University of Zagreb
Faculty of Electrical Engineering and Computing
Department of Electronics, Microelectronics, Computer and Intelligent Systems

INTRODUCTION TO COMPUTER THEORY

Ak. year 2022/2023

2. laboratory exercise

The task of the 2nd laboratory exercise is the program realization of the minimization of a deterministic finite automaton (DKA). Program achievement should be achieved by removing unreachable states and removing identical states. The input and output of the minimization program are text definitions of DKA. Figure 1 shows the basic operation of the program that needs to be implemented.

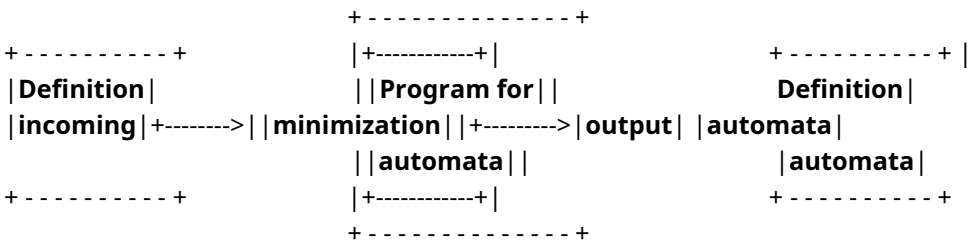

Figure 1 - Principle operation of the automaton minimization program

The format for recording the input and output automaton definition is as follows:

- 1st line: Set of comma-separated states, lexicographically ordered.
- 2nd line: A set of comma-separated alphabet symbols, lexicographically ordered.
- 3rd line: A set of acceptable states separated by a comma, lexicographically ordered.
- 4th line: Initial state.
- 5th line and all other lines: Transition function in the format currentState,Alphabet symbol->nextState, where the transitions are ordered lexicographically, observing only transition components. In other words, the transitions for the lexicographically smallest state are listed first, from the lexicographically smallest symbol to the largest, then the transitions for the lexicographically second smallest state, and so on.

An input automaton definition will never have more than 100 states or more than 100 alphabet symbols. The states and symbols of the alphabet are given as strings of lowercase letters of the English alphabet and decimal digits, where the length of the string is a minimum of 1 character and a maximum of 20 characters. The lexicographic order is defined based on the relationship of the ASCII values of the characters at the leftmost position where the two sets of characters differ. Furthermore, if the character string A is a prefix of the character string B, then A is lexicographically smaller than B. The decimal digits are smaller in the ASCII code than the lowercase letters of the English alphabet. For example, "xy" < "xyz", "xy" < "z", "1" < "a". This order can be obtained in all supported languages by built-in string comparison operators/functions.

In the automaton definition, each line ends with a newline character (\n). The last character in each automaton definition is also the end-of-line character. In other words, when you print the expected output, you should put a newline after the content of each line (including the last line).

An illustrative but incomplete example of the definition of DKA is shown in Figure 2. The numbers on the left side of the figure indicate

rare and not part of the definition.

01 q3,state1,state2,state3,state4,state5,z 02 a,befg,cce,ddd

03 q3, condition 2, condition 5
04 state 1
05 state1,a->state2
06 state3,befg->q3
. . .
N state5,ddd->state5

Figure 2 - An example of the definition of a finite automaton

The realized program should remove inaccessible and identical states and transitions related to these states from the automaton definition. At the same time, if the set of states is the same, it is necessary to preserve the lexicographic first state, while the other states should be removed. For example, if the states state1 and state2 and q3 are identical, it is necessary to remove the states state2 and q3, and preserve the state state1. Figure 3 shows the basic operation of the DKA program given in Figures 2.13 and 2.14 in the textbook, while Figures 4 and 5 show the input and expected output of the program.

