.)

A nyelv (példa)

Legyen

$$\Sigma = \{a, b\}$$

akkor

$$\Sigma^* = \{\lambda, a, b, aa, ab, ba, bb, aaa, \dots\}$$

Legyen \mathcal{L}_p nyelv a következő:

$$\mathcal{L}_p \subset \Sigma^* = \{a^n b^n \mid n = 1, 2, \dots\}$$

A nyelvtan

Definíció: A

$$G = (N, \Sigma, P, S)$$

négyest nyelvtannak (grammatika, generatív grammatika) nevezzük, ahol:

N *A nemterminálisok (segédváltozók) halmaza.*

Σ *Az nyelv ábc-je ($N \cap \Sigma = \emptyset$).*

P *$\alpha \rightarrow \beta$ alakú helyettesítési szabályok halmaza,
ahol $\alpha \in (N \cup \Sigma)^* N (N \cup \Sigma)^*$ és $\beta \in (N \cup \Sigma)^*$*

S *Startszimbólum (mondatszimbólum), amelyre
 $S \in N$.*

Levezetés

Definíció: Legyen $G = (N, \Sigma, P, S)$ egy nyelvtan és legyen $\gamma \in (N \cup \Sigma)^*$, valamint $\delta \in (N \cup \Sigma)^*$.

Azt mondjuk, hogy a G nyelvtan alapján a γ jelsorozatból a δ jelsorozat levezethető és azt írjuk, hogy

$$\gamma \vdash_G^* \delta$$

ha P beli behelyettesítési szabályok alkalmazásával γ átalakítható δ formára.

Levezetés (példa)

Legyen $G = (N, \{a, b\}, P, S)$ egy nyelvtan, ahol $N = \{S, A, B\}$, valamint P :

$$S \rightarrow AB \quad (1)$$

$$A \rightarrow abB \quad (2)$$

$$B \rightarrow baA \quad (3)$$

$$A \rightarrow \lambda \quad (4)$$

Akkor $A \vdash_G^* abba$, mert:

$$A \vdash_G^2 abB \vdash_G^3 abbaA \vdash_G^4 abba$$

Megjegyzés

Vegyük észre, hogy az eddig elmondottak fényében a matematika nem más, mint behelyettesítési szabályok egy halmaza, amelynek segítségével egyik kifejezés a másiktól levezethető.

Ilyen megközelítésben beszélhetünk a matematika nyelvéről, amely a formális nyelvelmélet eszköztárával kezelhető.

A nyelv és a nyelvtan

Tétel: Minden $G = (N, \Sigma, P, S)$ nyelvtan meghatároz egy $\mathcal{L}_G \subset \Sigma^*$ nyelvet, amelyre

$$\mathcal{L}_G = \{w \in \Sigma^* \mid S \vdash_G^* w\}$$

A nyelv azon Σ^* beli jelsorozatok halmaza, amelyek az adott nyelvtan behelyettesítési szabályai alapján levezethetők S startszimbólumból.

Nemterminálisok: a szabályok alkalmazását nem fejezhetjük be amíg a kifejezésben nemterminális van, mert $N \cap \Sigma = \emptyset$ és $w \in \Sigma^*$.

Nyelv (példa)

Legyen P szabályhalmaz G -ben:

$$S \rightarrow aSb \quad (5)$$

$$S \rightarrow \lambda \quad (6)$$

Akkor

$$\mathcal{L}_G = \{a^n b^n \mid n = 0, 1, 2, \dots\}$$

Chomsky nyelvtanosztályok

NOAM CHOMSKY munkája kiindulópontjaként a nyelvtanokat osztályozta a levezetési szabályokra bevezetett szigorításokkal.

Nyilvánvaló, hogy minél komolyabb szigorításokat vezetünk be a használható behelyettesítési szabályokra, annál egyszerűbb a nyelvtan (egyszerűbb kezelni).

Egy nyelvet többféle nyelvtan segítségével is megadhatunk, de a nyelveket osztályozni lehet az alapján, hogy mennyire egyszerű a leírásra használható *legegyszerűbb* nyelvtan.

Ilyen okokból általában nem nyelvtanosztályokról, hanem Chomsky nyelvosztályokról beszélünk!

