Оптимизация алгоритма лексического анализа динамически формируемого кода

Автор: Александр Байгельдин, студент СПбГУ
Научный руководитель: ст.пр. С.В. Григорьев

Санкт-Петербургский государственный университет
Кафедра системного программирования

26 апреля 2016г.

Динамически формируемый код

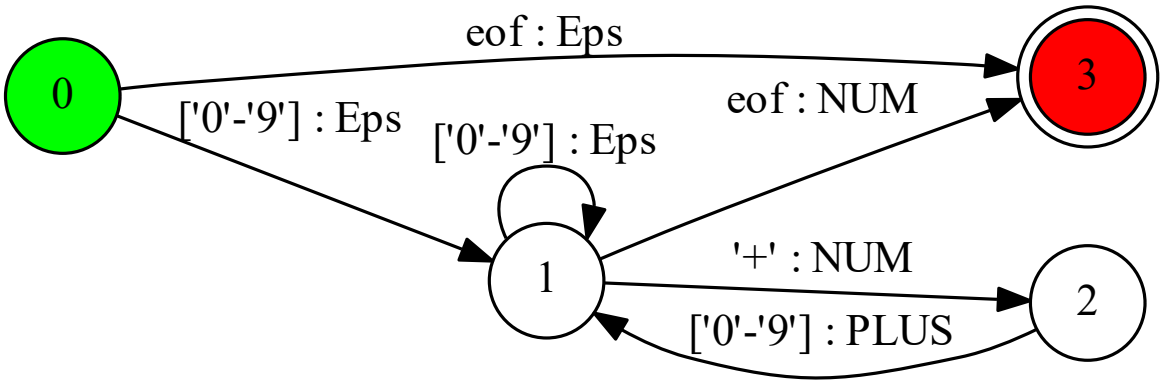
```
1  private void Go(bool cond)
2  {
3    string tableName = cond ? "Sold" : "OnSale ";
4    string queryString =
5        "SELECT ProductID, UnitPrice, ProductName "
6            + "FROM dbo.products_" + tableName
7            + "WHERE UnitPrice > 1000 "
8            + "ORDER BY UnitPrice DESC;";
9    Program.ExecuteImmediate(queryString);
10 }
```

Условные выражения, циклы, строковые операции (replace, concat)

Лексический анализ

В классическом лексическом анализе применяются конечные преобразователи (Finite State Transducers)

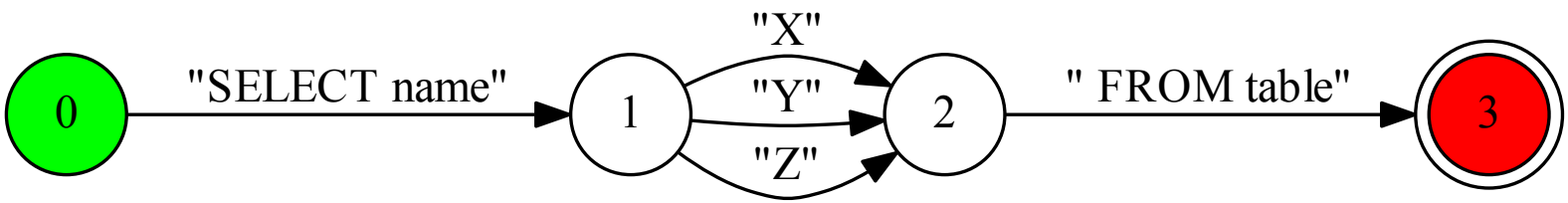
```
rule token = parse
| ['0'-'9']  { Some(NUM(gr)) }
| '+'  { Some(PLUS(gr)) }
```



Регулярная аппроксимация

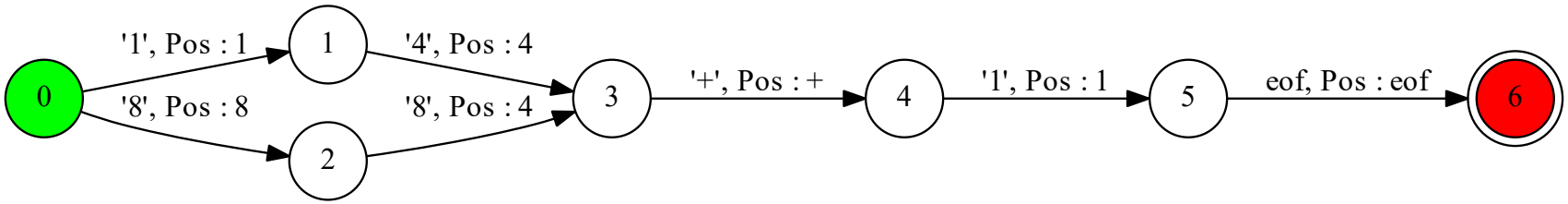
Для динамически формируемого кода можно построить регулярную аппроксимацию

