Conference Paper for MAFIOK 2016.
Óbuda University Alba Regia Technical Faculty, Hungary, Székesfehérvár

LaTeX-ben írt feladatok szövegének és TikZ ábráinak variálása az ec-sorter feladatvariáló programmal

Horváth Árpád
Óbudai Egyetem, Alba Regia Műszaki Kar, Székesfehérvár
Email: horvath.arpad@amk.uni-obuda.hu

Kulcsszavak: formális nyelvek, LaTeX, TikZ, Python, felsőoktatás

Kivonat—Kifejlesztettem egy ec-sorter nevű feladatvariáló programot, amely lehetőséget teremt rá, hogy az egyes feladatokat paraméterekkel ellátva különböző variációkat hozzunk létre azokból, és a végén egy nyomtatásra is alkalmas feladatlapot hozzunk létre a hallgatók számonkéréséhez. A program egyszerű LaTeX formátumú fájlban található feladatokat dolgoz fel. Az újabb változata lehetőséget nyújt arra, hogy bizonyos paramétereket a feladatszövegben elhelyezett táblázatból nyerjen ki, ily módon akár egy TikZ ábra feliratait. Ez lehetőséget teremt például arra, hogy például egy automata állapotdiagrammjával kapcsolatos feladatnak számos változatát állítsuk elő.

Abstract—I have developed the program called ec-sorter, that make it possible to create test papers with several version of the exercises. This can be possible if we use some parameters in those exercises. The program uses exercises stored in plain LaTeX format. The new version makes it possible to get the parameters from tables located in the text of the exercise. In this way we can change the text of a TikZ figure in the several version of the same exercise. For example in a state diagram of an automaton we can change the symbols on the arrows.

1 Bevezetés

A számonkérések és gyakorlás változatos feladatainak létrehozása figyelmet igénylő feladat. Sok esetben egy feladatváltozat kigondolása és a feladathoz tartozó ábrák létrehozása komoly munkát jelent. A feladat megkönnyítésére hoztam létre az EC programcsomagot, amelynek ec-sorter parancsa képes létrehozni a beállításoknak megfelelő részben véletlen feladatsorokat. A program LaTeX-fájlokban található speciális utasításokat tartalmazó feladatokat képes kiválogatni egy feladatsorhoz, és a speciális utasításoknak megfelelő módon különböző változatokat létrehozni az egyes feladatokból.

A LaTeX formátum mellett szól [1, 2], hogy ebben a szöveges formátumban elég könnyen lehet ábrákat létrehozni a LaTeX TikZ nevű csomagjának az utasításaival. Olyan ábrákat, amelyek akár matematikai képleteket is tartalmazhatnak [3]. A variálást lehetővé tevő utasításokat ezekben az

ábrákban is elhelyezhetjük, így akár az ábrák különböző változatait is létrehozhatja a program.

A variálásnak két módszerét ismeri a program. A táblázatos módszer egy táblázatból keresi ki az egyes változatokhoz szükséges adatokat. Ez a táblázat szintén a feladat szövegében található megjegyzésben elrejtve. A jelen cikkben bemutatott példák ezt a módszert használják, ezért erre a továbbiakban konkrét példát láthatunk.

A másik módszer inkább fizikai vagy mérnöki számításos példákban alkalmazható. Abban a módszer esetén az egyes mennyiségek közötti összefüggéseket adjunk meg képletekkel, és a szükséges számú fizikai mennyiség esetén megadjuk a számintervallumot is, amelyből az értékei kikerülnek. Ez a módszer képes variálni azt is, hogy melyik mennyiségeket kelljen megkeresnie a hallgatóknak a feladatban. A második módszer működésének részleteit egy korábbi konferencia proceedings kötetében írtam le [4].

