Formális nyelvek és fordítóprogramok

Programozási feladatok (laborfeladatok)

Bodó Zalán, Kopacz Anikó, Lieb Hanna

Babeș–Bolyai Tudományegyetem
Matematika és Informatika Kar

I. Formális nyelvek és automaták

I.A.1. – Automata grafikus megjelenítése

I.A.2. – Automata reguláris nyelvtanból

I.A.3. – Elérhetetlen és nem produktív állapotok

I.A.4. – NDVA

I.B.1. – Automaták ekvivalenciája

I.B.2. – Veremautomata

I.B.3. – Automata minimalizálása

I.B.4. – NDVA $\longrightarrow$ DVA

II. Fordítóprogramok

II.1. – Lexikális elemzés

II.2. – Szintaktikai elemzés

II.3. – Szemantikai elemzés

II.4. – Kódgenerálás

Tartalom

I. Formális nyelvek és automaták
- I.A.1. – Automata grafikus megjelenítése
- I.A.2. – Automata reguláris nyelvtanból
- I.A.3. – Elérhetetlen és nem produktív állapotok
- I.A.4. – NDVA
- I.B.1. – Automaták ekvivalenciája
- I.B.2. – Veremautomata
- I.B.3. – Automata minimalizálása
- I.B.4. – NDVA $\rightarrow$ DVA

II. Fordítóprogramok
- II.1. – Lexikális elemzés
- II.2. – Szintaktikai elemzés
- II.3. – Szemantikai elemzés
- II.4. – Kódgenerálás

I. Formális nyelvek és automaták

I.A.1. – Automata grafikus megjelenítése

I.A.2. – Automata reguláris nyelvtanból

I.A.3. – Elérhetetlen és nem produktív állapotok

I.A.4. – NDVA

I.B.1. – Automaták ekvivalenciája

I.B.2. – Veremautomata

I.B.3. – Automata minimalizálása

I.B.4. – NDVA $\longrightarrow$ DVA

II. Fordítóprogramok

II.1. – Lexikális elemzés

II.2. – Szintaktikai elemzés

II.3. – Szemantikai elemzés

II.4. – Kódgenerálás

I. Formális nyelvek és automaták

I.A.1. – Automata grafikus megjelenítése

Tetszőleges véges automata grafikus megjelenítése:
Írjunk egy programot, mely tetszőleges véges automata esetén megfelelő `dot` forráskódot generál, majd a megjelenítéshez használjuk a `Graphviz` nevű alkalmazás `dot` nevű programját (graphviz.org vagy webgraphviz.com).
Az automatát bemeneti fájlból olvassuk be.

A bemeneti állomány alakja

Egy automatát a következő módon adunk meg:

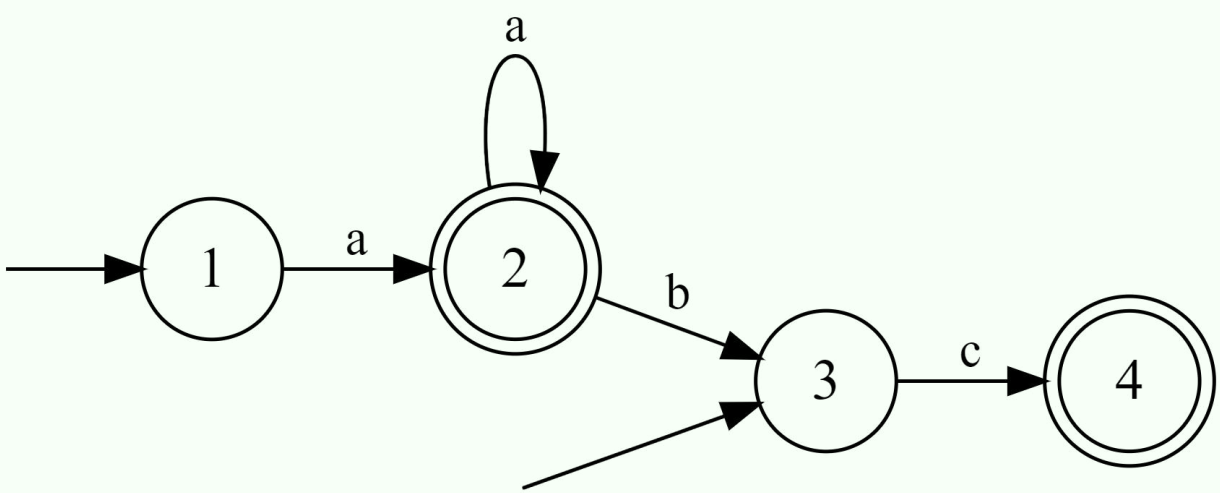
- első sor: **állapotok**, szóközökkel elválasztva
- második sor: **bemeneti ábécé elemei**, szóközökkel elválasztva
- harmadik sor: **kezdő állapotok**, szóközökkel elválasztva
- negyedik sor: **végállapotok**, szóközökkel elválasztva
- következő sorokban egy-egy átmenet:
  állapot bemenet állapot, szóközökkel elválasztva