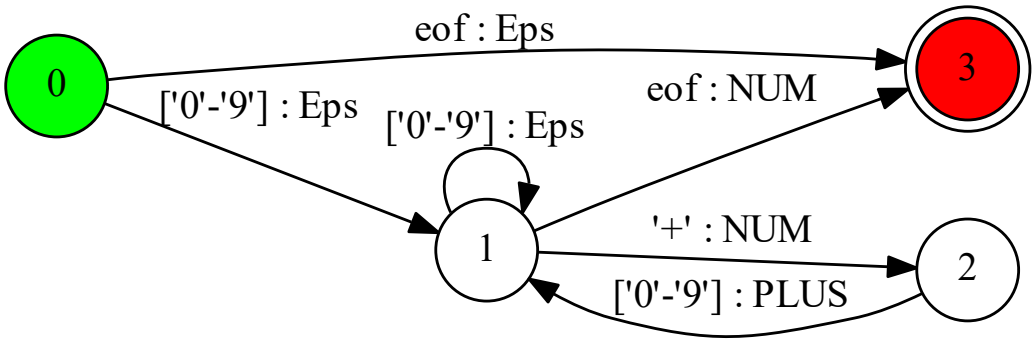
```
1 private void Go(int cond){
2   string columnName = cond > 3 ? "X" : (cond < 0 ? "Y" : "Z");
3   string queryString = "SELECT name" + columnName + " FROM table";
4   Program.ExecuteImmediate(queryString);}
```

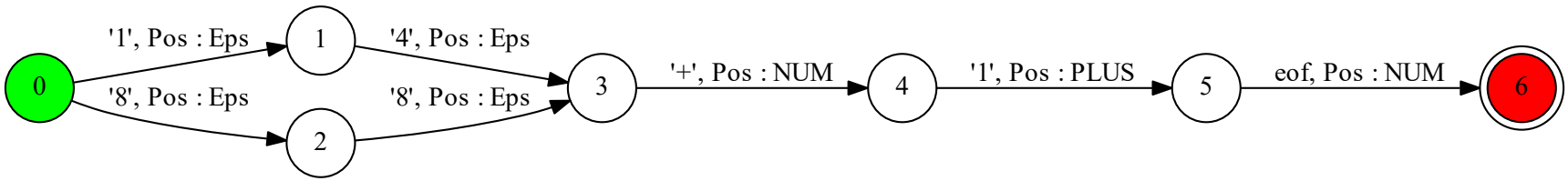


Композиция FST

Важной частью лексического анализа динамически формируемого кода является операция композиции FST ($\mathbf{T} = \mathbf{T1} \circ \mathbf{T2}$)







YaccConstructor

- YaccConstructor — исследовательский проект в области лексического и синтаксического анализа
- В YaccConstuctor для лексического анализа динамически формируемого кода применяется подход, в основе которого лежит построение регулярной аппроксимации и композиция FST
- Реализованный в YaccConstructor алгоритм композиции FST обладает недостаточной производительностью

Постановка задачи

Целью работы является исследование возможности улучшения производительности лексического анализа динамически формируемого кода

Задачи:

- Исследовать различные алгоритмы композиции FST
- Реализовать наиболее оптимальный алгоритм композиции
- Сравнить производительность реализаций текущего и выбранного алгоритмов

Решение

- Временная сложность текущего алгоритма:

$$O(V_1 * V_2 * D_1 * D_2)$$

где E — число ребер, V — число вершин, D — максимальное количество исходящих ребер

- В текущем алгоритме образуются недостижимые вершины, которые приходится удалять
- Временная сложность выбранного алгоритма:

$$O(V_1 * V_2 * D_1 * (log(D_2) + M_2))$$

где M — степень недетерминированности

- В выбранном алгоритме недостижимых вершин не образуется

Измерения

Кол-во вершин	Кол-во ребер	Время работы текущего алгоритма (мс)	Время работы выбранного алгоритма (мс)	Ускорение (раз)
700	2652	17215	4885	3.5
74	4452	2964	687	4.0
711	1766	15451	2565	6.0
250	738	8708	1414	6.0
128	374	2976	411	7.5
31	195	279	36	8.0

Технологии

- F# — язык семейства .NET с уклоном в функциональную парадигму
- YaccConstructor — исследовательский проект в области лексического и синтаксического анализа
- QuickGraph — библиотека .NET для работы с графами

Результаты

- Исследованы различные алгоритмы композиции FST
- Найден алгоритм, обладающий наилучшей производительностью
- Произведено сравнение производительности реализаций текущего и выбранного алгоритмов