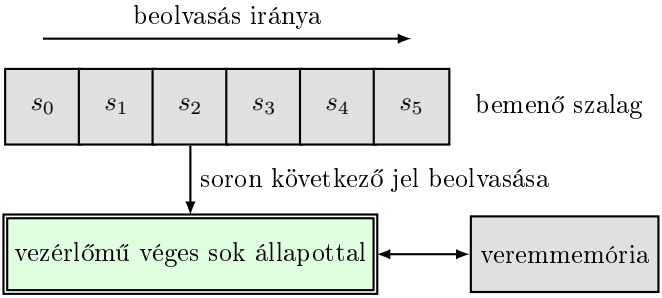
A programhoz főként magyar nyelvű feladatok állnak rendelkezésre több témakörhöz. Ezek a témakörök a következők: matematika (analízis, komplex számok, differenciálegyenletek), fizika, diszkrét matematika, híradástechnika, formális nyelvek és gépek, valamint egyéb informatikai területek.

A program a magyar szabályok mellett az angol szabályokhoz is képes igazodni. A nyelvi különbségek a programfutás során megjelenő üzeneteken kívül a véletlenszerűen kiválasztott számok írásmódjában van: például különbözőképpen írjuk magyar és angol feladatban a $6{,}022 \cdot 10^{23}$ számot.

2 A végrehajtáshoz szükséges szoftverek

A feladatok kiválogatását, és a variációk létrehozását egy Python programozási nyelven írt program végzi [5]. A variációkat először szintén LaTeX-fájlként hozza létre a program, de ebből a beállításoktól függően PDF vagy PostScript formátumot hoz létre a LaTeX szövegformázó rendszer segítségével, amelyek már közvetlenül nyomtatásra alkalmasak. A programot jelenleg Linux (Debian, Ubuntu) alatt teszteltem, de a megírt kódot vélhetően könnyű módosítani úgy, hogy akár Windows alatt is futtatható legyen, vagy akár egy weboldalon lehessen létrehozni a teszteket. Jelenleg a program a Python 2-es változataival (2.5–2.7) működik, de tervben van a 3-as változatra történő áttérés [6].

A program futtatásához tehát a Python értelmező és a LaTeX szövegformázó rendszer telepítése szükséges. Az előbbi általában a Linux-terjesztések esetén alapból telepítve van, a LaTeX pedig azok szoftvertárolóiból könnyen telepíthető.



1. ábra. A veremautomaták felépítése. A véges automaták esetén a vermet nem használjuk. Egy konkrét automata vezérlőművét állapotdiagrammal szemléltethetjük. A bemenő szalagon szimbólumok sorozata, más néven egy szó, található (itt 5 darab). A vezérlőmű feladata, hogy a szóról (programról) megállapítsa, hogy helyes vagy nem.

3 A formális nyelvek és az automaták tömör összefoglalása

A számítástudományban fontos szerepe van a formális nyelvek elméletének, és a nyelvek szintaktikai elemzésében szerepet játszó automatáknak. Ezt a témakört az informatikushallgatók tanulják az egyetemeken, és több magyar nyelvű tankönyv is leírja a témát [7, 8, 9].

A formális nyelvek alatt valamilyen szimbólumsorozatokból álló halmazokat értünk. Ezeket a szimbólumsorozatokat általában szavaknak nevezi a szakirodalom. Gyakran a formális nyelveket nyelvtanokkal adjuk meg, hasonlóan a beszélt nyelvekhez. Ha összegyűjtjük azokat a szavakat, amelyeket a nyelvtanból le lehet vezetni, akkor egy nyelvek kapunk tehát, amelyet a nyelvtan által generált nyelvnek nevezünk.

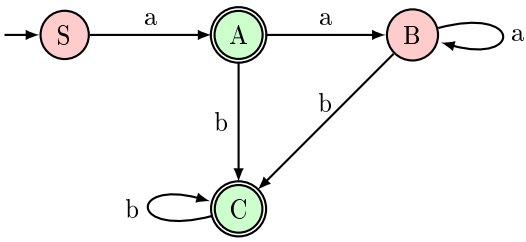
Például a C programozási nyelvnek is van ilyen nyelvtana, és az általa generált nyelv tartalmazza az összes szintaktikailag helyes C-programot. A szokásos elnevezés szerint tehát ezeket a C-programokat nevezzük ebben az esetben a C programozási nyelv szavainak. Természetesen ez a nyelv végtelen sok C-programot tartalmaz, hiszen akármilyen mélységben egymásba illeszthetünk ciklusokat, feltételes elágazásokat.