I. Formális nyelvek és automaták
I.A.1. – Automata grafikus megjelenítése
I.A.2. – Automata reguláris nyelvtanból
I.A.3. – Elérhetetlen és nem produktív állapotok
I.A.4. – NDVA
I.B.1. – Automaták ekvivalenciája
I.B.2. – Veremautomata
I.B.3. – Automata minimalizálása
I.B.4. – NDVA ⟶ DVA
II. Fordítóprogramok
II.1. – Lexikális elemzés
II.2. – Szintaktikai elemzés
II.3. – Szemantikai elemzés
II.4. – Kódgenerálás

Példa bemenetre:

```
1 2 3 4
a b c
1 3
2 4
1 a 2
2 a 2
2 b 3
3 c 4
```

A fenti bemenetnek megfelelő automata:



I. Formális nyelvek és automaták
I.A.1. – Automata grafikus megjelenítése
I.A.2. – Automata reguláris nyelvtanból
I.A.3. – Elérhetetlen és nem produktív állapotok
I.A.4. – NDVA
I.B.1. – Automaták ekvivalenciája
I.B.2. – Veremautomata
I.B.3. – Automata minimalizálása
I.B.4. – NDVA ⟶ DVA
II. Fordítóprogramok
II.1. – Lexikális elemzés
II.2. – Szintaktikai elemzés
II.3. – Szemantikai elemzés
II.4. – Kódgenerálás

Kimenet: DOT kód

```
digraph G{
  ranksep=0.5;
  nodesep=0.5;
  rankdir=LR;
  node [shape="circle",fontsize="16"];
  fontsize="10";
  compound=true;

  i1 [shape=point, style=invis];
  i3 [shape=point, style=invis];
  2  [shape=doublecircle];
  4  [shape=doublecircle];

  i1 -> 1;
  i3 -> 3;
  1  -> 2 [label=a];
  2  -> 2 [label=a];
  2  -> 3 [label=b];
  3  -> 4 [label=c];
}
```


I. Formális nyelvek és automaták

I.A.1. – Automata grafikus megjelenítése

I.A.2. – Automata reguláris nyelvtanból

I.A.3. – Elérhetetlen és nem produktív állapotok

I.A.4. – NDVA

I.B.1. – Automaták ekvivalenciája

I.B.2. – Veremautomata

I.B.3. – Automata minimalizálása

I.B.4. – NDVA $\longrightarrow$ DVA

II. Fordítóprogramok

II.1. – Lexikális elemzés

II.2. – Szintaktikai elemzés

II.3. – Szemantikai elemzés

II.4. – Kódgenerálás

I.A.2. – Automata reguláris nyelvtanból

Reguláris nyelvtan átalakítása véges automatává.
A nyelvtant a következő módon adjuk meg a bemeneti állományban:

- 1. sor: nemterminális szimbólumok
- 2. sor: terminális szimbólumok
- 3. sor: kezdőszimbólum
- következő sorok: szabályok a köv. formában:

$$A\ a\ B$$

vagy

$$A\ a$$

ahol A, B nemterminális, a terminális szimbólum.

A kimeneti állomány formátumához lásd az I.A.1. feladatnál a bemeneti állomány alakját.

I. Formális nyelvek és automaták
I.A.1. – Automata grafikus megjelenítése
I.A.2. – Automata reguláris nyelvtanból
I.A.3. – Elérhetetlen és nem produktív állapotok
I.A.4. – NDVA
I.B.1. – Automaták ekvivalenciája
I.B.2. – Veremautomata
I.B.3. – Automata minimalizálása
I.B.4. – NDVA ⟶ DVA
II. Fordítóprogramok
II.1. – Lexikális elemzés
II.2. – Szintaktikai elemzés
II.3. – Szemantikai elemzés
II.4. – Kódgenerálás

I.A.3. – Elérhetetlen és nem produktív állapotok

Egy programban oldjuk meg:

(a) Írjunk eljárást, amely kizárja a determinisztikus véges automatából a nem elérhető állapotokat.

