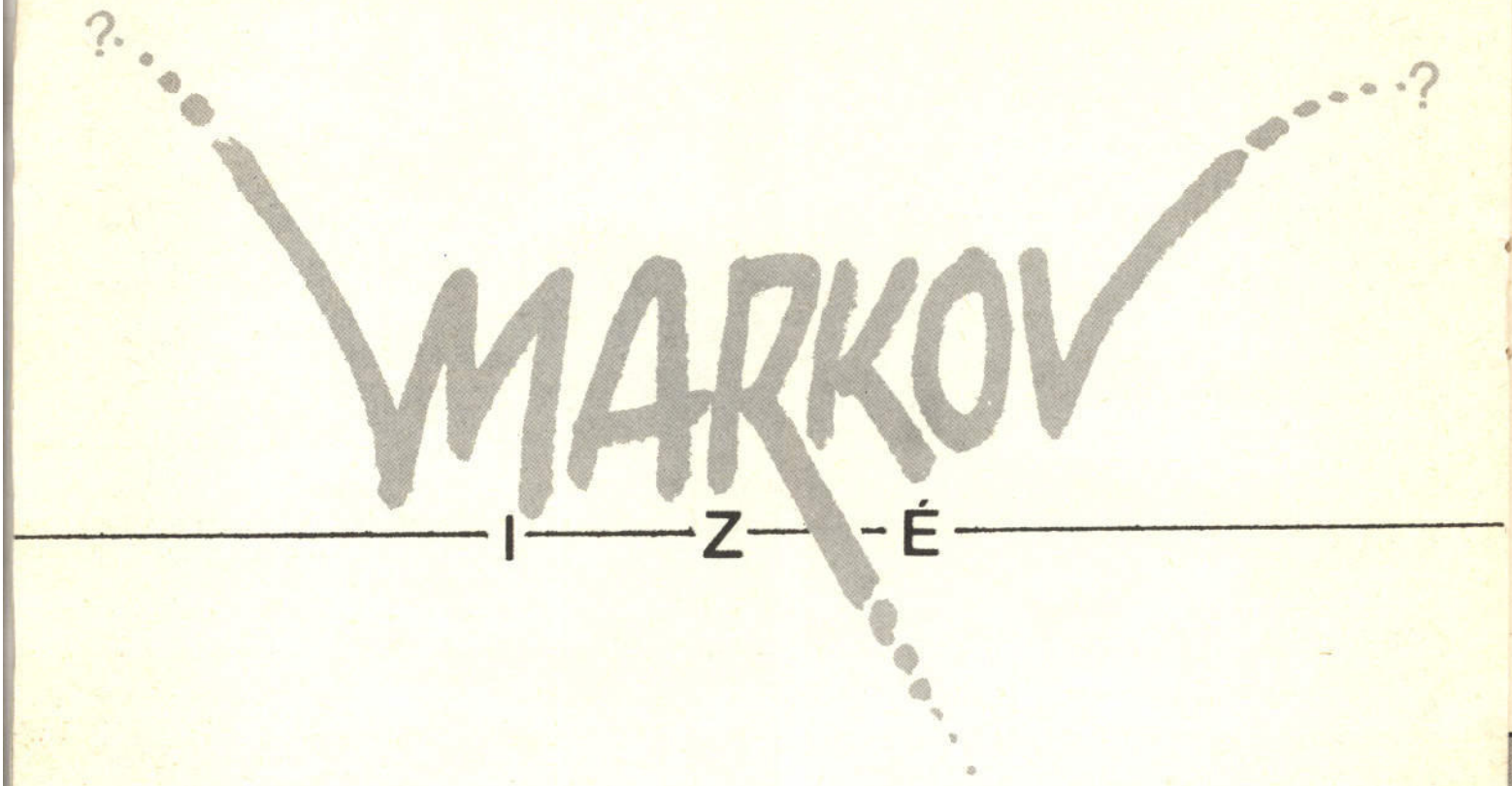

Ajánlás:

Amikor kezembe vettem a cikket, kicsit boszszús voltam: matematikus öcsémnek megint „elsült" az agya, pedig én egy szórakoztató programot kértem a Bit-let számára. De aztán amikor másodszor is kihűlt a vacsora, mert képtelen voltam felállni a gép elől, úgy döntöttem, mégiscsak jó lesz ez. (Ehhez hozzátartozik, hogy számítógépes játékkal eddig még fél óránál többet nem sikerült eltöltenem.) Tehát senki ne riadjon vissza a definícióktól.

Lukács József

Vágjunk rögtön a közepébe, lássuk a definíciókat.

Tekintsünk egy valamilyen jelkészlet (pl. ASCII kód) elemeiből álló véges hosszú sorozatot. Ezt nevezzük *szónak*. Ha e szó elejéről és/vagy végéről elhagyunk tetszőleges jeleket, akkor egy új szót kapunk (esetleg üres szót). Az új szó az előzőnek *részszava* lesz. Ha egy részszót egy másik jelsorozattal helyettesítünk, akkor ismét egy új szót kapunk. Pl. a GIZIKE szó IZI részszavának ÖZE helyettesítésével új szót kapunk: GÖZEKE. Itt IZI → ÖZE a *helyettesítési szabály*.

Most tekintsünk n darab helyettesítési szabályt meghatározott sorrendben:

$X_1 \rightarrow Y_1 \quad X_2 \rightarrow Y_2 \ldots X_i \rightarrow Y_n$

Alkalmazzuk a szabályokat az alábbiak szerint!

Vegyünk egy szót – ez lesz az *Inputszó*. Próbáljuk meg ezen a szón alkalmazni az első helyettesítési szabályt. Ha nem sikerül (X_1 nem része az Inputnak), akkor vegyük a második szabályt, és így tovább. Ha alkalmazható a szabály, akkor végezzük el a helyettesítést, és az új szóval kezdjük elölről az egészet. További kikötés, hogy ha egy szabály alkalmazható, akkor X részszó balról első előfordulását kell helyettesíteni Y-nal.

Ha már egyik szabály sem alkalmazhato, akkor álljunk meg. Az utolsó szó a végeredmény, az *Outputszó*.

Érdemes felhívni a figyelmet, hogy a szabályok sorrendje nagyon lényeges, valamint hogy a szabályok jobb és bal oldala nem feltétlenül egyenlő hosszú. A jobb oldal lehet akár üres is!

Természetesen előfordulhat, hogy sohasem tudunk leállni. Ekkor a szavak végtelen sorozata keletkezik.

Összefoglalva egy olyan eljáráshoz jutottunk, aminek van egy bemenete, és szolgáltat egy kimenetet, vagy végtelen sorozatba kezd – elszáll.

Tulajdonképpen a szokványos számítógépek ugyanezt csinálják, csak más formában van megadva az algoritmus, vagyis a program. Bizonyítható, hogy minden olyan feladatra, ami más számítógépeken megoldható, lehet az előbbiek szerinti „programot" írni, és megfordítva. A *program* itt persze felcserélési szabályok adott sorrendű összessége.

Az itt leírtakat először A. A. Markov definiálta, így róla *Markov-algoritmusnak* nevezték el.

Példák

Az eddigiek alapján talán még nem világos, hogy a valóságban hogyan is lehet egy Markov-féle elven működő számítógépre programot készíteni. Ennek megértésére legegyszerűbb, ha megtekintünk néhány mintaprogramot. Az Input elejét és végét mindig # jelzi. Első példánk 0 1 és 2 karakterekből álló tetszőleges jelsorozatokat rendez sorba úgy, hogy elöl álljanak a nullák, utána az egyesek, majd a kettesek.

A program így néz ki.