A nyelvtani szabályok megengedett alakjai alapján a nyelvtanokat osztályokba soroljuk. A nyelveket is osztályokba soroljuk azalapján, hogy milyen osztályba tartozó nyelvtan szükséges a generálásukhoz. Minden nyelvosztályhoz van egy olyan automatatípus, amely az adott nyelvosztályba tartozó nyelvek esetén képes felismerni, hogy egy szó hozzá tartozik-e a nyelvhez, vagy nem. Egy programozási nyelv esetén ez annak felel meg,

hogy az automata képes ellenőrizni, hogy szintaktikailag helyes programot írtunk-e.

Az automaták matematikai konstrukciók, mégis ezek alapján lehet megtervezni egy olyan szoftvert vagy eszközt, amely a nyelv felismerésére (esetleg fordítására) képes. Az automatáknak van egy bemenő szalagjuk, amiről a rajta található szó egyes szimbólumait tudja olvasni egyesével, balról jobbra (1. ábra). A bemenő szalagon található szó határozza meg, hogy milyen állapotokon fog keresztülhaladni az automata. (Vannak nem determinisztikus automaták is. Azoknál egy szimbólumsorozat több állapotsorozatot is meghatározhat, mivel egy állapotból ugyanaz a szimbólum többfelé is vezethet, de a szabály könnyen általánosítható azokra is.)

Az automatákat lényegében a vezérlőművük különbözteti meg. Ennek a vezérlőműnek véges sok állapota lehet. A különbség az egyes automaták között abban rejlik, hogy melyik automata milyen módon vált állapotot egy adott helyzetben. Az egyes automaták szabályainak egyik szemléletes ábrázolási módja az állapotdiagramm. Az állapotdiagrammon az egyes állapotoknak körök felelnek meg. Azt, hogy egy helyzetben melyik állapotból melyikbe mozog, nyilakkal jelöljük. A nyilak feliratai mutatják az adott helyzetet. Az állapotok közül lehetnek kitüntetett állapotok, az úgynevezett végállapotok. Ezeket az állapotdiagrammon dupla karikázással jelöljük. Az automata akkor ismer fel egy szót, ha végigolvasva a szót egy ilyen végállapotba kerül. (A veremautomatáknál van olyan működési mód, amikor nem a végállapotba kerülést figyeli, de a többi részlet ott is hasonló.)



2. ábra. Egy véges automata állapotdiagrammja. A szó egyes szimbólumainak olvasása során egyik állapotból másikba mozog. A dupla karikás állapotok a végállapotok. Ha a szó végére érve ilyen állapotba jut, akkor felismeri a szót. Ez a helyzet például az *abb* szó esetén. Ha nem végállapotba jut (pl. *aaa*) vagy elakad (*aba*), akkor nem ismeri fel a szót.

Az automaták legegyszerűbb típusánál, a véges automatáknál, a pillanatnyi állapot és a bemenő szalagról olvasott betű fogja meghatározni, hogy melyik lesz a következő állapot. Az egyes nyilakon tehát csak egyetlen felirat lesz: az olvasott szimbólum. Ez az automatatípus tehát nem használja a veremmemóriát. Egy véges automata állapotdiagrammját mutatja a 2. ábra.

A veremautomata még a veremmemóriát is használhatja bizonyos dolgok nyomonkövetésére. Azt, hogy a pillanatnyi állapotból melyik állapotba mozdul át, a beolvasott szimbólumon kívül a verem legfelső szimbóluma is meghatározhatja, és minden mozgás

után még írhat is valahány szimbólumot a veremre. A mozgásszabályt jelképező nyílra tehát ebben az esetben a bemenő szalagról beolvasott szimbólum mellett a veremről olvasott szimbólumot, és a veremre írt szimbólumokat is fel kell tüntetni.