Chomsky nyelvosztályok

Legyen $G = (N, \Sigma, P, S)$ és legyen $A, B \in N$, valamint $a \in \Sigma$ és $\alpha, \beta, \gamma \in (N \cup \Sigma)^$.*

\mathcal{L}_0 A szabályokra nem vezetünk be újabb korlátozásokat.

\mathcal{L}_1 Környezetfüggő nyelvek, ahol P minden szabálya $\beta A \gamma \rightarrow \beta \alpha \gamma$ alakú.

\mathcal{L}_2 Környezetfüggetlen nyelvek, ahol P minden szabálya $A \rightarrow \alpha$ alakú.

\mathcal{L}_3 Reguláris (jobbreguláris) nyelvek, ahol P minden szabálya $A \rightarrow a$ vagy $A \rightarrow aB$ alakú.

Chomsky nyelvosztályok

Tétel:

$$\mathcal{L}_3 \subset \mathcal{L}_2 \subset \mathcal{L}_1 \subset \mathcal{L}_0$$

Ez egy különösen fontos és szép tétel, amelynek a bizonyítása több hónapig tartó előtanulmányokat igényel.

A programozási nyelvek

A legtöbb programozási nyelv a reguláris vagy a környezetfüggetlen nyelvek osztályába tartozik (azaz a legegyszerűbben kezelhető nyelvtanaik a reguláris vagy a környezetfüggetlen nyelvtanok közé tartoznak).

Úgy is fogalmazhatnánk, hogy a reguláris és a környezetfüggetlen nyelvtanok kifejezőereje általában elegendő programozási nyelv készítéséhez.

Példa

A programozási nyelveket gyakran egy lehetséges nyelvtanuk alapján adjuk meg. A következő példa egy ilyen nyelvtan részletét mutatja be:

feltételes:

```
if (feltétel) {  
    utasításlista  
}
```

feltétel: *szám reláció szám*

reláció: == | != | < | > | <= | >=

Automaták

A tartalmazás problémája

A gépi nyelvek kezelésekor az egyik alapvető probléma a tartalmazás problémája, annak eldöntése, hogy egy adott mondat az adott nyelvtan által meghatározott nyelv része-e.

Ha a mondat (kifejezés) a nyelv része, azaz levezethető az adott nyelvtan alapján, azt mondjuk, hogy a kifejezés nyelvtanilag helyes (jól formált kifejezés, szintaktikailag helyes kifejezés).

A számítástechnika szempontjából fontos, hogy olyan géppel is végrehajtható módszereket adjunk, amelyek alapján a \mathcal{L}_3 és \mathcal{L}_2 nyelvek esetében eldönthető a tartalmazás kérdése.

Automaták

A tartalmazás kérdésének eldöntésére adott módszereket általában egyszerű automaták formájában adjuk meg.

Ezeknek az automatáknak a működése könnyedén szimulálható bármely számítógépen a megfelelő programok segítségével.

A végesautomata

Definíció: Az $M(Q, \Sigma, \delta, q_0, F)$ ötöst végesautomatának nevezzük, ahol:

Q az automata állapotainak véges halmaza,

Σ az elemzendő nyelv jelkészlete,

δ mozgási szabályok véges halmaza $Q \times \Sigma \rightarrow Q$ alakban,

q_0 a kezdőállapot,

F a végállapotok (elfogadó állapotok) halmaza, hogy $F \subseteq Q$.

A végesautomata működése

A végesautomata működése:

1. Kezdetben az automata q_0 állapotban van, a mondat első szimbólumát olvassa.
2. Ha az automata elolvasta az utolsó szimbólumot is, leáll.
3. Mozgás:
 - a) ha létezik $\delta(q_n, a) = q_m$ a jelenlegi állapotra és az olvasott szimbólumra vonatkozó mozgási szabály, az automata felveszi a q_m állapotot és egy szimbólummal továbblép,
 - b) ha nem létezik ilyen szabály, az automata leáll
4. A következő lépés a 2.

Egyértelműség

Az automata működésének leírásakor feltételeztük, hogy legfeljebb egy $\delta(q_n, a) = q_m$ mozgási szabály van adott $q_n \in Q, a \in \Sigma$ értékekre.

Az ilyen végesautomatákat determinisztikus végesautomatáknak nevezzük.

A gyakorlatban nem jelent hátrányt ez a megszorítás, viszont az ilyen automatákat nyilvánvalóan egyszerűbb kezelni mint a nem determinisztikus társaikat.

