Документация и теория для функции `check_string`

Супрун Артём Сергеевич

14 апреля 2025 г.

Содержание

1 Введение

В данном документе представлена подробная документация и теория реализации функции `check_string` на языке Python. Основная идея заключается в применении детерминированного конечного автомата (ДКА) для проверки того, что входная строка оканчивается последовательностью `"ab"`. Такой подход демонстрирует классический метод валидации входных данных и синтаксического анализа.

2 Теория конечных автоматов

Конечные автоматы являются базовой моделью вычислений, используемой для распознавания регулярных языков.

2.1 Определение ДКА

Детерминированный конечный автомат (ДКА) определяется следующими элементами:

- **Множество состояний:** Q.

- **Алфавит:** Σ.
- **Функция переходов:** $\delta : Q \times \Sigma \to Q$.
- **Начальное состояние:** $q_0 \in Q$.
- **Множество принимающих состояний:** $F \subseteq Q$.

Для рассматриваемого примера:

- $Q = \{0, 1, 2\}$
- $\Sigma = \{a, b\}$
- Начальное состояние: $q_0 = 0$
- Принимающее состояние: $F = \{2\}$

2.2 Функция переходов

Переходы между состояниями задаются следующей таблицей:

	a	b
0	1	0
1	1	2
2	1	0

Это означает:

- Из состояния 0 по символу `a` происходит переход в состояние 1, а по символу `b` автомат остаётся в состоянии 0.
- В состоянии 1 символ `a` сохраняет состояние, а символ `b` переводит автомат в состояние 2.
- Из состояния 2 символ `a` переводит автомат в состояние 1, а символ `b` возвращает его в состояние 0.

2.3 Распознаваемый язык

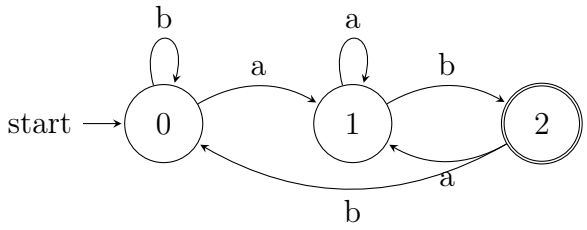
Автомат обрабатывает строку, начиная с начального состояния. Если после прочтения всей строки итоговое состояние оказывается равным принимающему (в данном случае, 2), строка считается принадлежащей языку, то есть:

$$L = \{w \in \{a, b\}^* \mid \delta^*(0, w) = 2\}$$

Анализ переходов показывает, что строка принимается только в случае, если она **заканчивается на последовательность** `"ab"`.

2.4 Графическое представление автомата

Ниже представлена диаграмма состояний данного ДКА:



3 Документация к коду

Ниже приведён исходный код функции `check_string` на Python с подробным описанием механизма работы.

3.1 Исходный код на Python

```python
def check_string(s: str) -> bool:
    state = 0

    for char in s:
        if char not in ('a', 'b'):
            return False

        if state == 0:
            if char == 'a':
                state = 1
        elif state == 1:
            if char == 'a':
                state = 1
            elif char == 'b':
                state = 2
        elif state == 2:
            if char == 'a':
                state = 1
            elif char == 'b':
                state = 0

    return state == 2
```

Листинг 1: Функция check$_string$

3.2 Описание работы функции

1. **Инициализация.** Устанавливается начальное состояние (`state = 0`).

2. **Валидация входных символов.** Каждый символ строки проверяется на принадлежность множеству $\{'a', 'b'\}$; присутствие недопустимого символа мгновенно приводит к возврату `False`.

3. **Переходы между состояниями.**

- Если текущее состояние 0 и символ равен `'a'`, происходит переход в состояние 1.
- При состоянии 1: символ `'a'` сохраняет текущее состояние, а символ `'b'` переводит автомат в состояние 2.
- Если состояние 2: символ `'a'` переводит автомат в состояние 1, а символ `'b'` возвращает его в состояние 0.

4. **Проверка результата.** По окончании обработки строки, если итоговое состояние равно 2, функция возвращает `True` (строка принимается), иначе — `False`.

3.3 Пример использования

Ниже приведён пример проверки набора тестовых строк с использованием функции `check_string`:

```
test_strings = [
    "ab",
    "aab",
    "abb",
    "aabb",
    "baba",
    "baab",
    "abc",
    ""
]

for s in test_strings:
    result = check_string(s)
    print(f"              {s!r}: {'                ' if result
   else '                    '}")
```

Листинг 2: Пример использования функции check$_string$

В данном примере строки, оканчивающиеся на `"ab"` (например, `"ab"`, `"aab"`, `"baab"`), возвращают `True`. Остальные строки не удовлетворяют условию либо из-за отсутствия требуемой последовательности, либо из-за наличия недопустимых символов.

4 Заключение

Представленный алгоритм демонстрирует применение конечных автоматов для проверки строк по заданному шаблону. Такой метод широко используется при валидации данных, синтаксическом анализе и во многих других задачах информатики. Реализацию можно легко расширить для поддержки более сложных языков и правил, что позволяет создавать гибкие инструменты для автоматизации обработки входных данных.