A szűkebb (úgynevezett reguláris) nyelvosztály elemzésére az egyszerűbb véges automaták elegendőek, a programozási nyelvek elemzéséhez viszont veremautomaták kellenek. Az említett két nyelvosztályon felül még bővebbek is léteznek. És ehhez tartozó nagyobb tudású automaták illetve gépek is. Ezeket a már említett tankönyvek részletezik.

4 A fájlok formája, amiből a feladatokat válogatjuk

Az ec-sorter parancs egy vagy több LaTeX-fájlban képes megtalálni a keresett feladatokat a feladat sorszáma alapján.

Az egyes feladatoknak `feladat` környezetben kell lennie. Ennek a környezetnek egy kötelező argumentuma van, a feladatot azonosító kódszám. A kódszámot az ec-coder utasítás automatikusan generálja az újonnan létrehozott feladatok esetén, ha üresen hagyjuk a sorszám helyét.

A feladaton belül lehet még egy `megoldas` környezet is, amely a megoldást tartalmazza. Ezt a feladatsor generálása során az ec-sorter program kiveszi a szövegből. Létrehoz viszont egy másik PDF-fájlt is, amelyben a megoldások szerepelnek olyan formában, amely a feladatok javítását segíti. Ennél kap szerepet a `megoldas` környezet.

A függelékben szerepel egy feladat LaTeX-kódja. A továbbiakban ezt a programkódot fogjuk elemezni. A feladat a 62-es sorszámot kapta:

```
1 \begin{feladat}{62}
```

A feladat utolsó sorai a `megoldas` környezetet és a `feladat` környezet lezárását tartalmazzák:

```
44 \begin{megoldas}
45 A szót \ecChoose{elfogadja}{result}.
46 \end{megoldas}
47 \end{feladat}
```

(Az verb"ecChoose" parancs már a feladatvariálás táblázatos módszeréhez tartozik, ott fogom leírni.)

Mint láthatjuk a feladat szövegében lényegében tetszőleges LaTeX-kód helyet kaphat. Ebben a példában például egy ábrát helyeztem el a 15-dik és a 33-dik sor között.

5 A feladatvariálás táblázatos módszere

A feladatok variálásának a bevezetőben említett első módszerének alkalmazása során a feladatban levő `ecChoose` parancsokat egy táblázatból vett értékekkel helyettesítjük. A feladat megjegyzésben tartalmazza a táblázatot. A táblázat akárhol lehet a feladat szövegén belül.

A függelékben található feladat kódjában a 35-dik sortól található a

táblázat. Az elejét a `begin table()` sor, a végét a `end table` sor jelöli.

```
% begin table()
% a b word   result
% a b aabaaa elfogadja
% b a bbbbab elutasítja
% 0 1 000101 elfogadja
% 1 0 110100 "elutasítja, mert elakad"
% end table
```

A táblázat első sora a fejléc. Ez tartalmazza a változóneveket, amilyen néven a feladatban szövegében hivatkozunk az egyes értékekre. A példában a változónevek: a, b, word és result.

Ez a táblázat szóközökkel elválasztott oszlopokat tartalmaz. Az elválasztásra egy vagy több szóköz is alkalmazható, hogy az oszlopokat szemmel áttekinthetőbb formába rendezhessük. A result változónév előtt például több szóköz található, de egy szóközzel is feldolgozná a táblázatot a program. A szóközöket tartalmazó adatokat idézőjelbe kell rakni, ahogy a táblázat utolso sorában látható.

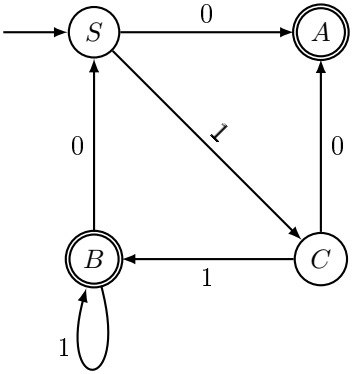
Az ec-sorter parancs minden egyes feladatban megvizsgálja, hogy van-e `ecChoose` parancs. Ha igen, akkor variációkat fog készíteni belőle az egyes feladatsorok számára.