(b) Írjunk eljárást, amely kizárja a determinisztikus véges automatából a nem produktív állapotokat.

A bemeneti állomány alakjához lásd az I.A.1. feladatot.
A kimeneti formátum megegyezik a bemeneti formátummal, ahol a reprezentált automatából már kizártuk a nem elérhető vagy nem produktív állapotokat, illetve a hozzájuk tartozó átmeneteket, illetve az eredeti ábécében szereplő, már nem szükséges szimbólumokat.

I. Formális nyelvek és automaták
I.A.1. – Automata grafikus megjelenítése
I.A.2. – Automata reguláris nyelvtanból
I.A.3. – Elérhetetlen és nem produktív állapotok
I.A.4. – NDVA
I.B.1. – Automaták ekvivalenciája
I.B.2. – Veremautomata
I.B.3. – Automata minimalizálása
I.B.4. – NDVA → DVA
II. Fordítóprogramok
II.1. – Lexikális elemzés
II.2. – Szintaktikai elemzés
II.3. – Szemantikai elemzés
II.4. – Kódgenerálás

I.A.4. – NDVA

Írjunk programot, amely nemdeterminisztikus véges automata esetén vizsgálja, hogy az felismer-e egy adott szót.
Az automatát bemeneti fájlból olvassuk be, lásd az I.A.1 feladatot.
A szavakat beolvashatjuk fájlból vagy billentyűzetről.

I. Formális nyelvek és automaták
I.A.1. – Automata grafikus megjelenítése
I.A.2. – Automata reguláris nyelvtanból
I.A.3. – Elérhetetlen és nem produktív állapotok
I.A.4. – NDVA
I.B.1. – Automaták ekvivalenciája
I.B.2. – Veremautomata
I.B.3. – Automata minimalizálása
I.B.4. – NDVA ⟶ DVA
II. Fordítóprogramok
II.1. – Lexikális elemzés
II.2. – Szintaktikai elemzés
II.3. – Szemantikai elemzés
II.4. – Kódgenerálás

I.B.1. – Automaták ekvivalenciája

Vizsgáljuk meg, hogy két véges determinisztikus automata ekvivalens-e.
A bemenetet két állomány képezi, melyekben a két automata adatai szerepelnek. A bemenetek formátumához lásd az I.A.1. feladatot.

I. Formális nyelvek és automaták
I.A.1. – Automata grafikus megjelenítése
I.A.2. – Automata reguláris nyelvtanból
I.A.3. – Elérhetetlen és nem produktív állapotok
I.A.4. – NDVA
I.B.1. – Automaták ekvivalenciája
I.B.2. – Veremautomata
I.B.3. – Automata minimalizálása
I.B.4. – NDVA $\longrightarrow$ DVA
II. Fordítóprogramok
II.1. – Lexikális elemzés
II.2. – Szintaktikai elemzés
II.3. – Szemantikai elemzés
II.4. – Kódgenerálás

I.B.2. – Veremautomata

Vizsgáljuk meg, hogy egy veremautomata felismer-e egy adott szót (végállapottal vagy üres veremmel).
A bemeneti állomány alakja:

- 1. sor: állapotok, szóközökkel elválasztva
- 2. sor: bemeneti ábécé elemei, szóközökkel elválasztva
- 3. sor: veremábécé elemei, szóközökkel elválasztva
- 4. sor: kezdőállapot
- 5. sor: veremmemória kezdőjele
- 6. sor: végállapotok, szóközökkel elválasztva
- következő sorokban egy-egy átmenet:

$$q_i \ a \ z_i \ z_{i1}z_{i2}\ldots z_{ik} \ q_j$$

vagyis állapot, bemeneti szimbólum, veremszimbólum, új veremszimbólumok, új állapot.

I. Formális nyelvek és automaták
I.A.1. – Automata grafikus megjelenítése
I.A.2. – Automata reguláris nyelvtanból
I.A.3. – Elérhetetlen és nem produktív állapotok
I.A.4. – NDVA
I.B.1. – Automaták ekvivalenciája
I.B.2. – Veremautomata
I.B.3. – Automata minimalizálása
I.B.4. – NDVA ⟶ DVA
II. Fordítóprogramok
II.1. – Lexikális elemzés
II.2. – Szintaktikai elemzés
II.3. – Szemantikai elemzés
II.4. – Kódgenerálás

I.B.3. – Automata minimalizálása

Mondjuk meg adott determinisztikus véges automatáról, hogy az minimális-e vagy sem. Ha nem minimális, akkor adjunk meg egy vele ekvivalens minimális automatát.
A bemenet és kimenet formátumához lásd az I.A.1. feladatot.

