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

1. laboratory exercise

In the first laboratory exercise, the task is to program a simulator of a non-deterministic finite automaton with epsilon transitions (**ε-NKA**). The input to the automaton simulator is a textual record of its definition and the input string, and the output is a textual record of the sets of states the automaton has been in for each loaded character of the input string.The mode of operation of the program that needs to be completed as part of the exercise is shown in principle in Figure 1.

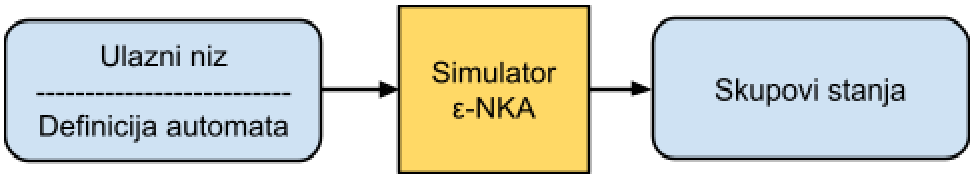

Figure 1 - Principle operation of the ε-NKA simulator

The format for recording the input string and definition of ε-NKA is:

- 1 line: Input strings separated by |. The symbols of each individual string are separated by a comma.
- 2nd line: Lexicographically ordered set of states separated by a comma.
- 3rd line: A lexicographically ordered set of alphabet symbols separated by a comma.
- 4th line: A lexicographically ordered set of acceptable states separated by a comma.
- 5th line: Initial state.
- 6th line and all other lines: Transition function in the format currentState, symbolAlphabet->setNextStates. In the set of NextStates, the states are separated by a comma.In the epsilon-transition definition, the alphabet symbol will be replaced by a sign**$**(symbol**ε** is replaced by a symbol**$**). The transitions are ordered lexicographically, looking only at the 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. The set of next states is empty if no corresponding transitions are defined. An empty set of states is denoted by the symbol**#**.

The length of the 1st line of the input will not exceed 1500 characters. An 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". The symbol $ is lexicographically the smallest, that is, smaller than all numbers and letters.

In the input record, each line ends with a newline character (\n). If no transition is defined for the pair (state, input_character) in the automaton definition, it will be considered that there is a transition:

state,input_character->#.

An example of the definition of ε-NKA is shown in Figure 2. The numbers on the left side of the figure indicate rows and are not part of the definition.

```
01 a,pnp,a|pnp,lab2|pnp,a|pnp,lab2,utr,utr 02
p5,s3,s4,st6,state1,state2
03 a, lab2, pnp, utr
04 p5
05 state 1
06 s3,a->state2
07 s3,lab2->p5,s4
08 s4,$->st6
09 s4,tr->p5,s3
10 state1,a->state2
11 state1,pnp->s3
12 state2,$->st6
13 state2,a->#
```

Figure 2 - Example of definition of ε-NKA

The realized ε-NKA simulator should print in which set of states the automaton was located for each input character of a particular string. Sets of states are separated by |, and the states within each set are ordered lexicographically. The results for each input string are printed on a separate line, that is, each output line is separated by a newline character (\n). Each record begins**together**which contains the initial state of the automaton (**remark:**the set can contain several states, depending on the transitions of the automaton). An example of the output for the automaton and the input strings defined in Figure 2 is shown in Figure 3. The numbers on the left side of the figure indicate the lines and are not part of the output.

```
01 state1|st6,state2|#|# 02 state1|s3|
p5,s4,st6 03 state1|s3|st6,state2 04
state1|s3|p5,s4,st6|p5,s3|#
```

Figure 3 - Example of the automaton simulator output

Notes:

1) When printing the simulator output, it is necessary to preserve the lexicographic order within each set of states.

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 input string defined, there will always be at least one state in the state set and at least one symbol in the alphabet symbol set, there will always be exactly one initial state defined (though not necessarily the first specified state in the state set). The automaton definition will not necessarily be complete, and in the event that no transition is defined for the pair (state, input_sign) in the automaton definition, it will be considered that there is a transition: state, input_sign->#. 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) The time limit on program execution for any input definition of the automaton is 10 seconds

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