Ha zárthelyi feladatlapokat készítünk, akkor megadjuk a zárthelyi csoportjainak a feladatszámait, és meg kell adnunk hogy hány variációt készítünk minden egyes csoprtból. Például ha A és B csoportot készítünk, és 3 variációt állítunk be, akkor mindkét csoportból 3 variációt fog készíteni a program, azaz összesen hat feladatsorunk lesz A_1, A_2, A_3, B_1, B_2 és B_3 nevekkel. Ha például az A csoportban szerepel a példaként mutatott 62-es számú feladat, akkor az a táblázat sorait összekeveri (permutálja) és az egyes változatok a soron következő sor adataival fogja feltölteni. Az A_1 csoportban az összekeverés utáni első sor adatai fognak szerepelni, az A_2-esben a második, az A_3-asban a harmadik. Természetesen ebben az esetben két változat nem kerül felhasználásra. (Amennyiben a változatok száma több, mint a sorok száma, akkor az összekevert sorokon ciklikusan halad végig a program. Eszerint, amennyiben 6 vagy több variációt állítunk be, a A_6-os csoport adatai megegyeznek az A_1-esével ennél a feladatnál.)

Az alábbiakban két változatot látunk a kapott feladatra:

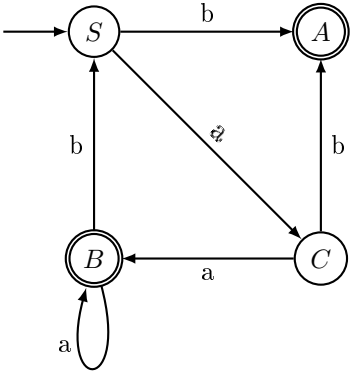
A_1 csoport

1. feladat Határozzunk meg egy olyan nyelvtant, amely ugyanazt a nyelvet generálja, mint amit az alábbi automata felismer! Elfogadja-e az automata a 110100 szót? Vezessük le a szót konfiguráció-sorozattal (amíg el nem akad)! Tegyük teljessé az automatát!



A_2 csoport

1. feladat Határozzunk meg egy olyan nyelvtant, amely ugyanazt a nyelvet generálja, mint amit az alábbi automata felismer! Elfogadja-e az automata a aabaaa szót? Vezessük le a szót konfiguráció-sorozattal (amíg el nem akad)! Tegyük teljessé az automatát!



Látható, hogy az egyik változatban a nyilakon 0-ák és 1-ek vannak, a másiknál a és b. Az is különbözik, hogy az automatának melyik szót kell felismernie. (Természetesen egy olyan automata, ahol 0-ák és 1-ek hatására mozog egyik állapotból a másikba, nem tud felismerni olyan szót, amelyben a és b szimbólumok vannak.)

Hogyan hozta létre az ec-sorter program ezeket a változatokat? Láthatjuk, hogy a kódban több helyen szerepel `ecChoose` parancs. Például az 5. sorban:

```
5  Elfogadja-e az automata
6  a \ecChoose{baaaba}{word} szót? (...)
```

Az `ecChoose` parancs két kötelező paraméterrel rendelkezik. A feladatgenerálás szempontjából csak a második paraméter a lényeges. Ez a változó neve. (Az első egy alapértelmezett érték. Ez jelenik meg az eredeti feladatsorban, amelyekből a feladatokat kiválogatjuk, ha a nyers, még `ecChoose` parancsokat tartalmazó forrásfájlokat lefordítjuk. Van egy elhagyható paramétere is a mértékegység, amellyel itt nem találkozunk.)

Amikor egy adott sort kiválasztottunk, akkor az `ecChoose` parancs helyére a változó megfelelő értéke helyettesítődik be. Az első változatban azért az 110100 szó szerepel, mert a táblázat 5. sora alapján helyettesít, melyben ez áll:

```
38 % a b word   result
39 %
40 % (...)
41 %
42 % 1 0 110100 "elutasítja, mert elakad"
```

A változók értéke tehát a következő lesz:

$$a = 1, \quad b = 0, \quad word = 110100 \quad (1)$$

$$result = \text{"elutasítja, mert elakad"} \tag{2}$$

Az ábrában találhatóak olyan részek, ahol az ecChoose-ban szereplő változó a. Ezek helyére 1-et helyettesít a program. Ahol b a változó, oda nullát. A megoldásnál meg megjelenik, hogy azért nem fogja felismerni az automata az 110100 szót, mert elakad a levezetés. (Természetesen a megoldás nem fog megjelenni a hallgatók feladatsorában.)

