# Лексический анализ динамически формируемых строковых выражений

**Автор:** Полубелова Марина Игоревна, 444 гр.
**Научный руководитель:** ст.преп. Григорьев С.В.
**Рецензент:** программист ООО “ИнтеллиДжей Лабс” Беляков А.М.

Санкт-Петербургский государственный университет
Математико-Механический факультет
Кафедра системного программирования

15 июня 2015г.

# Примеры

- Встроенный SQL в C#

```
private void Go (int cond){
string columnName = cond > 3 ? "X":(cond < 0 ? "Y":"Z");
string queryString =
                "SELECT name" + columnName + " FROM table";
Program.ExecuteImmediate(queryString);
}
```

- Динамически генерируемый HTML в PHP-программах

```
<?php
    $name = 'your name';
    echo '<table>
        <tr><th>Name</th></tr>
        <tr><td>'.$name.'</td></tr>
        </table>';
?>
```

# Область применения

- Реинжиниринг программного обеспечения
  - Анализ и трансформация систем, использующие строковые выражения
- Поддержка строковых выражений в IDE
  - Статический поиск ошибок
  - Подсветка синтаксиса
  - Рефакторинги

# Существующие подходы

Проверка корректности программ, получающихся в результате использования строковых выражений:

- Проверка включения языков
- Проведение лексического анализа и синтаксического разбора компактного представления множества динамически формируемых строковых выражений

# Обзор существующих решений и аналогов

- Java String Analyzer
  - регулярная аппроксимация строкового выражения
- PHP String Analyzer
  - контексно-свободная аппроксимация строкового выражения
- Alvor
  - нет поддержки строковых операций, за исключением конкатенации, и циклов
- Алгоритм абстрактного синтаксического анализа Kyung-Goo Doh, Hyunha Kim, David A. Schmidt
- Курсовые работы Вербицкой Екатерины, Полубеловой Марины

# Постановка задачи

**Цель**: реализация инструмента для проведения лексического анализа динамически формируемых строковых выражений

- Реализовать механизм для лексического анализа выражений, формируемых с помощью циклов и строковых операций
- Сохранить привязку лексических единиц к исходному коду
- Реализовать генератор лексических анализаторов

# Лексический анализ строковых выражений

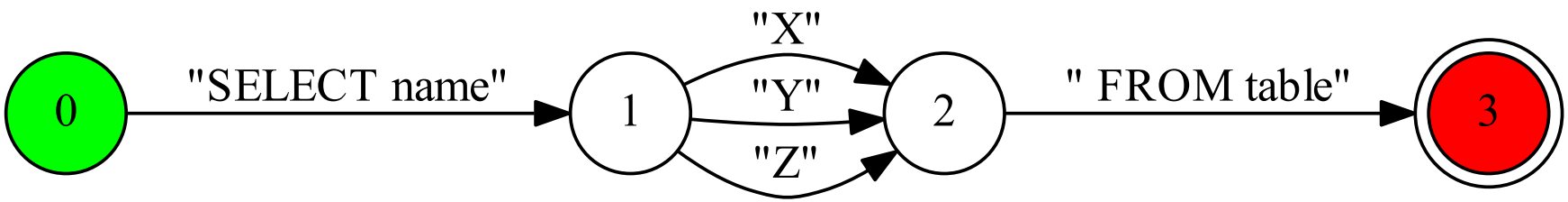
- На вход анализатору подается конечный автомат, полученный в результате аппроксимации строкового выражения
- На выходе получаем либо конечный автомат над токенами, либо список лексических ошибок. Токен содержит в себе:
  - идентификатор токена
  - конечный автомат — часть множества значений строкового выражения, которая выделена лексическим анализатором в данный тип токена

**Задача лексического анализа**: получение конечного автомата над токенами из конечного автомата над символами

# Пример

```
private void Go (int cond){
string columnName = cond > 3 ? "X":(cond < 0 ? "Y":"Z");
string queryString =
                "SELECT name" + columnName + " FROM table";
Program.ExecuteImmediate(queryString);
}
```

- Результат аппроксимации:

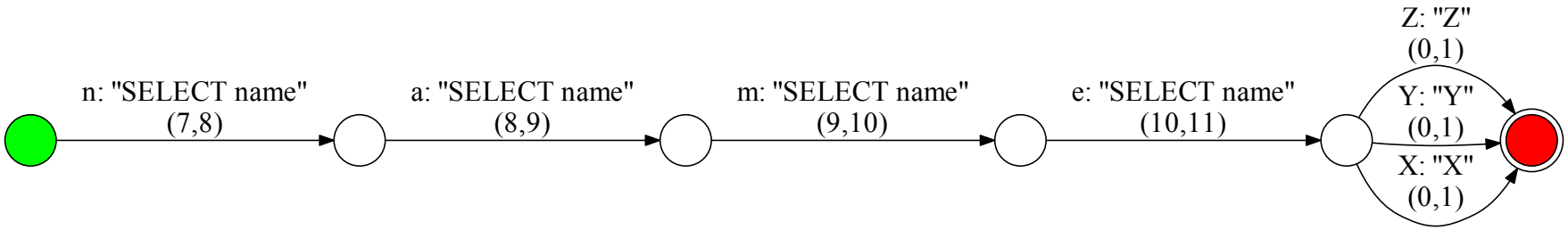



- Результат лексического анализа:

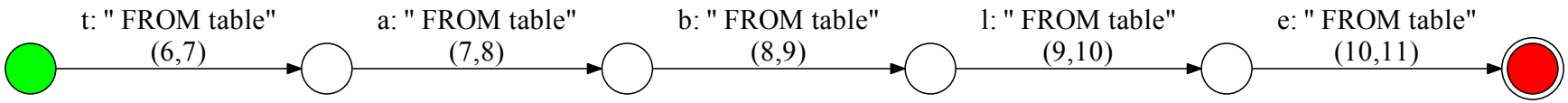
# Пример

- Результат лексического анализа:


- Конечный автомат первого токена `IDENT`:




- Конечный автомат второго токена `IDENT`:

