Дискретная математика Лабораторная работа

Вариант 2: Извлечение дат из текста (форматы: `DD.MM.YYYY` и `DD Month YYYY`)

---

1. Задача

Необходимо разработать программный модуль на языке Python, реализующий детерминированный конечный автомат (ДКА). Автомат должен принимать на вход строку произвольного текста и извлекать из нее даты, соответствующие двум паттернам:

1. **Цифровой:** День.Месяц.Год (например, `3.01.2025`)
2. **Словесный:** День Месяц(словом) Год (например, `2 декабря 1934`)

Также требуется реализовать графическое представление автомата и покрыть код тестами.

---

2. Проектирование автомата

Логика работы автомата построена на последовательной смене состояний при чтении символов строки. Ключевая особенность данного варианта это ветвление после считывания дня: автомат должен определить тип разделителя (точка или пробел), чтобы понять, какой формат даты ожидать дальше.

Диаграмма переходов (Графическая модель):

Описание состояний:

- `START`: Начальное состояние, ожидание первой цифры дня.
- `DAY`: Накопление цифр дня. Переход дальше происходит при встрече точки (`.`) или пробела ( ).
- `SEP1`: Анализ первого разделителя.
  - Если была точка — переходим в ветку цифрового месяца.
  - Если был пробел — переходим в ветку словесного месяца.
- `MONTH_NUM` / `MONTH_WORD`: Считывание месяца (цифрами или буквами соответственно).
- `SEP2`: Ожидание второго разделителя перед годом.
- `YEAR`: Считывание цифр года.

- `FINISH`: Дата сформирована, запись результата в буфер.

---

3. Реализация

Для реализации был создан класс `DateDFA`. В методе `step` реализована логика переходов.

Определение состояний и инициализация: Для читаемости кода состояния вынесены в константы класса.

```
class DateDFA:
    S_START = 'START'
    S_DAY = 'DAY'
    S_SEP1 = 'SEP1'
    # Ветвление для разных типов месяцев
    S_MONTH_NUM = 'MONTH_NUM'   # для "01"
    S_MONTH_WORD = 'MONTH_WORD' # для "декабря"
    S_SEP2 = 'SEP2'
    S_YEAR = 'YEAR'

    def __init__(self):
        self.reset()
        self.found_dates = []
```

Ключевая логика ветвления (Состояние SEP1): В этом месте решается, по какой ветке пойдет автомат. Это позволяет избежать ложных срабатываний (например, исключает варианты вида `12.декабря.2020`).

```
# Состояние: Разделитель 1 (между Днем и Месяцем)
elif self.state == self.S_SEP1:
    # Если разделитель был точкой -> ждем цифру
    if char.isdigit() and self.current_sep == '.':
        self.buffer_month += char
        self.state = self.S_MONTH_NUM

    # Если разделитель был пробелом -> ждем букву (русский язык)
    elif self.is_rus_letter(char) and self.current_sep == ' ':
        self.buffer_month += char
        self.state = self.S_MONTH_WORD

    else:
        self.reset() # Ошибка формата, сброс
```

Сброс и валидация (Reset): Автомат является жадным до ошибок. При любом несоответствии ожидаемому символу происходит сброс в состояние `START`. Это позволяет игнорировать мусорные данные внутри текста.

Визуализация: Для генерации графического представления использована библиотека `graphviz`. Метод `visualize` строит граф на основе узлов и ребер, соответствующих логике ДКА.

---

4. Тестирование

Программа была протестирована на различных наборах данных.

Тест №1: Смешанный текст (из задания) *Вход:* `Событие произошло 2 декабря 1934 года. Ничего не произошло 3.01.2025.` *Ожидание:* Две даты, одна словесная, одна цифровая.
Результат:

```
[[2, 'декабря', 1934], [3, 1, 2025]]
```

Тест №2: Поток дат без текста *Вход:* `12.12.2020 01.05.1999`
Результат:

```
[[12, 12, 2020], [1, 5, 1999]]
```

Тест №3: Невалидные данные (шум) *Вход:* `Это 123 декабря, а это 12.привет.2000` *Результат:*

```
[]
```

Пояснение: `12.привет` отсеялось на этапе `SEP1`, так как после точки ожидалась цифра, а пришла буква.