Реализована структура данных описывающая произвольный автомат, будь то детерминированный, недетерминированный или недерминированный с эпсилон-переходами.
Структура данных включает в себя

1. Множество состояний
2. Начальное состояние
3. Множество конечных состояний
4. Функцию перехода заданную в виде словаря dict[from_state][symbol] = to_state
5. Алфавит

Класс Automaton включает в себя функцию get_type возвращающую тип автомата, поддерживается определение DFA, NFA, eNFA.

Автоматы программе можно задавать с помощью кода, а также в dot формате. Восстановление и запись в dot вынесены в отдельный модуль readwrite.py. В тесты добавлены примеры правильно и неправильно заданных в dot автоматов.

Задача получения по заданному недетерминированному автомату эквивалентного детерминированного реализована в виде чистой функции, принимающей на вход объект класса Automaton и возвращающей построенный детерминированный автомат, также являющийся объектом класса Automaton.

Алгоритм рассказывался нам ещё Расиным на первом курсе.

Основная идея: каждое состояние в детерминированном автомате представляет собой набор состояний недетерминированного.

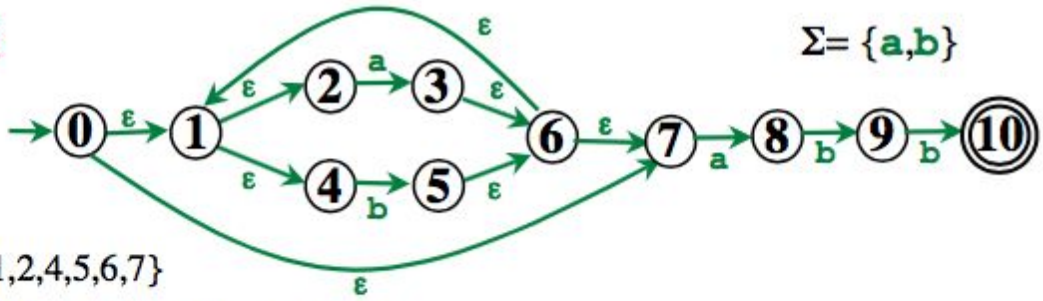
Используются вспомогательные функции:

1. _nfa_moves(nfa, from_states, symbol) для получения набора состояний достижимых из указанного множества по заданному переходу.
2. _epsilon_closure(nfa, state) для получения состояний, которые достижимы по ε-переходам из указанного состояния.
3. _epsilon_set_closure(nfa: Automaton, from_states: FrozenSet[str]) для состояний, которые достижимы по ε-переходам из множества указанных состояний.

Алгоритм:

1. Инициализируем для строящегося детерминированного автомата следующие вспомогательные структуры:
    - Словарь переходов вида dfa_transitions[current_dfa_state][symbol] = next_dfa_state, где каждое состояние детерминированного автомата – множество состояний недетерминированного.
    - Словарь состояний вида dfa_states[dfa_state] = state_mark, где dfa_state – множество состояний недетерминированного автомата, а state_mark – новая метка для состояния недетерминированного.
    - Очередь из непомеченных состояний. В неё будут добавляться уникальные новые наборы состояний недетерминированного автомата, которые будут являться впоследствии состояниями детерминированного.
2. Получим набор состояний достижимых из начального по эпсилон переходу, добавим это множество в словарь состояний детерминированного автомата приписав к этому множеству метку, полученное множество состояний добавим в очередь непомеченных
3. В цикле, пока существуют непомеченные состояния, будем искать следующие множества состояний недетерминированного автомата доступные по символу алфавита и эпсилон переходу. Те множества, что не были встречены ранее будут добавляться в очередь непомеченных. При этом пополняется словарь переходов для исходного состояния и текущего символа алфавита и алфавит используемых в ДКА символов.
4. По окончании работы цикла строится объект детерминированного автомата с полученным алфавитом, множеством состояний, начальным состоянием, множеством конечных состояний (если среди состояний во множествах состояний детерминированного автомата присутствует то состояние что было в недетерминированном конечным, то такое состояние детерминированного считается также конечным), функцией перехода.

Example



Process C = {1,2,4,5,6,7}

$Move_{DFA}$(C,a) = {1,2,3,4,6,7,8} = B
$Move_{DFA}$(C,b) = {1,2,4,5,6,7} = C

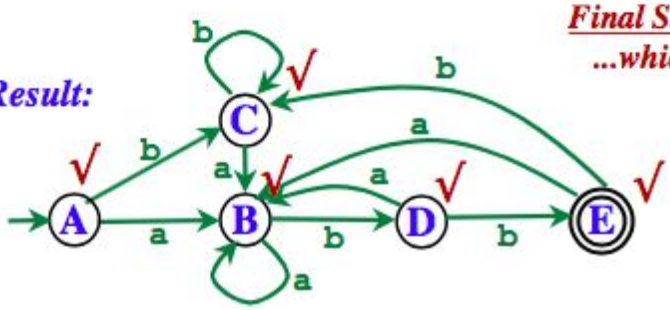
Process D = {1,2,4,5,6,7,9}

$Move_{DFA}$(D,a) = {1,2,3,4,6,7,8} = B
$Move_{DFA}$(D,b) = {1,2,4,5,6,7,10} = E

Process E = {1,2,4,5,6,7,10}

$Move_{DFA}$(E,a) = {1,2,3,4,6,7,8} = B
$Move_{DFA}$(E,b) = {1,2,4,5,6,7} = C

Final Result:



Final States in DFA?
...which state(s) contain 10?