Felismerés automatával

Definíció: Az $M(Q, \Sigma, \delta, q_0, F)$ determinisztikus végesautomata a mondatot felismeri, ha azt végigolvassa és az utolsó szimbólum olvasása után $q_n \in F$ elfogadó végállapotba kerül.

Tétel: A determinisztikus automatával felismerhető nyelvek osztálya az \mathcal{L}_3 , azaz minden reguláris nyelvhez kreálhatunk a tartalmazást eldöntő determinisztikus végesautomatát és minden determinisztikus végesautomata által felismert nyelv generálható reguláris nyelvtannal.

Feladat

Feladat: *Konstruáljunk végesautomatát, amely felismeri az $\mathcal{L} = \{a^n b^m \mid n, m = 1, 2, 3, \dots\}$ nyelvet! Törekedjünk arra, hogy a lehető legegyszerűbb megoldást adjuk meg!*

A feladat megoldásához használjunk olyan mozgási szabályokat, amelyek megőrzik az automata állapotát az azonos karakterek olvasásakor.

Vegyük észre, hogy a nyelv végtelen, amelyet végesautomatával csak akkor ismerhetünk fel, ha az automata állapotaiban hurkok vannak.

A veremautomata

Definíció: A $P(Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ hetest veremautomatának nevezzük, ahol

Q az állapotok véges halmaza,

Σ a nyelv elemkészlete,

Γ a veremszimbólumok véges halmaza,

δ a mozgási szabályok véges halmaza

$Q \times (\Sigma \cup \lambda) \times \Gamma \rightarrow Q \times \Gamma^n$ alakban
 $n = 0, 1, \dots k$ -ra,

q_0 az automata kezdőállapota ($q_0 \in Q$),

Z_0 a verem kezdőértéke ($Z_0 \in \Gamma$),

F az elfogadó állapotok halmaza ($F \subset Q$).

A veremautomata működése

A veremautomata működése:

1. Kezdetben az automata q_0 állapotban van, az első jelet olvassa, a verem tartalma pedig Z_0 .
2. Ha az automata az utolsó jelet is elolvasta, leáll.
3. Ha $q_n, q_m \in Q$, $a \in (\Sigma \cup \lambda)$, $Z_n \in \Gamma$ jelenlegi állapotra $\delta(q_n, a, Z_n) = q_m, \Gamma^n$ alakban
 - a) létezik szabály, az automata elolvassa a szimbólumot, eltávolítja Z_n elemet a verem tetejéről, elhelyezi a vermen Γ^n szimbólumokat és felveszi q_m állapotot
 - b) ha nem létezik szabály, az automata leáll.
4. A következő lépés a 2.

Felismerés veremautomatával

A következő két definíció bizonyos értelemben egyenértékű:

Definíció: *A veremautomata egy jelsorozatot a nyelv részeként elfogad, ha a jelsorozatot végigolvassa és utána $q_n \in F$ elfogadó állapotba kerül λ mozgásokkal vagy azok nélkül.*

Definíció: *A veremautomata egy jelsorozatot a nyelv részeként elfogad, ha a jelsorozatot végigolvassa és utána üres – csak a Z_0 szimbólumot tartalmazó – veremállapotot ér el λ mozgásokkal vagy azok nélkül.*

Amint látjuk a definíciók megengedik, hogy az automata a jelsorozat elolvasása után λ mozgást vagy mozgásokat végezzen.

Felismerés veremautomatával

Tétel: *A veremautomatával felismerhető nyelvek osztálya az \mathcal{L}_2 , azaz minden környezetfüggetlen nyelvhez kreálható a tartalmazást eldöntő veremautomata és minden veremautomata által felismert nyelv leírható környezetfüggetlen nyelvtannal.*

Feladat

Feladat: *Konstruáljunk veremautomatát, amely felismeri a $\mathcal{L} = \{a^n b^n \mid n = 0, 1, 2, \dots\}$ nyelvet üres veremmel!*

Ez a nyelv végesautomatával nem ismerhető fel, mert az állapotok halmaza véges. Veremautomatával azonban felismerhető a nyelv, mivel a verem befogadóképessége véges ugyan, de nem korlátos.