1. 10 → 01
2. 20 → 02
3. 21 → 12

A program futásának menete 101210 Inputszó esetén:

1. #011210# 1. #011201# 2. #011021#

1. #010121# 1. #001121# 3. #001112#

Először az 1. szabályt alkalmazzuk a szó elején, majd újra 1-et a sor végén. A következő körben az 1. nem alkalmazható, viszont a 2. igen- a negyedik pozícióban. Az így keletkezett szóban ismét 1.-et lehet alkalmazni. Az ötödik lépésben megint 1., hatodszorra pedig sem 1. sem 2. hanem csak 3. működik. A hetedik próbálkozásra már egyik szabály sem működik az Outputszó tehát 001112 – a számok rendezve vannak.

Ha itt a szabályokat felcseréljük, a végeredmény nem változik meg, viszont a rendezést más sorrendben végzi el a gép, érdemes kipróbálni!

A második példa Inputja tetszőlegesen sok „!". A program megszámolja (tízes számrendszerben), hány ! van az Inputban.

1. 0:→ 1
2. 1:→ 2
3. 2:→ 3
4. 3:→ 4
5. 4:→ 5
6. 5:→ 6
7. 6:→ 7
8. 7:→ 8
9. 8:→ 9
10. 9:→ :0
11. #:→ #1
12. /!→ :/
13. #!→ #1/

Ebben a programban az utasítsok nagy része 1–11-ig a számolás menetét definiálja, míg a 12 egy újabb !-et szüntet meg úgy, hogy egyben elindít egy számolást, végül a 13 indítja el az egész folyamatot. A program futása a mellékelt számítógépes modellben

egyszerűen követhető. Azt viszont érdemes megfigyelni, hogy az újonan bevezetett jeleknek (: és / de lehetne más is) külön jelentésük van. A / választja el a felkiáltójeleket a számlálótól, a : pedig azt jelenti, hogy az előtte álló számot növelni kell. A növelést mint műveletet itt külön definiálni kellett, mert a Markov-automata csak karakterekkel dolgozik, nem tudja, hogy vannak számok, és azoknak sorrendje. Ilyen értelemben a 1–10 utasításokat felfoghatjuk a tízes számrendszer definíciójaként is!

Egyébként itt már az utasítások nem mind cserélhetők fel. Például a 12 nem kerülhet előbbre, viszont a 13 bárhol jó lenne, és 1–11 is tetszőleges sorrendben állhat.

Még egy példa, ahol már csak a programot közöljük. Egy szöveg szavait különválogatja úgy, hogy a fölösleges betűközöket kihagyja, és a szavakat / választja el.

1. #⊥ → #
2. U # →
3. /␣/
4. → /

Ezen kívül persze még sok mindent szellemesen meg lehet oldani Markov-programokkal. Néhány probléma, amit megoldásra javasolunk.

bináris összeadó Inputja pl. 1101001+1010
binárisból decimálisba oda-vissza fordító
BCD összeadó, szorzó stb.
és még ami kinek-kinek eszébe jut.

Persze bonyolultabb programok írásához érdemes az algoritmus definícióját kiterjeszteni. Pl. egy jolly-joker (legyen a jele *) bevezetése igen hasznos lenne. Ezt úgy kell érteni, hogy a helyettesítési szabályban a * bármilyen karaktert jelenthet. Az algoritmus további bővítésének lehetőségeit, és azoknak az interpreter programba való beillesztését szintén az olvasóra bízzuk!

Az interpreter

A Markov-programok megírásához és lefuttatásához jó segédeszköz a cikk mellett leközölt Basic program, ami egy Markov-féle számítógépet modellez. A program eredetileg HOMELAB 3 gépre készült, de könnyen átírható más típusra is.

A$(I) és B$(I) az I-edik szabály bal és jobb oldalát tárolja, a B$ pedig az Inputszó, ill. a belőle származó módosítások.