6 Összefoglalás és kitekintés

Az ec-sorter program képes a feladatsorok többféle változatának előállítására. A variálásának a táblázatos módszerét ismertettem részletesen. A formális nyelvekkel és automatákkal kapcsolatos feladatok esetén általában ez a módszer alkalmazható. Láthattuk, hogy ezzel a módszerrel TikZ-ábrákon is kicserélhetünk feliratokat.

A programot 2002 óta rendszeresen fejlesztem és használom. Nagy segítséget jelent különböző tárgyak zárthelyi-feladatsorainak létrehozásában. Ahhoz viszont, hogy más is könnyedén használhassa, a program még fejlesztésre szorul. A program átalakításával sokkal áttekinthetőbb könyvtárszerkezetet lehet kialakítani, amelyben könnyebben megtalálhatóak a megfelelő beállítófájlok.

A példafeladat teljes kódja:

```
\begin{feladat}{62}
Határozzunk meg egy olyan nyelvtant,
amely ugyanazt a nyelvet generálja,
mint amit az alábbi automata felismer!
Elfogadja-e az automata a
\ecChoose{baaaba}{word} szót?  Vezessük
le a szót konfiguráció-sorozattal (amíg
el nem akad)! Tegyük teljessé az
automatát!

\tikzset{
  >=latex,
  state/.style ={thick,circle,draw},
  finalstate/.style ={state,double}
}
\begin{tikzpicture}[scale=3]
\node[state] (S) at (0,1) {S}
  edge[<-,thick] (-.4,1);
\node[finalstate] (A) at (1,1) {A};
\node[finalstate] (B) at (0,0) {B};
\node[state] (C) at (1,0) {C};
\path[thick,->]
(S) edge node[above]
      {\ecChoose{b}{b}} (A)
    edge node[above,sloped]
      {\ecChoose{a}{a}} (C)
(B) edge node[left]
      {\ecChoose{b}{b}} (S)
  edge[loop below] node[near end,left]
      {\ecChoose{a}{a}} ()
(C) edge node[right]
      {\ecChoose{b}{b}} (A)
    edge node[below]
      {\ecChoose{a}{a}} (B);
\end{tikzpicture}

% begin table()
% a b word   result
% a b aabaaa elfogadja
% b a bbbbab elutasítja
% 0 1 000101 elfogadja
% 1 0 110100 "elutasítja, mert elakad"
% end table
\begin{megoldas}
A szót \ecChoose{elfogadja}{result}.
\end{megoldas}
```

```
\end{feladat}
```

Hivatkozások

[1] „The LaTeXproject site.” [Online]. Available: http://www.latex-project.org

[2] F. Wettl, G. Mayer, and P. Szabó, *LaTeX kézikönyv*. Budapest: Panem, 2004.

[3] „pgf – Create PostScript and PDF graphics in TeX.” [Online]. Available: http://www.ctan.org/pkg/pgf

[4] H. Arpad, „The two methods of the ec-sorter latex test paper creator to make several variation of exercises,” in *International Symposium on Applied Informatics and Related Areas, 2013*, 2013.

[5] „Official Python site.” [Online]. Available: http://www.python.org

[6] M. Summerfield, *Programming in Python 3: A Complete Introduction to the Python Language*. Addison-Wesley Professional, 2010.

[7] I. Bach, *Formális nyelvek*. TypoTeX Kiadó, 2001.

[8] Dömösi-Falucskai-Horváth-Mecsei-Nagy, *Formális nyelvek és automaták*. Kelet-magyarországi Informatika Tananyag Tárház, 2011.

[9] Demetrovics-Denev-Pavlov, *A számítástudomány matematikai alapjai*. Tankönyvkiadó, 1985.