Az ismeretek hasznosítása

Fordítóprogram

Feladat: *Készítsünk fordítóprogramot, amely a " karakterek közt található szöveget zöldre színezi!*

A program az olvasott karaktereket küldje a kimenetre. A zöld szín bekapcsolására használjuk a `` kifejezést, az eredeti szín visszaállításához pedig a `` kifejezést, ahogyan azt a HTML nyelvben megszoktuk!

Nyilvánvaló, hogy legkönnyebben egy kétállapotú végesautomatával oldhatjuk meg a feladatot. Az automata állapota jelzi, hogy páros vagy páratlan " karaktert olvasunk.

Oszlopok számlálása

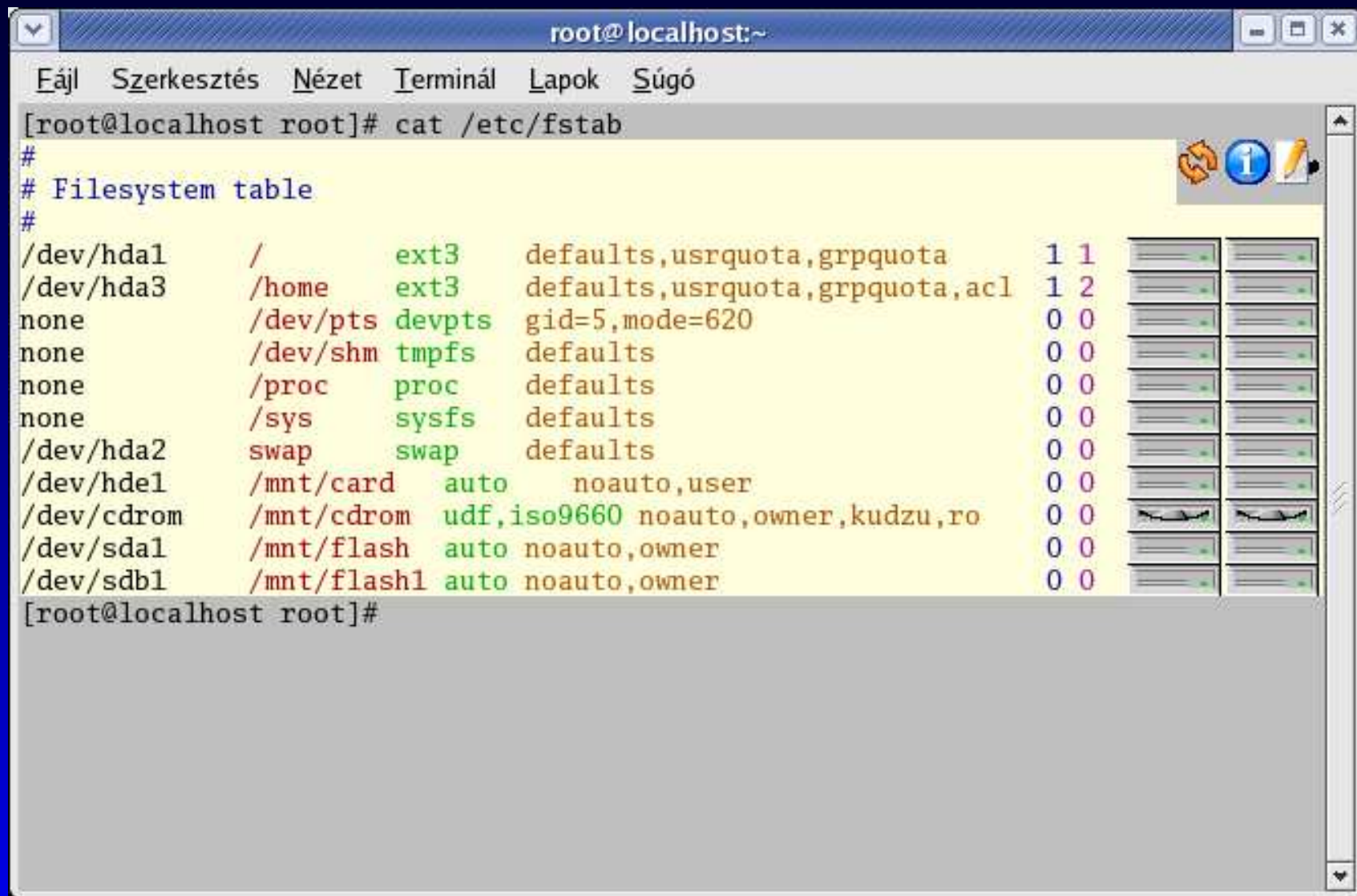
Feladat: *Készítsünk programot, amely a szöveg minden karakteréről eldönti, hogy az hanyadik oszlopban van! Az oszlopokat egy vagy több szóköz vagy tabulátor karakter választja el egymástól.*