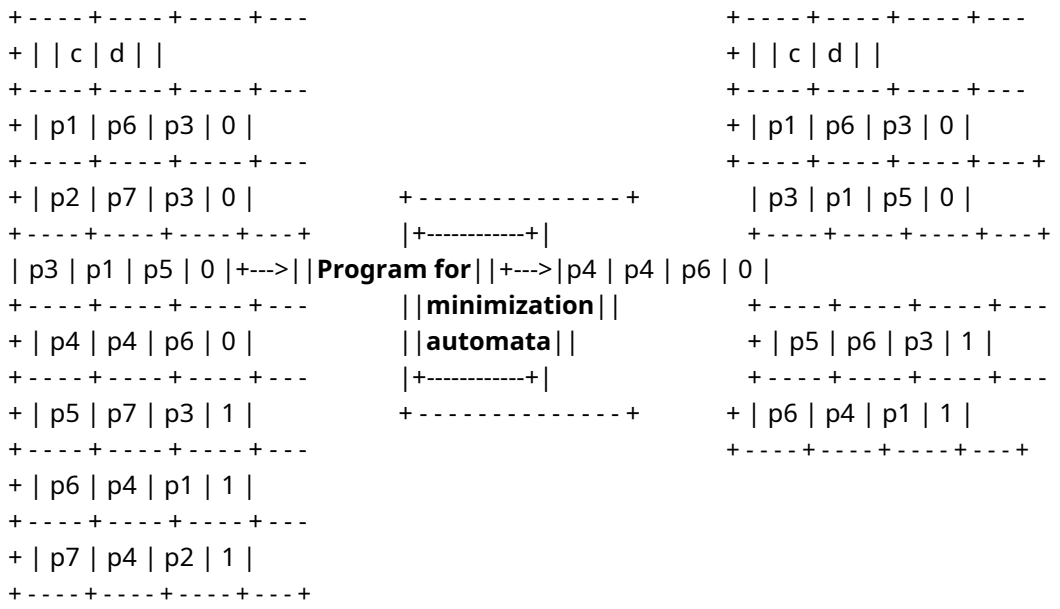

Figure 3 - Principle operation of the automaton minimization program for the example given in Figures 2.13 and 2.14 in the textbook

p1,p2,p3,p4,p5,p6,p7
CD
p5, p6, p7
p1
p1,c->p6
p1,d->p3
p2,c->p7
p2,d->p3
p3,c->p1
p3,d->p5
p4,c->p4
p4,d->p6
p5,c->p7
p5,d->p3
p6,c->p4
p6,d->p1

```
p7,c->p4
p7,d->p2
```

Figure 4 - Definition of the input automaton given by Figure 2.13 in the textbook

```
p1,p3,p4,p5,p6
CD
p5, p6
p1
p1,c->p6
p1,d->p3
p3,c->p1
p3,d->p5
p4,c->p4
p4,d->p6
p5,c->p6
p5,d->p3
p6,c->p4
p6,d->p1
```

Figure 5 - Expected output of the minimization program for the automaton definition from Figure 4

Notes:

1) In the definition of the output automaton, the lexicographic order of states, alphabet symbols and transitions should be preserved.

2) It is not necessary to check the correctness of the formatting of the input file or the correctness of the automaton. In other words, there will always be at least one state in the state set and at least one symbol in the alphabet symbol set, exactly one initial state will always be defined (though not necessarily the first specified state from the state set), all transitions for all states will be defined, there will be no transitions for undefined states or undefined alphabet symbols and the definitions will certainly be valid DKAs. It is not necessary that every automaton will have acceptable states. There will be no overlap between the state set and the alphabet symbol set.

3) It is not necessary that the procedures literally implement some of the algorithms described in the textbook, i.e. the operation of the algorithm is not checked, but only the final result. All the same, it is necessary to carry out the procedure of removing unreachable states first, and only then the procedure of removing identical states.

4) The time limit on program execution for any input definition of the automaton is 10 seconds. Since typical implementations of the minimization procedure will run in less than 1 second, 10 seconds is more than enough.

5) The entry point for Java solutions should be in the MinDka class, and the entry point for Python solutions should be in the MinDka.py file.