

Compilers 2021/2022

## **Projekt 1**

Anna Kelm  
110455

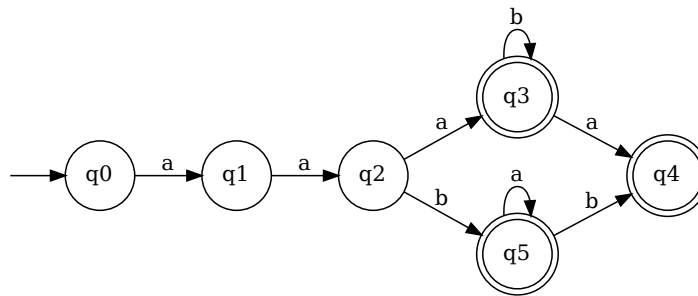
1 maja 2022

## 1 Cel

Celem projektu jest stworzenie programu sprawdzającego, czy podany na wejściu łańcuch znaków pasuje do wyrażenia regularnego:  $(aa|bb)a(b)^*(a)?$ , gdzie  $a, b \in \{0, 1\}$ .

Ponieważ wyrażenie  $(aa|bb)a(b)^*(a)?$  jest równoważne wyrażeniu  $aa((a(b)^*(a)?)|(b(a)^*(b)?))$ , zdecydowałam się pracować na tym drugim.

## 2 Niedeterministyczny automat skończony



Rysunek 1: Automat skończony dla wyrażenia  $aa((a(b)^*(a)?)|(b(a)^*(b)?))$ .

## 3 Tablica przejść

symbole wejściowe	stany					
	q0	q1	q2	q3	q4	q5
a	q1	q2	q3	q4	–	q5
b	–	–	q5	q3	–	q4
akceptujący?	N	N	N	A	A	A

Tabela 1: Tablica przejść dla automatu z rys. 1.

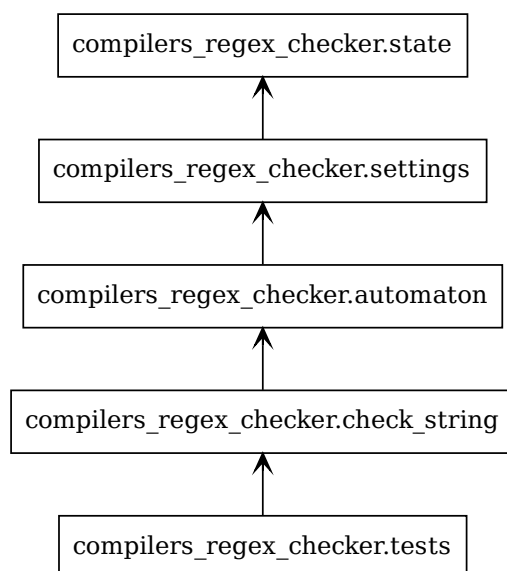
## 4 Struktura projektu

Pliki:

- automaton.py – klasa Automaton, która przyjmuje listę stanów i wykonuje tranzycje.
- check\_string.py – główny plik z funkcją realizujący sprawdzanie łańcucha wejściowego,
- dokumentacja.pdf
- README.md
- requirements.txt
- settings.py – definicje listy stanów i alfabetu,

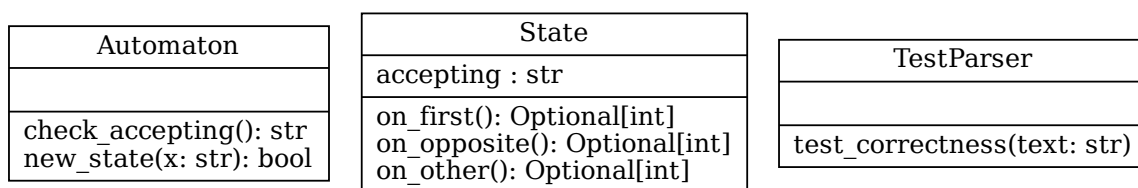
- state.py – klasa reprezentująca stan z dozwolonymi przejściami,
- tests.py – testy jednostkowe.

Diagram zależności między plikami w projekcie przedstawia rys. 2.



Rysunek 2: Diagram zależności między składnikami pakietu.

## 5 Klasy



Rysunek 3: Diagram klas stworzonych w projekcie.

### 5.1 State

Pola publiczne:

- `accepting` : ‘N’ lub ‘A’ w zależności od tego, czy jest to stan akceptujący.

Metody publiczne:

- `__init__(on_first, on_opposite, on_other, accepting)` – konstruktor klasy przyjmujący numer kolejnego stanu w odpowiedzi na znak odpowiadający `a`, następnie odpowiadający `b`, a następnie dla znaków spoza alfabetu. Jeśli przejście dla danego znaku nie istnieje, numer zawiera wartość `None`. Parametr `accepting` o możliwych wartościach `'A'` lub `'N'` wskazuje na to, czy stan jest akceptujący.
- `on_first` – zwraca numer kolejnego stanu dla symbolu odpowiadającego `a`,
- `on_opposite` – zwraca numer kolejnego stanu dla symbolu odpowiadającego `b`,
- `on_other` – zwraca numer kolejnego stanu dla symbolu odpowiadającego innym znakom.

## 5.2 Automaton

Metody publiczne:

- `__init__(first, states_list, alphabet)` – konstruktor klasy przyjmujący znak odpowiadający `a`, listę stanów zgodnie z numeracją, znaki alfabetu (np. `{0,1}`),
- `new_state(x)` – próbuje wykonać przejście - zmienić stan automatu dla symbolu `x`. Zwraca `'true'` dla sukcesu, `'false'` dla porażki (gdy przejście nie istnieje),
- `check_accepting` – zwraca wartość pola `State.accepting` dla bieżącego stanu.

## 6 Algorytm

1. Inicjalizacja instancji klasy `Automaton` listą stanów i pierwszym znakiem ze słowa (`a`). Przypisanie przejść w zależności od `a`.
2. Zmiana stanu automatu w odpowiedzi na kolejne znaki ze słowa. Jeśli dla danego znaku nie istnieje przejście, zwracany jest `'fałsz'` i wypisywane `'N'`.
3. W przypadku przetworzenia całego łańcucha znaków, wypisywane jest `'A'`, jeśli ostatni stan jest akceptujący – w przeciwnym przypadku: `'N'`.