I. Formális nyelvek és automaták
I.A.1. – Automata grafikus megjelenítése
I.A.2. – Automata reguláris nyelvtanból
I.A.3. – Elérhetetlen és nem produktív állapotok
I.A.4. – NDVA
I.B.1. – Automaták ekvivalenciája
I.B.2. – Veremautomata
I.B.3. – Automata minimalizálása
I.B.4. – NDVA $\longrightarrow$ DVA
II. Fordítóprogramok
II.1. – Lexikális elemzés
II.2. – Szintaktikai elemzés
II.3. – Szemantikai elemzés
II.4. – Kódgenerálás

I.B.4. – NDVA $\rightarrow$ DVA

Alakítsunk át egy adott nemdeterminisztikus véges automatát egy vele ekvivalens determinisztikus automatává.
A bemenet és kimenet formátumához lásd az I.A.1. feladatot.

I. Formális nyelvek és automaták
I.A.1. – Automata grafikus megjelenítése
I.A.2. – Automata reguláris nyelvtanból
I.A.3. – Elérhetetlen és nem produktív állapotok
I.A.4. – NDVA
I.B.1. – Automaták ekvivalenciája
I.B.2. – Veremautomata
I.B.3. – Automata minimalizálása
I.B.4. – NDVA $\rightarrow$ DVA
II. Fordítóprogramok
II.1. – Lexikális elemzés
II.2. – Szintaktikai elemzés
II.3. – Szemantikai elemzés
II.4. – Kódgenerálás

II. Fordítóprogramok

Amit a fordítóprogramnak ismernie kell:

- 2 alaptípus (egész és valós)
- változók deklarálása
- értékadás
- legkevesebb 3 aritmetikai művelet, zárójelek használata az ezekkel felépített kifejezésekben (pl. összeadás, kivonás, szorzás)
- kiírás, beolvasás
- `if-then-else` + `while` utasítás (egymásba ágyazás!)
- logikai relációk az `if` utasítás feltételében; legkevesebb 2 ilyen (pl. `==` és `!=`)
- logikai összekötők; legalább 2 ilyen (pl. a logikai és, vagy, vagy a tagadás)
- hibajelzés + hibaelfedés

A fordítóprogram szimbólumainak, utasításainak megválasztása tetszőleges.

I. Formális nyelvek és automaták
I.A.1. – Automata grafikus megjelenítése
I.A.2. – Automata reguláris nyelvtanból
I.A.3. – Elérhetetlen és nem produktív állapotok
I.A.4. – NDVA
I.B.1. – Automaták ekvivalenciája
I.B.2. – Veremautomata
I.B.3. – Automata minimalizálása
I.B.4. – NDVA ⟶ DVA
II. Fordítóprogramok
II.1. – Lexikális elemzés
II.2. – Szintaktikai elemzés
II.3. – Szemantikai elemzés
II.4. – Kódgenerálás

Pluszfeladatok:

- tömb típus bevezetése (0.2p)
- lokális változók (0.4p)
- típuskonverzió (0.4p)

I. Formális nyelvek és automaták

I.A.1. – Automata grafikus megjelenítése

I.A.2. – Automata reguláris nyelvtanból

I.A.3. – Elérhetetlen és nem produktív állapotok

I.A.4. – NDVA

I.B.1. – Automaták ekvivalenciája

I.B.2. – Veremautomata

I.B.3. – Automata minimalizálása

I.B.4. – NDVA $\longrightarrow$ DVA

II. Fordítóprogramok

II.1. – Lexikális elemzés

II.2. – Szintaktikai elemzés

II.3. – Szemantikai elemzés

II.4. – Kódgenerálás

II.1. – Lexikális elemzés

Listázó program elkészítése **Flex**-szel.

Ehhez természetesen szükséges a programnyelv elgondolása, előzetes megtervezése, hogy tudjuk, milyen terminális szimbólumok (lexikális egységek, tokenek) fognak szerepelni benne. A listázó program kimenete olyan szöveges fájl legyen, melyben minden sor a programnyelv egy terminális szimbólumát tartamazza a következő formában:

`[sor: <x>, oszlop: <y>, hossz: <z>] <terminális szimbólum>`

ahol `<x>`, `<y>` és `<z>` rendre a terminális szimbólum sora a bemeneti fájlban, a szimbólum kezdeti karakterpozíciója az illető sorban, illetve a lexikális egység hossza, a `<terminális szimbólum>` pedig maga a terminális szimbólum.
(A listázó programban használhatjuk az `yylineno` változót.)