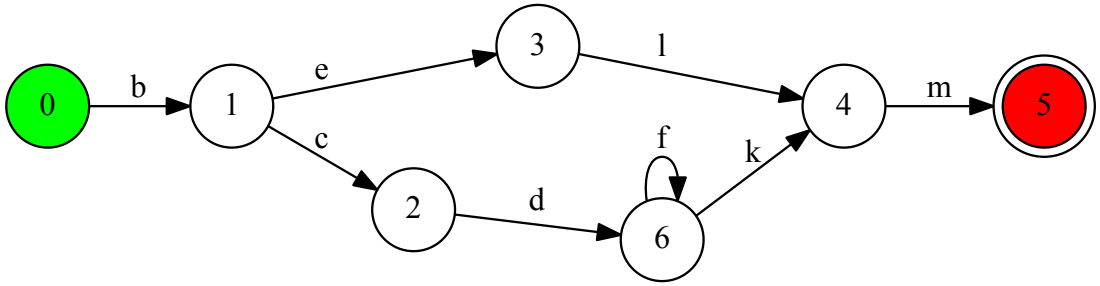


# Строковые операции

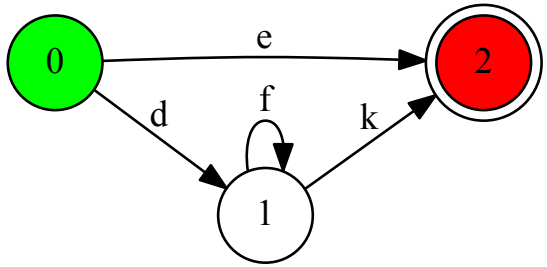
- ```
  string s = "SELECT nameX FROM tableY";
  s = s.Replace("SELECT nameX", "b");
  ```
- Многие строковые операции могут быть выражены через строковую операцию Replace, каждый аргумент которой является конечный автомат
- Для обработки строковой операции `Replace` использовался алгоритм, описанный в статье