A programban 10–90-ig történik a menü lekezelése. 120–220-ig a programok szerkesztője – vagyis a Markov interpreter editorja található. (120–160 listázza a programot, 170–220 pedig új sort vesz be.)

300-tól a program futtatója következik. Először az Inputszót veszi be, majd 320–350 megkeresi a legelőször alkalmazható szabályt. 360–370 végrehajtja a helyettesítést, és kiírja a gép aktuális állapotát. Így menet közben követhető, hogy mi történik. 380–400 a billentyűzet figyelése. Itt meg lehet állítani a futtatást, vagy vissza lehet térni. 500 a végeredményt írja ki.

A program a korábban mondottakkal egyezésben az inputszó elé és mögé egy-egy #-et tesz.

A Markov-programok futtatása közben senki ne lepődjön meg a sebességen: egy-egy szabály kitalálása 1–10 másodpercig is eltarthat, hiszen állandóan a különben is lassú Basic leglassúbb részét, a sztringműveleteket használjuk. Gépi nyelven ez nyilván sokkal gyorsabb lenne, de a mai számítógépekben sajnos így sem lenne igazán hatékony ez az algoritmus. Alkalmas hardverrel viszont igen egyszerűen programozható gépet lehetne készíteni. Hatékonyságát az is növelhetné, hogy az egymástól független szabályokat egyszerre is lehetne tesztelni, így a párhuzamos működést is meg lehetne oldani.

Az is nyilvánvaló, hogy ez a nyelv végül is nem alkalmas hosszadalmas aritmetikai műveletek elvégzésére, vagy folyamatszabályozásra. Ellenben szövegszerkesztésre, adatkezelésre, szintaktikai elemzésre, szövegek kódolására kiválóan alkalmas lehetne, hiszen programjai tulajdonképpen nem mások, mint az adott feladat pontos és szigorú definíciói.

Ezért ha a napi gyakorlatban ma még közvetlenül nem is használható – mindenkinek épülésére szolgál megismerkedni egy más elvű gondolkodásmóddal.

Lukács Endre

```
10 L=255: Dim A$(50): Dim B$(50)
20  Print chr$(12)
30  Print  cur 10,0"1. Programozás" cur 10,2"2. Futtatás"
40 A$=Inkey$
50  If A$="1" then  Goto 120
60  If A$="2" then  Goto 300
90  Goto 40
120  Print chr$(12): For V=1 to 50
130  If A$(V)="" then  Pop : Goto 170
140  Print  cur 32*int(V/26),V-25*(V>25);rgh$(str$(V),2)". ";
150  Print A$(V)" --> "B$(V)
160  Next
170  Input  cur 0,26"A parancs sorszáma:"A$
180  If A$="" then  Goto 20
190 V=val(A$): If V>50 then  Goto 20
200  Print : Input "Az első szó:"A$(V)
210  If A$(V)="" then  Goto 120
220  Print : Input "A második szó:"B$(V): Goto 120
300  Print chr$(12): Input  cur 0,5"Az input szó:"B$
306 B$="#"+B$+"#": Print  cur 0,10"Az aktuális szó:"
307  Print  cur 0,15"Az aktuális parancs:"
320  For V=1 to 50:A=len(A$(V)): If A=0 then  Pop : Goto 500
330  For X=1 to len(B$)+1-A
340  If mid$(B$,X,A)=A$(V) then  Goto 360' Next
350  Next : Goto 500
360  Print  cur 0,11;B$,, cur 0,16;A$(V)" --> "B$(V),
370  Pop :B$=lft$(B$,X-1)+B$(V)+mid$(B$,X+A,L)
380 A$=Inkey$: If A$="" then  Goto 320
390 A$=Inkey$: If A$="S" then  Pop : Goto 30
400  If A$=" " then  Goto 320  Goto 390
500  Print  cur 0,20;"Az output szó:": Print B$
520  Goto 30

Ok
```