Finite Automata

Github Link:
https://github.com/pauladam2001/Sem5_FormalLanguagesAndCompilerDesign/tree/master/Lab4/src

Finite Automata:

It is structured as a class with fields for the alphabet, states, initial states, final states and transitions.

The initial state is a String.

The alphabet, states and final states are respresented as a Set of Strings (Set<String>).

The transitions are respresnted as a Map, having as key a Pair of 2 String values (the source state and the value through which it accesses the destination state) and as value a Set

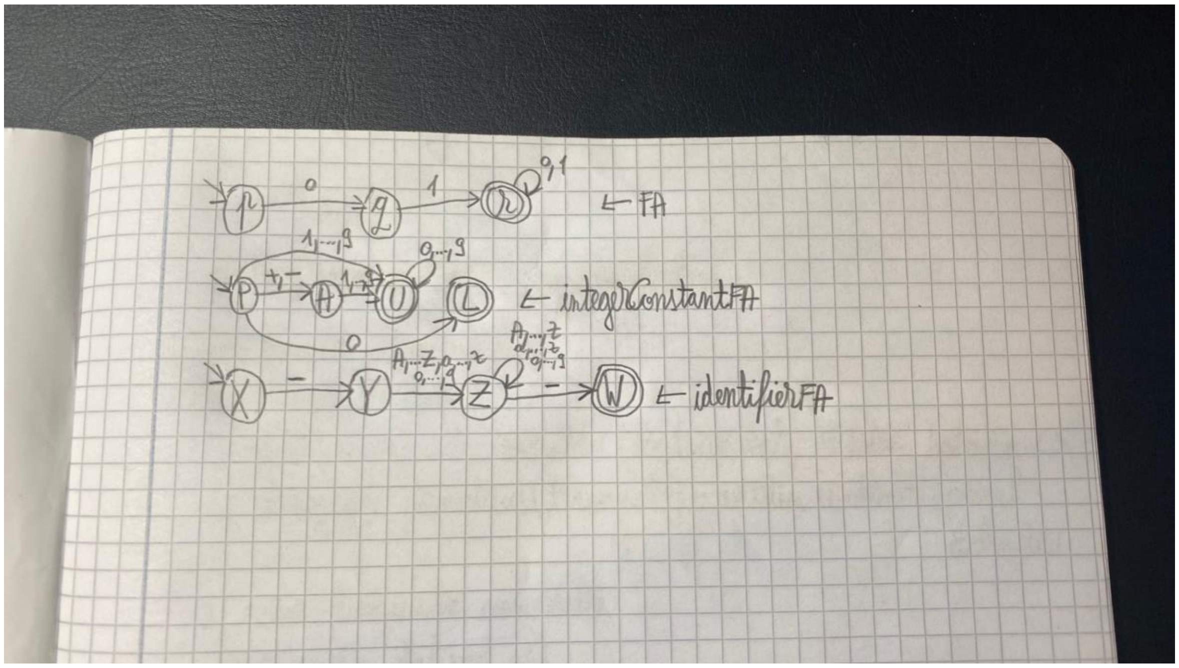
of Strings (the destination states) (Map<Pair<String, String>, Set<String>>).

In the case of a DFA a set will contain only one value.

We read the Finite Automata from a file, FA.in. Transitions that contain invalid states or characters, as well as duplicated transitions are ignored.

For a DFA, verify if a sequence is accepted by the FA:

We start from the initial state, we go through each character of the given sequence and we check that the Pair formed by the current state and the value of the current character in the sequence is mapped to a set containing exactly 1 value. This value will become the new current state in the next iteration. If the Pair is not mapped to any set the algorithm stops (the sequence is not accepted by the FA). When we reach the end of the sequence, if the last state belong to the set of final states, then the sequence is accepted by the FA.



FA.in:

letter ::= 'a' | 'b' | ... | 'z' | 'A' | 'B' | ... | 'Z'

digit ::= '0' | '1' | ... | '9'

alphabetChar ::= letter | digit

state ::= letter

transition ::= state alphabet state

states ::= {state}+

alphabet ::= {alphabetChar}+

initialState ::= state

finalStates ::= {state}+

FAFile ::= states '\n' alphabet '\n' initialState '\n' finalStates '\n' transitions

identifierFA.in:

identifier ::= '_' letter {(letter | digit)} '_'

letter ::= 'a' | 'b' | ... | 'z' | 'A' | 'B' | ... | 'Z'

digit ::= '0' | '1' | ... | '9'

integerConstantFA.in:

integer_constant ::= [sign] nz_digit {digit} | '0'

sign ::= '+' | '-'

nz_digit ::= '1' | '2' | ... | '9'

Integration with the Scanner:

We will use the files integerConstantFA.in and identifierFA.in in the LanguageSpecification class, instead of the regex used for identifying integer constants and instead of the one used for identifiers. These 2 FAs will be read and initialized when the LanguageSpecification class is initialized.