Опис реалізації регулярного виразу на основі скінченних автоматів

1 Загальний опис

Мій код працює на базі скінченних автоматів, за допомогою станів яких я можу розробити обмежений функціонал Regex. Основою архітектури є абстрактний клас `State`, від якого наслідуються конкретні стани: `AsciiState`, `DotState`, `StarState`, `PlusState` та `TerminationState`.

Кожен стан має метод `check_self`, що перевіряє відповідність символу.

Під час побудови автомату вираз аналізується посимвольно, і для кожного символу створюється відповідний стан, який з'єднується з попереднім. Оператори `*` і `+` модифікують попередній стан, забезпечуючи циклічний перехід для повторення.

Метод `check_string` виконує перевірку вхідного рядка, використовуючи глибокий пошук з поверненням (DFS) по можливих переходах станів.

2 Класи і стани

1. **State (абстрактний клас)**
   Базовий клас для всіх станів. Містить список наступних станів (`next_states`) та абстрактний метод `check_self(char)`, який визначає, чи поточний символ задовольняє умову стану.

2. **StartState**
   Початковий стан. Він сам по собі не обробляє жоден символ, а лише вказує на перший реальний стан у побудованому автоматі.

3. **AsciiState**
   Стан, який приймає один конкретний ASCII-символ (наприклад, 'a' або '4'). Метод `check_self` повертає `True`, лише якщо символ збігається з визначеним.

4. **DotState**
   Універсальний стан, що приймає будь-який символ (еквівалент `.` у регексах). Метод `check_self` завжди повертає `True`.

5. **StarState**
   Циклічний стан, що відповідає за повторення підрядка 0 або більше разів (`*`). Він містить посилання на "внутрішній" стан, який має бути повторений, і створює петлю на себе.

6. **PlusState**
   Стан для оператора `+`, що означає 1 або більше входжень підрядка. Як і `StarState`, містить внутрішній стан, але для проходження по FSM хоча б один раз має бути відповідність.

7. **TerminationState**
   Фінальний стан. Не приймає жодного символу, сигналізує про кінець успішного розпізнавання.

8. **RegexFSM**
   Головний клас, який будує граф станів згідно з регулярним виразом. Метод `check_string` виконує пошук у побудованому графі, щоб з'ясувати, чи вхідний рядок відповідає шаблону.

3 Приклад виразу

Розглянемо вираз: `a*4.+hi`

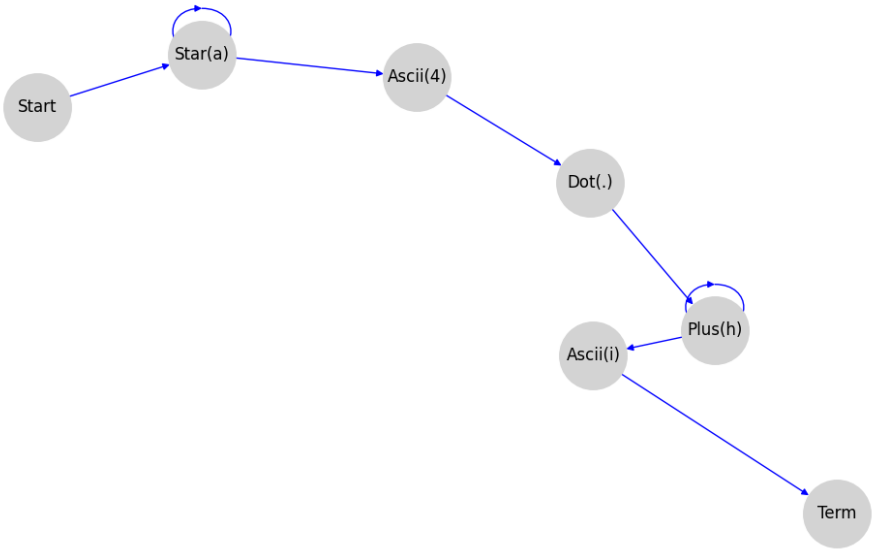

Рис. 1: Граф автомату для виразу `a*4.+hi`