Fang Yu “Automata-based symbolic string analysis for vulnerability detection”
  ```

# Пример

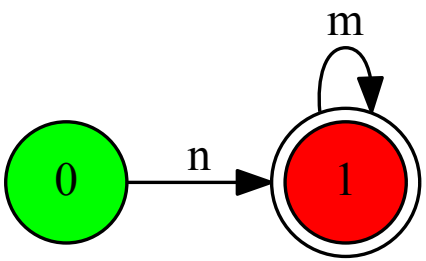
$M = \texttt{replace}(M_1, M_2, M_3)$



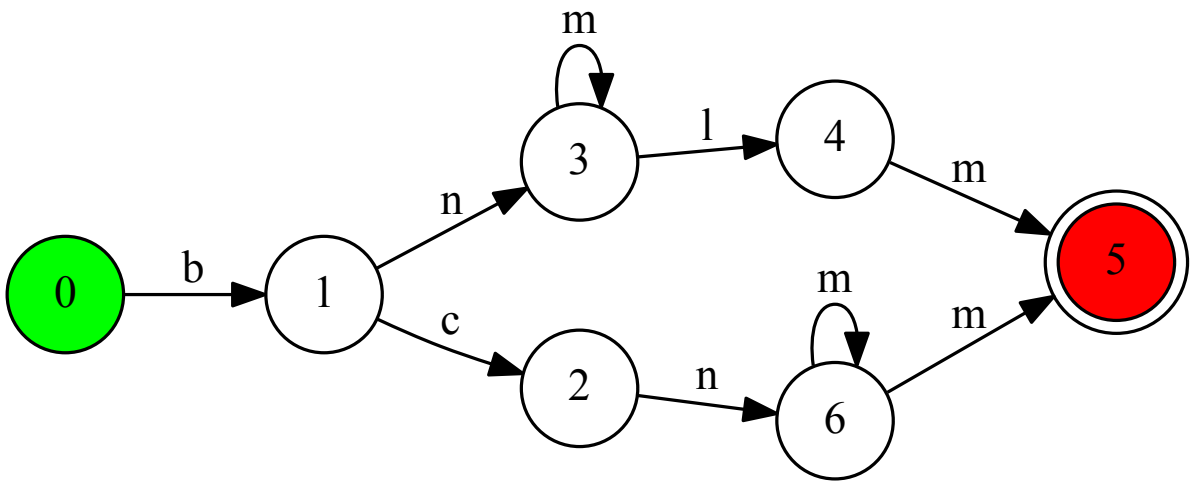

$M_1$




$M_2$



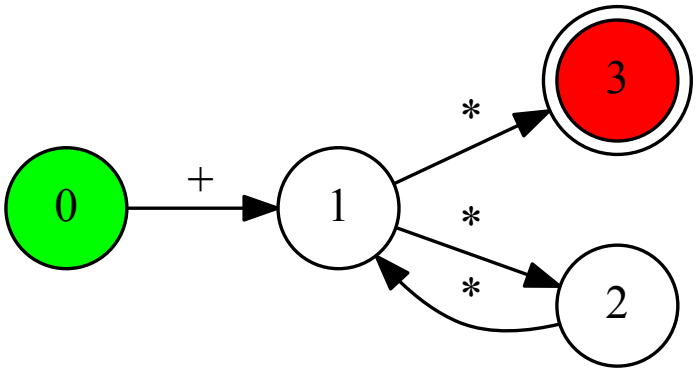

$M_3$




$M$

# Пример

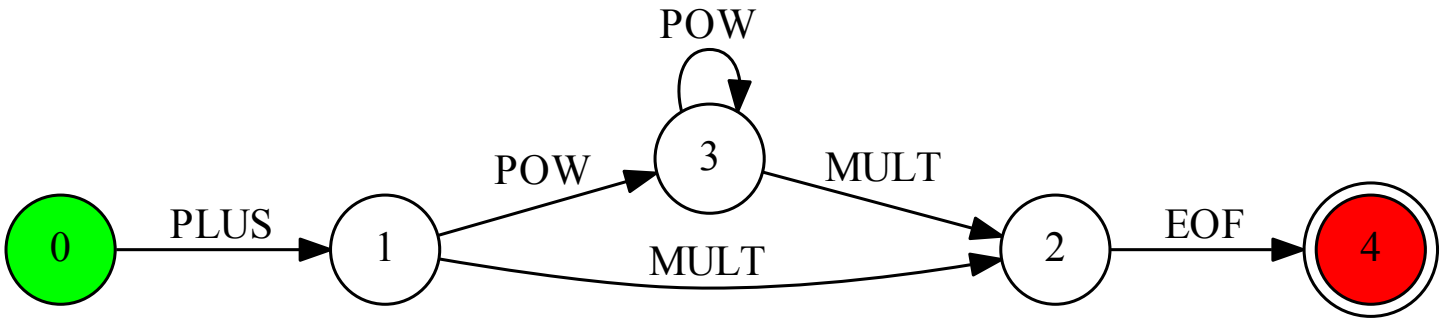
Входной граф:



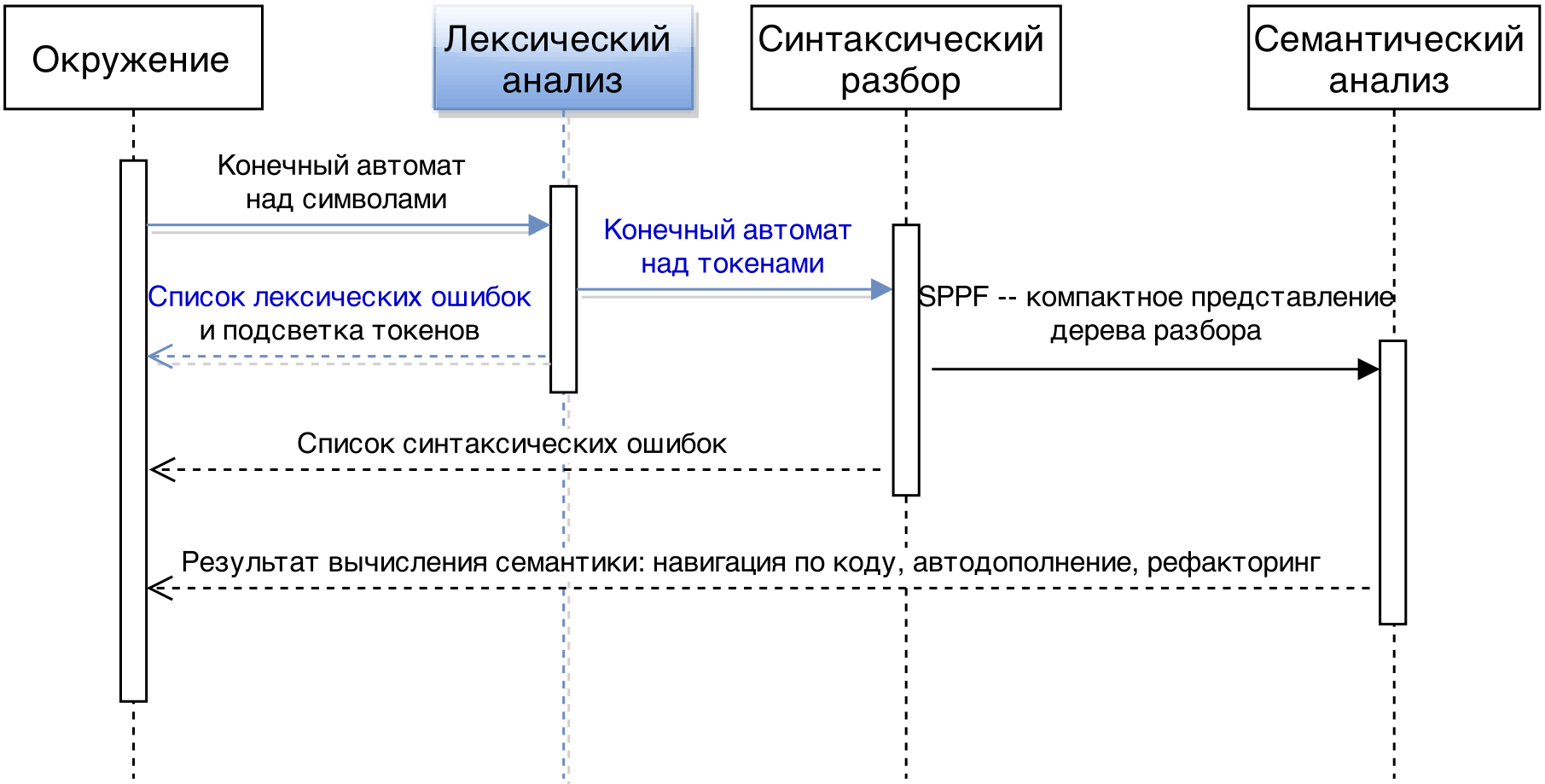

Спецификация:

```
PLUS : '+'
POW : "**"
MULT: '*'
```

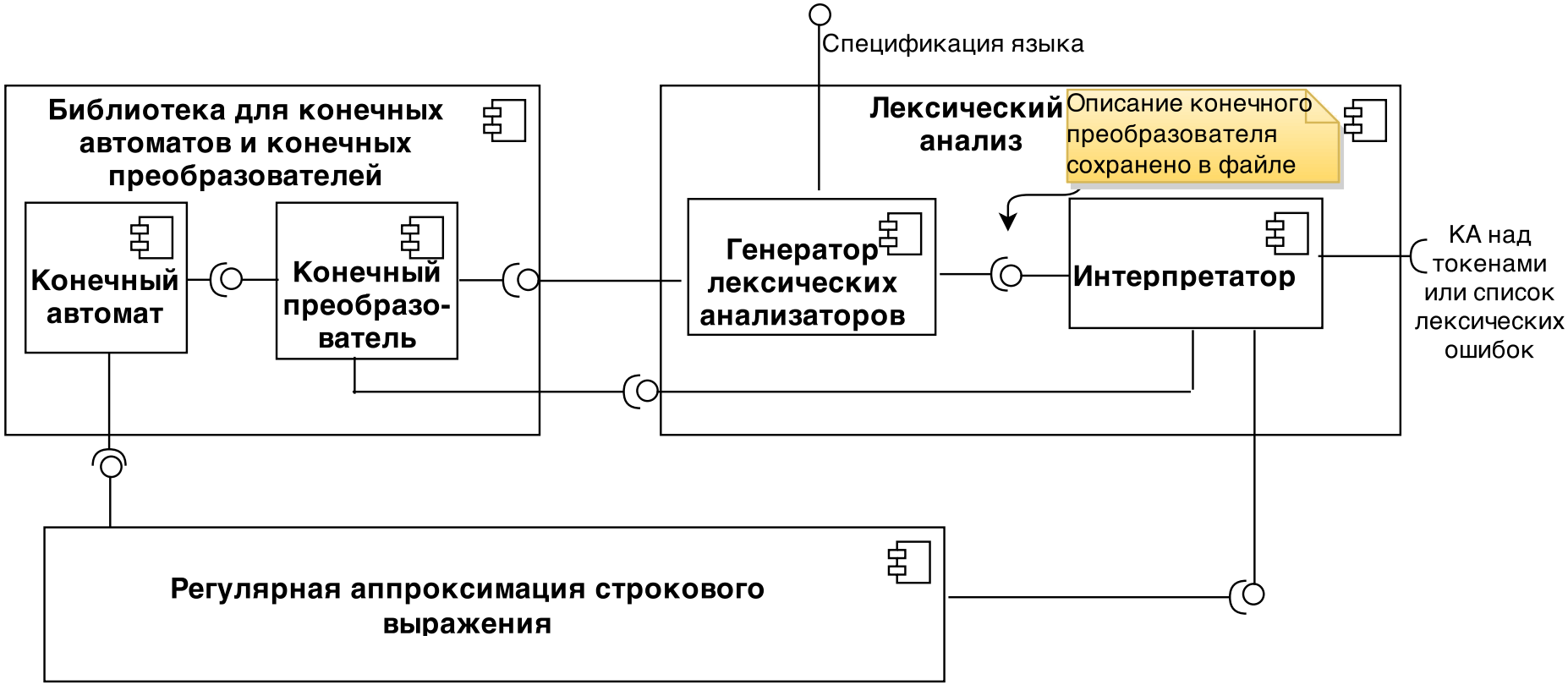
Результат лексического анализа:



# Инструмент YaccConstructor



# Архитектура инструмента



# Результаты

- Разработан алгоритм лексического анализа строковых выражений, формируемых с помощью циклов и строковых операций, сохраняющий привязку к исходному коду
- Реализована архитектура инструмента в рамках проекта YaccConstructor
- Проведена апробация полученного инструмента
- Результаты представлены на конференции CEE-SECR-2014
- Публикация “Инструментальная поддержка встроенных языков в интегрированных средах разработки” (ВАК)