I. Formális nyelvek és automaták
I.A.1. – Automata grafikus megjelenítése
I.A.2. – Automata reguláris nyelvtanból
I.A.3. – Elérhetetlen és nem produktív állapotok
I.A.4. – NDVA
I.B.1. – Automaták ekvivalenciája
I.B.2. – Veremautomata
I.B.3. – Automata minimalizálása
I.B.4. – NDVA ⟶ DVA
II. Fordítóprogramok
II.1. – Lexikális elemzés
II.2. – Szintaktikai elemzés
II.3. – Szemantikai elemzés
II.4. – Kódgenerálás

II.2. – Szintaktikai elemzés

A többkarakteres lexikális egységek elnevezése + a listázó program módosítása, hogy a **Bison**-nak vissza is térítse a tokeneket + a grammatika megírása.

- Tervezzük meg a programnyelv grammatikáját használva a már definiált tokeneket, majd fordítsuk egybe a **Flex** és **Bison** által generált elemzőket. Küszöböljük ki a megjelenő léptetés/redukció és redukció/redukció típusú konfliktusokat.
- Jelezzük ki a lexikális és szintaktikai hiba pontos helyét (használjuk az `%error-verbose` direktívát is).
- Legalább 2 helyen használjuk a speciális `error` tokent.

I. Formális nyelvek és automaták
I.A.1. – Automata grafikus megjelenítése
I.A.2. – Automata reguláris nyelvtanból
I.A.3. – Elérhetetlen és nem produktív állapotok
I.A.4. – NDVA
I.B.1. – Automaták ekvivalenciája
I.B.2. – Veremautomata
I.B.3. – Automata minimalizálása
I.B.4. – NDVA $\longrightarrow$ DVA
II. Fordítóprogramok
II.1. – Lexikális elemzés
II.2. – Szintaktikai elemzés
II.3. – Szemantikai elemzés
II.4. – Kódgenerálás

II.3. – Szemantikai elemzés

Szimbólumtábla (hash) építése és szemantikai műveletek helyességének ellenőrzése.

- Készítsünk szimbólumtáblát a használt változók menedzselésére. Ellenőrizzük, hogy minden hivatkozott változó be volt-e jelentve, illetve nem történt-e újradeklarálás.
- [Lokális változók esetén megengedett egy már létező, azonos nevű változó eltakarása.]
- Értékadás és más műveletek (pl. aritmetikai műveletek változókkal) esetén ellenőrizzük a típusok helyességét. [Típuskonverzió esetén is ellenőrizzük a konverzió helyességét (pl. `double` $\rightarrow$ `int`).]
- Szemantikai hiba pozíciójának jelzése.

I. Formális nyelvek és automaták
I.A.1. – Automata grafikus megjelenítése
I.A.2. – Automata reguláris nyelvtanból
I.A.3. – Elérhetetlen és nem produktív állapotok
I.A.4. – NDVA
I.B.1. – Automaták ekvivalenciája
I.B.2. – Veremautomata
I.B.3. – Automata minimalizálása
I.B.4. – NDVA $\rightarrow$ DVA
II. Fordítóprogramok
II.1. – Lexikális elemzés
II.2. – Szintaktikai elemzés
II.3. – Szemantikai elemzés
II.4. – Kódgenerálás

II.4. – Kódgenerálás

Assembly vagy valamilyen magasszintű programkód generálása (pl. C/C++, Python stb.).

Írjunk **két** (rövidebb) programot az általunk készített programozási nyelvben (pl. prímszámteszt, n-edik Fibonacci szám kiszámítása, legnagyobb közös osztó meghatározása, valamely rendezési algoritmus stb.), bemutatva ezzel a nyelv tulajdonságait, képességeit.

I. Formális nyelvek és automaták
I.A.1. – Automata grafikus megjelenítése
I.A.2. – Automata reguláris nyelvtanból
I.A.3. – Elérhetetlen és nem produktív állapotok
I.A.4. – NDVA
I.B.1. – Automaták ekvivalenciája
I.B.2. – Veremautomata
I.B.3. – Automata minimalizálása
I.B.4. – NDVA $\longrightarrow$ DVA
II. Fordítóprogramok
II.1. – Lexikális elemzés
II.2. – Szintaktikai elemzés
II.3. – Szemantikai elemzés
II.4. – Kódgenerálás