Elegendő egyszer végigolvasni a szöveget és minden esetben növelni az oszlopszámlálót, amikor normál karakter után oszlopelválasztó jelet olvasunk.

Oszlopok számlálása

```
n = 0; i = 0;
for (q = 0; q < strlen(line); ++q) {
    if (line[q] == ' ' || line[q] == '\t') {
        if (i == 0) {
            n++;
            i = 1;
        }
    } else {
        i = 0;
    }
    _vte_set_color(terminal, q, c_y, n, -1);
}
```


Oszlopok számlálása



```
root@localhost:~
Fájl Szerkesztés Nézet Terminál Lapok Súgó
[root@localhost root]# cat /etc/fstab
#
# Filesystem table
#
/dev/hda1      /            ext3      defaults,usrquota,grpquota    1 1
/dev/hda3      /home        ext3      defaults,usrquota,grpquota,acl 1 2
none          /dev/pts     devpts    gid=5,mode=620                0 0
none          /dev/shm     tmpfs     defaults                      0 0
none          /proc        proc      defaults                      0 0
none          /sys         sysfs     defaults                      0 0
/dev/hda2      swap         swap      defaults                      0 0
/dev/hde1      /mnt/card    auto      noauto,user                   0 0
/dev/cdrom     /mnt/cdrom   udf,iso9660 noauto,owner,kudzu,ro        0 0
/dev/sda1      /mnt/flash   auto      noauto,owner                   0 0
/dev/sdb1      /mnt/flash1  auto      noauto,owner                   0 0
[root@localhost root]#
```

Shift-reduce automata

Feladat: *Készítsünk számológépet a következő nyelvtan alapján:*

<i>szám</i> :	0		1		2		3		4	
		5		6		7		8		9
		(<i>szám</i>)						
		<i>szám</i>	+	<i>szám</i>						
		<i>szám</i>	-	<i>szám</i>						
		<i>szám</i>	*	<i>szám</i>						
		<i>szám</i>	/	<i>szám</i>						
		;								

Shift-reduce automata

Az automata működése:

1. Megpróbáljuk illeszteni a verem tetején található elemekre valamelyik szabály jobb oldalát.
 - a)* Ha sikerül, behelyettesítjük a bal oldallal és a következő lépés az 1.
 - b)* Ha nem, tovább.
2. Megkísérlünk újabb karaktert olvasni.
 - a)* Ha sikerült, a következő lépés az 1.
 - b)* Ha vége a kifejezésnek, tovább.
3. Ha a veremben egyetlen elem van, az az eredmény, ha több, a kifejezés nyelvtanilag hibás.

Shift-reduce automata

```
# ./a.out '(3-1)/2'
push('(') Stack: '('
push('3') Stack: '(3'
push('-') Stack: '(3-'
push('1') Stack: '(3-1'
reduce x-y Stack: '(2'
push(')') Stack: '(2)'
reduce (x) Stack: '2'
push('/') Stack: '2/'
push('2') Stack: '2/2'
reduce x/y Stack: '1'
Value: 1
#
```

Shift-reduce automata

```
# ./a.out '(3-1(/2'
push('(')    Stack: '('
push('3')    Stack: '(3'
push('-')    Stack: '(3-'
push('1')    Stack: '(3-1'
reduce x-y   Stack: '(2'
push('(')    Stack: '(2('
push('/')    Stack: '(2(/'
push('2')    Stack: '(2(/2'
Syntax error
#
```

Irodalomjegyzék

Irodalomjegyzék

- NOAM CHOMSKY: *Mondattani szerkezetek*
Nyelv és elme, Osiris-Századvég (1995), ISBN:
963 379 0441
- Bach Iván: *Formális Nyelvek*, Typotex (2001),
ISBN: 963-9132-92-6
- A *bison* program dokumentációja (`info`
`bison`)